Modelling and designing IT-enabled service systems driven by requirements and collaboration

Yong Peng

N° d'ordre 2012ISAL0024

Année 2012

Thèse

Modelling and Designing IT-enabled Service Systems Driven by Requirements and Collaboration

Présentée devant
L'institut national des sciences appliquées de Lyon

Pour obtenir
Le grade de docteur

Formation doctorale : Informatique

École doctorale : École Doctorale Informatique et Mathématiques

Par
Yong Peng
(Ingénieur en Informatique)

Soutenue le 22 mars 2012 devant la Commission d'examen

Jury

TATA Samir	Professeur à TELECOM SudParis	Président
VALLESPIR Bruno	Professeur à l'Université Bordeaux 1	Rapporteur
FRONT Agnès	Maître de Conférences HDR à l'Université Pierre Mendès France de Grenoble	Rapporteur
BENSLIMANE Djamal	Professeur à l'Université Claude Bernard	Examinateur
BENALI Khalid	Maître de Conférences HDR à l'Université de Lorrain	Examinateur
BADR Youakim	Maître de Conférences à l'INSA de Lyon	Directeur de Thèse
BIENNIER Frédérique	Professeur à l'INSA de Lyon	Co-directrice de thèse

Laboratoire de recherche: Laboratoire d'Informatique en Image et Systèmes d'information (LIRIS)

Résumé

Comparé aux services traditionnels du secteur tertiaire, les services facilités par les technologies de l'information et des communications (ITeS, à partir du sigle en anglais, IT-enabled Services) suscitent un intérêt croissant de clients et fournisseurs d'une part du fait de l'automatisation des processus et d'autre part grâce aux nouveaux canaux de communication (Internet, réseaux mobiles, …) que ces services supportent. De ce fait, les ITeS co-créent de la valeur ajoutée due à la collaboration entre les clients et les fournisseurs lors de la conception et la livraison de services. Cet enrichissement des services traditionnels conduit à une remise à plat des méthodes actuelles de conception de biens et de services. En effet, elles ne permettent pas de répondre aux exigences imposées par ce contexte de collaboration multidisciplinaire qui intègrent les entreprises, les technologies de l'information et de la communication et les acteurs sociaux. Les caractéristiques intrinsèques des services (à savoir, l'intangibilité, l'inséparabilité, la périssabilité, la simultanéité) et leur nature sociotechnique requière à la fois une méthodologie de conception globale dirigée par les exigences des clients en vue de leur satisfaction et une approche systémique prenant en compte la dimension collaborative, le cycle de vie des services et les changements organisationnels, métiers et technologiques.

Pour faire face à ces enjeux, nous proposons une méthodologie descendante pour modéliser et concevoir un système de services dirigé par les exigences des clients et supportant la collaboration entre tous les acteurs afin de permettre la co-création de ce système. Notre méthodologie repose sur une approche pluridisciplinaire et offre un ensemble de modèles interconnectés (modèle de référence de service, modèle d'exigence et modèle de collaboration) ce qui permet d'une part de donner de la flexibilité au système et de la rendre adaptable en cas de changements des exigences et d'autre part de supporter la collaboration entre tous les acteurs. Le modèle de référence offre une description des différentes dimensions du système de services (ontologique, caractéristiques et systémique) et explicite ainsi les connaissances liées aux domaines différents. En se basant sur le modèle d'exigences, les besoins du client sont spécifiés dans un langage commun et compréhensible par tous les acteurs. Ceci permet leur propagation dans tout le cycle de vie de service et leur diffusion à tous les acteurs. Le modèle de collaboration préconise une approche guidée par les données - une approche opposée aux processus métiers collaboratifs traditionnels - ce qui favorise l'interopérabilité technique et sémantique et augmente la stabilité du système face aux changements. Enfin, La collaboration s'appuie sur les canaux de communication qui engendrent des flux d'objets métiers (Business Artifacts) selon lesquels des règles d'affaires sont générées afin d'invoquer les composants logiciels sous-jacents.

Mots-clés: système de service – IT-enabled service – architecture d'entreprise - collaboration – Modèle de référence– ingénierie des exigences – interopérabilité – approche descendante – ontologie – SWRL règles métiers – business artifact – Science de services – SBVR – co-creation de valeur

Abstract

Compared to traditional business services, IT-enabled services provide more value to customers and providers by enabling traditional business services with Information and Communication Technologies (ICT) and delivering them via e-channels (i.e., Internet, Mobile networks). Although IT-enabled service systems help in co-creating value through collaboration with customers during service design and delivery, they raise challenges when we attempt to understand, design and produce innovative and intelligent IT-enabled services from a multi-disciplinary perspective by including businesses, technology and people for value addition and increasing benefits. Due to their social-technical nature and characteristics (i.e., Intangibility, Inseparability, Perishability, Simultaneity), IT-enabled services also lack common methods to systemize services driven by customer requirements and their satisfactions and co-produce them through ad-hoc collaboration.

In this thesis, we propose a middle-out methodology to model, design and systemize advanced IT-enabled service driven by customer requirements and collaboration among all actors to jointly co-create service systems. From a multi-disciplinary perspective, the methodology relies on a multi-view models including a service system reference model, a requirement model and a collaboration model to ensure system flexibility and adaptability to requirement changes and take into account joint efforts and collaboration of all service actors. The reference model aims at a multi-disciplinary description of services (ontological, systematical and characteristic-based descriptions), and formalizing business knowledge related to different domains. As for the requirement model, customer needs are specified in common expressiveness language understandable by all service actors and made possible its top-down propagation throughout service lifecycle and among actors. The collaboration model advocates a data-driven approach, which increases business, technical and semantic interoperability and exhibits stability in comparison to business processes centric approaches. Finally, the collaboration hinges on delivery channels expressed as data flows and encapsulating business artifacts as per which business rules are generated to invoke underlying software components.

Keywords: service system – IT-enabled service – Knowledge-intensive business service – enterprise architecture - collaboration – reference model – requirement engineering – interoperability – top-down – middle-out – ontology – SWRL business rules – business artifact – service science – SBVR – OWL – value co-creation

Acknowledgements

First of all, I would like to thank my doctoral supervisor, Dr. Youakim Badr, for offering me this interesting dissertation topic. I appreciate his excellent personal and scientific support and supervision without which my thesis would not have been brought to fruition.

I also appreciate the help of Professor Frédérique Biennier and her scientific support. I have greatly benefited from frequent and inspiring discussions with them. I was extremely fortunate to have had the opportunity to work alongside them during the three-and-a-half years I dedicated to my research and the writing of this thesis.

Additionally, Professor Bruno Vallespir and Dr. Agnes Front deserve my grateful thanks. They spend their valuable time reviewing my dissertation and gave me numerous constructive suggestions. I am also grateful to Professor Djamal Benslimane, Professor Samir Tata and Dr. Khalid Benali who served as jury members of my thesis committee.

Further, I am grateful to the Chinese Scholarship Council (CSC) for granting me a scholarship for three-and-a-half-year period of study.

I also thank to my colleges in my laboratory: including Ziyi Su, Francis Ouedraogo, Wenbin Li, and Juan Li. Special thanks also go out to my parents for their encouragement and education.

Table of Contents

Table of Figures

Table of Tables

Chapter 1

General Introduction

Abstract: Compared to traditional business services, IT-enabled services provide more value by enabling traditional business services with Information and Communication Technologies (ICT) and delivering them via e-channels (i.e., Internet, Mobile networks). However, due to their social-technical nature, IT-enabled services lack of common understanding about how to integrate services, human, organization/enterprise, software, and businesses for value addition and increasing benefits. Systemizing IT-enabled services potentially provide guidelines to understand all these components and their interdependencies and to foster innovation based on multidisciplinary perspectives including people, technology and businesses. In addition, IT-specialists play an important role in IT-enabled services by collaborating with upstream service actors and downstream service actors to co-create the system. They also cooperate with business analysts and customers to specify customer requirements and design business services and its business model. They also interact with developers to specify and design software and communication infrastructures that support the expected customer business services. In this chapter, we introduce our research context and identify the main research question. We shed light on the need for a methodology to model and design advanced IT-enabled service systems driven by customer requirements and collaboration among all actors to jointly co-create the service system. We also advocate a middle-out approach to ensure system flexibility and adaptability to requirement changes taking into account joint efforts and collaboration of service actors from different disciplines.

1.1 Research Background

In goods-centric era or industrial era, manufacturing firms play a dominant role in the whole economy. Services are just viewed as embedded value in goods and serve for the delivery or the consumption of them. However, with the increasing importance of services in today's economy, services are placed in an overarching position compared to their counterpart – goods and as a result, more and more researchers and practitioners view modern economy as service economy or service-centric era (Vargo et al. 2008). Different from goods-centric era which states that manufacturers produce tangible goods and sell them to their clients, the service-centric era emphasizes on service providers that effectively offer customized services to satisfy customer needs by introducing customer inputs (such as customer knowledge, skills, belongings, body and assets) as indispensable inputs to co-create added-value services with customers. Tangible goods may also be used or exchanged during the service production (e.g., lease service), but they are just viewed as carriers or intermediaries of service delivery to enable or facilitate the service production and consumption.

Compared to goods, services are identified by their characteristics such as intangibility, heterogeneity, inseparability, customization, and perishability (Rathmell 1966, Zeithaml et al. 1985, Wolak et al. 1998). Intangibility denotes that services are not physical and they cannot be seen or touched. Heterogeneity reflects the high variability in service delivery (Zeithaml et al. 1985). Inseparability denotes that service delivery and service consumption processes are not separated. Customization reflects that customers may expect customized services and even the same customer may expect different services in different contexts or situations. Customization also implies large quantities of interactions, negotiation, and trade-offs between providers and customers to co-design services. Finally, the perishability means that a service cannot be stored or carried forwards for future use (Zeithaml et al. 1985). Due to these characteristics, practitioners encounter difficulties to understand, design and produce services with the productivity and efficiency as high as they produce tangible goods.

Further, customers and economists agree that is quite easy to measure the economic benefits when producing tangible products, whereas due to the intangibility of services, it is relatively difficult to measure service value. Rather than economic benefits as tangible cost, the value of a specific service depends on the extent to which the service satisfies the customer requirements. As result, the same service has different values depending on the customer needs. In addition, services are quite complex systems and of great diversities in the service sector, ranging from tradi-

tional labour-intensive services (i.e., health care services) to knowledge-intensive services (i.e., consulting services) and further to IT-enabled services (i.e., e-learning services). Consequently, it is difficult and sometimes impossible to work out a generic service model or a service designing method appropriate for all types of services.

In our research work, we mainly focus on Knowledge-intensive Business Service (KIBS) firms since they are viewed as the core impetus of technological and organizational innovation characterizing new trends in the service economy (Miles 2003). Some examples of KIBS firms include e-learning providers, consulting companies and banks just to mention a few. KIBS firms heavily rely on professional knowledge to support customer business processes, in which the key service participants may refer professional and knowledgeable workers, such as engineers, business experts, specialists, scholars, and accountants, to provide their expertise and ingenuity in order to serve customers. Compared to manufacturing firms or non-KIBS firms in the service sector, KIBS firms compete by the skills and knowledge of their employees to deliver a range of customized solutions to meet or exceed customer needs/expectation, thereby they play a pivotal role in today's knowledge- and information-based economy. Due to these characteristics, KIBS firms are considered as the most innovative firms in the service sector and their innovation is recently flourished with by Information and Communication Technologies. It is worth noting the following potential benefits of ICT to KIBS firms:

1- Due to the development of ICT, the delivery of services is quite different from that many years ago (from manual work to automation). Nowadays, various service providers expose and deliver their services via e-channels (i.e., Internet and Mobile networks) to their customers for the sake of cost and efficiency.

2- ICT also change the way people interact or collaborate with one another. For example, ICT create new business models and enable new types of organization (i.e., virtual enterprises, network enterprises, alliances, e-market, etc) and also help IT specialists to create value-added IT-enabled services by integrating/bundling various e-services. We are witnessing that various stakeholders communicate and collaborate with one another via Internet to achieve common goals or projects.

3- The application of ICT is also enabling the digital transformation of service activities into computational components, automated processes and communicating systems, and their ability to codify, transmit business knowledge and reuse them into new added-value services. In this sense, service-based enterprises become interconnected in a system-of-systems structure by which they collaborate with one another for value co-

creation by applying ICT and mobilizing internal and external computational resources. In other words, KIBS tend to expose their traditional business services as computational features by means of IT-enabled services that could be integrated or bundled together through efforts and collaboration of various service actors into advanced and complex services delivered via e-channels for value addition.

The integration of ICT and KIBS services results in innovative IT-enabled services. Different from e-services that are limited to traditional services delivered via e-channels (Javalgi et al. 2004), IT-enable services are extended by their economic-social-technical features and also involve the close collaboration among customers, business experts, IT specialists, and developers and integrating businesses, technology and people.

Generally, IT-enabled services are viewed as traditional services (i.e., traditional learning, traditional commerce, etc) enabled by ICT technologies (Udaipur 2000). For example, e-learning services deliver traditional learning services via e-channels to make students freely study anytime and in any location. This may provide more value than traditional learning services that dominantly provide learning services at regular time and in specified classrooms. The general understanding on IT-enabled services is illustrated in Figure 1.1. Some inspiring examples for IT-enabled services involve call centres, financial services, insurance services, e-commerce, e-learning and legal and consultancy services.

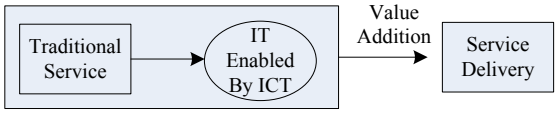


Figure 1.1 The general understanding of IT-enabled service (source: Udaipur 2000)

Meyer and Fahnrich (2009) provide a similar definition and view that IT-enabled services are being partly software and partly services. They categorize three kinds of IT-enabled services by differing degrees of interaction between services and information technology: respectively 1) IT accompany services, 2) IT supported services, and 3) hybrid solutions. IT accompany services refer to services that use software, which have the lowest interaction degree. For example, a teacher relies on an IT-infrastructure to provide e-learning services. The second category refers to services supported by software such as data management and call centre whereas the third category reflects the fact that the boundary of services and software is blurring and they are subject to functioning as a whole for value addition.

IT-enabled services add value to both service providers and customers. On the basis of customer requirements, various service participants just have to focus on their core businesses and IT specialists will provide

solutions to bundle together various autonomous and self-organized IT-enabled services that some of them might be distributed to create benefits without managing each of them. This could greatly improve their design efficiency and reduce their communication costs. As for customers, IT-enabled services also render services more valuable compared to traditional business services since they can freely access and consume IT-enabled services via e-channels and can also easily collaborate with service providers to co-design IT-enabled services.

Different from approaches in traditional product design or software engineering where various product components or software components can be easily integrated or composed together, IT-enabled services refer to appropriate arrangements or interactions of human and ICT, and the management of complex relationships between IT infrastructures and traditional services for value co-creation. In other words, IT-enabled services have to elaborate how behaviours of service actors influence the service design and innovation and how various service actors collaborate with one another to reach an agreement on the service perception and service expectation. They also have to articulate how to represent the interactions among human, software and hardware systems to enable the bundling of various IT-enabled services for value addition.

In order to fully explore the social-technical nature of services, especially IT-enable services and to improve the efficiency of designing IT-enabled services with high innovation rate, on May 17, 2004, IBM Research and Business Consulting Services (BCS) brought together academic participants from the various research fields to examine the changing business environment and to explore the case for the development of Service Science Management and Engineering (SSME), namely Service Science for short, as a new academic discipline (IBM 2004). SSME targets at exploring systematic models and approaches to design innovative and customized services from multiple disciplines, such as Management, Economy, Social Science, Computer Science, and Engineering (Spohrer and Maglio 2008, Maglio et al. 2006).

Meyer and Fahnrich (2009) discuss the need of systematic approaches to explore innovative services due to the complexity of managing the relationships between IT infrastructures and services. Service systems are generally defined as value co-creation between service providers and their customers (Vargo et al. 2008). Service systems stress on the collaboration process of various service stakeholders (e.g., people, organizations, systems, etc) who work together to co-create value. Generally speaking, a service system comprises human or organization, resources, technology, shared information (e.g., languages, laws, methods, knowledge, measures,

etc) (Spohrer et al. 2007). In the context of IT-enabled services for KIBS firms, a service system is viewed as an economical-social-technical system, in which human, especially professional experts, play an important role and customers take part in all aspects and levels of decision makings throughout the whole service lifecycle (broadly from requirement identification, to service design, to service implementation, and at last to service evaluation).

In such context, our research background is limited to exploring systematic approaches to create IT-enabled service systems for customized, context-aware and innovative IT-enabled services for KIBS firms since systematic approaches provide the following benefits to service design:

1- Help in understanding how service system components influence one another within a whole (Jackson 2003).

2- Due to the decomposability of a system, the systematic view makes possible to decompose a complex design problem into a set of relatively easier sub-problems.

3- Provide an effective way to help various service actors to identify the big picture of an expected service and its components and also to help to innovate in the service through their collaboration (Kim and Nam 2009).

4- Make possible to describe complex interdependencies between service system components as a set of interactions.

5- Integrate various IT-enabled services in a system-of-systems value chain for value addition.

1.2 Research Questions

Although IT-enabled service systems are beneficial to integrate various services in a system-of-systems value chain and help in co-creating value through collaboration with customers, they raise challenges when we attempt to understand, design and produce innovative and intelligent IT-enabled services from a multi-disciplinary perspective by including businesses, technology and people (i.e., customers). In this research context, we mainly identify four interrelated research questions that are important to design IT-enabled services driven by customer requirements and their satisfactions, and co-produce with service actors through collaboration during service design and delivery:

How to share a common understanding regarding services and their systems among various service actors and deal with their divergence due to how differently they conceive, design and implement them from different perspectives?

Due to the economic-social-technical-systematic nature of IT-enabled service systems, they involve diverse service actors from different

disciplines such as service engineers, IT professionals, businesses, developers, regulators, and customers. These service actors may use different models and languages and have different views on service systems when they intend to collaborate with one another for a common goal (value co-creation). Their interests are quite diverse and divisive and their interdependencies are complex and behaviours are dynamic and unpredictable, which satisfies the identified wicked problem that "*defy conventional approaches to understanding, planning, design, implementation and execution because: 1) the stakeholders interests are so diverse and divisive, 2) interdependencies are so complex and so little understood, and 3) behaviours are so dynamic and chaotic (unpredictable)*" in designing an enterprise information system by Gall et al. (2010) in Gartner Group.

Many research initiatives attempt to develop a common understanding of collaboration in services during their design and consumption processes. However, service actors still encounter barriers in practice during their collaboration to co-create services and seek for a common understanding on service systems suitable for the diversity of their disciplines. Ontological-based models, which are widely used to represent and formalize knowledge in a domain of interest, allow actors to deal with semantic ambiguities among disciplines (Smith 1998, Jurisica et al. 1999, Touzi 2007). In his work, Baida (2006) presents, for exemple, an ontology model for services in order to formalize knowledge from the business and technical domains taking into account both providers' and customers' perspectives for automating service bundling. Although ontologies are widely used to formalize knowledge on service actors (i.e., business rules, vocabulary, competencies, …), they are not yet explored to capture service systems and formalize knowledge on service system components, their interdependencies from a systematic perspective.

How can we develop an appropriate design methodology for IT-enabled services taking into account, service characteristics, requirements and customer satisfactions, as well as mobilizing various service actors to co-design and co-create IT-enabled services?

Top-down or bottom-up approaches are widely adopted to architect information systems or software systems in traditional Enterprise Architecture (EA) Frameworks (i.e., Information System Architecture (ISA) (Sowa and Zachman (1992), The Open Group Architecture Framework (TOGAF) (TOGAF 2009), and Federal Enterprise Architecture Framework (FEAF) (FEAF 1999), etc). Traditional top-down approaches for EA emphasize on the clear understanding of business strategies and the abundant knowledge about how to create an information system in the beginning of design stage (Wheeler 1979, Sabatier 1986). These approaches also high-

light the dominant role of designers during the design of enterprise information systems. Designers usually restrict customers' participation to the requirement analysis and later during the evaluation phase, and exclude them from intermediary phases, which eliminate their roles in the value co-creation. Moreover, traditional top-down design approaches "*worked well when applied to complex, fixed functions — that is, human artefacts, such as aircraft, ships, buildings, computers and even EA software. However, it works poorly when applied to an equally wide variety of domains because they do not behave in a predictable way*" (Gartner Group 2009). IT-enabled service system is driven by collaboration in which various service actors collaborate with one another to co-create and consume the service. Their collaboration activities are quite dynamic, and emergent, and they cannot be defined in advanced. Top-down approaches are not flexible enough to encompass dynamic (ad hoc) collaboration and not flexible enough to mobilize external and internal business resources and adapt them to changes during service design and service consumption. Conversely, the bottom-up design approaches start from a quite small part of the business or system components in an organization and then gradually scales it out to the whole organization (i.e., composing various systems together) (Sabatier 1986, Kautz et al. 1994). However, due to the lack of common understanding among developers and business managers, the improvements applied to the initial business (i.e., system composition) may be quite difficult to be extended to the whole organization in a bottom-up manner. By composing various small parts of businesses together through technological-based solutions (I.e., middleware), business managers find out that their underlying information systems are not aligned with their day-to-day businesses. Traditional design approaches (or methods) are not quite appropriate for IT-enabled service systems since they mainly rely on top-down or bottom-up approaches. In this sense, IT-enabled service systems require a middle-out approach for designing services to enable and facilitate the collaboration of diverse service actors. A middle-out approach generally initiates from a much higher level than the implementation/technical level but lower than the business strategy level (Raccoon and Puppydog 1998). It is an appropriate level to "*support changes, to innovate and to be better tailored to downstream needs, better communicated upstream and more acceptable by both up and down stream*" (Parag and Janda 2010). Parag and Janda (2010) also view that "*middle-out is not necessarily an alternative to the top or bottom but rather an additional, supportive, and maybe more effective way of delivering change*".

Since the middle level is neither quite strategic nor technical, business-level workers can refer to the middle level to manage their busi-

nesses and adapt them to changes. Technology-level workers can also refer to the middle level to develop software to support business services whereas customers can rely on the middle level to easily communicate with business-level and technical-level workers with less technical or business vocabularies to express their requirements.

IT-enabled services also require a system reference model to facilitate the collaboration among various service actors in a systematic process of designing and to make them focus on respective core business and the delivery channel rather than mutual businesses.

How to concretize imprecise, unstructured, and unclear customer needs into precise, structured, and clear customer requirements to deal with the gaps 1) among various service actors that use different models and/or languages to collaborate with one another and 2) between expected service by the customer and perceived service by the provider?

Customers play a significant role in IT-enabled service systems since they take part in all levels of the service design, which is a different approach in traditional information systems where customers partially take part in the system design through front-office applications (i.e., CRM). In this sense, how to exactly and clearly identify customer requirements and input is quite critical to design and consume services. Building IT-enabled service systems needs a rigorous engineering approach to identify and specify customer, business, and technical requirements through collaboration at each phase of the service lifecycle. Unfortunately, traditional requirement models (Rolland 1993, Schmitt 1993, Pohl 1993, Lee 1996, Toffolon and Dakhli 1999, Donzelli 2004, Baida 2006) are mainly developed for software or/and system developments. They rise challenges if they are applied to IT-enabled service systems; customers encounter problems to exactly express or identify appropriate requirements to take full advantages of Information Technologies and service benefits in order to increase the value of their expected services. Service actors may also use different models or languages, which will make them difficult to collaboration to co-specify requirements. There are gaps between services as they are perceived by the providers and services as they are expected by the customers. In this sense, any requirement model for services should be expressive to be understood by all service actors (customers, IT specialists, engineers, developers, …) and rigorous to be processed by computers (i.e., automation, software invocation, …). Particularly, IT specialists play important roles in integrating upstream service actors (i.e., customers, administrative officers and business analysts) and downstream service actors (i.e., developers, technicians).

How to represent exchanged knowledge and deal with ad hoc collaboration among service actors?

During collaboration among various service actors, the activities which have to be executed and their sequence are not often predetermined in advanced. Human workers, especially knowledgeable workers, do require flexibility in their activities that cannot be expressed by predefined processes. The service provider, that is to say the knowledgeable worker, first has to assess customer needs after which further actions have to be scheduled accordingly. In addition, customer satisfaction is the ultimate goal to be reached. However, the strategy to follow is unclear in the beginning. The service provider has a variety of options to reach the goal and the decisions made will gradually determine the appropriate strategy to deliver the service. Rather than being predefined and deterministic, the process of collaboration is adhoc and dynamic and evolve during service production and consumption based on customer decisions and input. Modelling ad hoc and dynamic collaboration activities reveal a challenge to design and implement service systems. In addition, in IT-enabled service systems, service actors mainly exchange explicit knowledge that tells them who should act, what should be done, and how to act. Documents, business records, notes and emails are a few examples of means to carry out data or knowledge between service consumers and providers. In the service delivery channel, the exchanged knowledge between consumers and providers are sometimes not well-structured. Therefore, in order to automate or semi-automate their collaboration, it is necessary to represent exchanged knowledge in explicit structures and build the collaboration based on these knowledge instead of predefined business processes.

In the domain of information system design or software system design, most work (i.e., Casati et al. 2000, Lammermann 2002, Issa et al. 2006, Ko et al. 2009, and Fujii and Suda 2009) rely on activity-based or process-centric modelling approaches to represent business processes. Each activity is usually supported by software applications, such as Web services, to automate its execution (Tidwell 2000). Web services have recently emerged as a powerful technology for integrating heterogeneous and distributed applications and increase interoperability among information systems. In this context, interoperability is generally achieved by encapsulating the access to applications with web services and interconnects them through composition mechanisms to build up business processes. The composition consists of matching the outputs of a Web service to the inputs of another web service that is sequentially after it. However, Web services are subject to predefining business processes as a set of sequential business activities. This makes businesses lack of flexibility and adaptability and does

not really match the way people do businesses (Cohn and Hull 2009). Some researchers (i.e., Nigam and Caswell 2003, Kumaran et al. 2008, Muller et al. 2008, and Narendra et al. 2009) use data-centric approaches to achieve interoperability in information systems. They model business entities as a set of business artifacts (business records or documents) and business processes are modelled as a set of interacting among artifact lifecycles. As per these initiatives, data-driven modelling proves to be useful to deal with interoperability issues and increase the system stability.

However, some high-level interoperability issues also have to be taken into account, for example, semantic interoperability and the interoperability among collaborative enterprises with different business strategies, organizational constraints, and IT infrastructures. Business data passing from one service actor to another is a core activity in any interoperability issue. Service actors always have difficulties in identifying the data they require from their partners due to lack of common understanding and restricted access to data resources. In addition, exchanged knowledge also has to consider privacy and securities issues. In different service systems, some collaboration activities recur throughout the whole service lifecycle. As a result, it is urge to rethink dynamic collaboration in services from a data-driven perspective and find means to effectively introduce collaboration patterns to describe recurred collaboration activities in order to reuse them in service design and delivery.

1.3 Research Approach and Major Contribution

In response to the abovementioned research questions, we follow a middle-out approach to design IT-enabled services. The design methodology relies on a set of inter-connected models to build the service system. These models also guide service actors to collaborate and propagate their requirements downstream to help in implementing the underlying infrastructure and applications and upstream to facilitate interactions with customers. Our approach intends to develop requirement-drive service system based on collaboration during design and delivery time. The big picture of our major contributions is illustrated in Figure 1.2, which includes the following models:

Service System Reference Model: provides different representations or views of the service system by illustrating their static structure (i.e., fundamental components and their relationships), as well as dynamic behaviours to show how to formalize business and technical domain knowledge in service systems, how various service systems are integrated in a system-of-systems value chain, how they exchange explicit business knowledge, and how they collaborate with one another for value co-

creation. The service system reference model is expected to address the second research question. The service system reference model includes three kinds of representations consisting of systematic view, ontological view and characteristics view. The systematic view statically describes interactions between providers and customers by presenting fundamental service system components (including service actors, resources, competencies, technologies, channels, and customer inputs) and their relationships. The ontological view introduces concepts, their relationships and dynamic behaviours related to service actors in order to provide a common understanding on how to deal with ambiguities in collaboration due various disciplines. Finally, the characteristics view suggests that the service system exposes the service in terms of its characteristics. Service characteristics mainly describe non-functional features of the service and to represent possible collaboration/partner relationships by matching service characteristics of a service provider to customer requirements characteristics at design time.

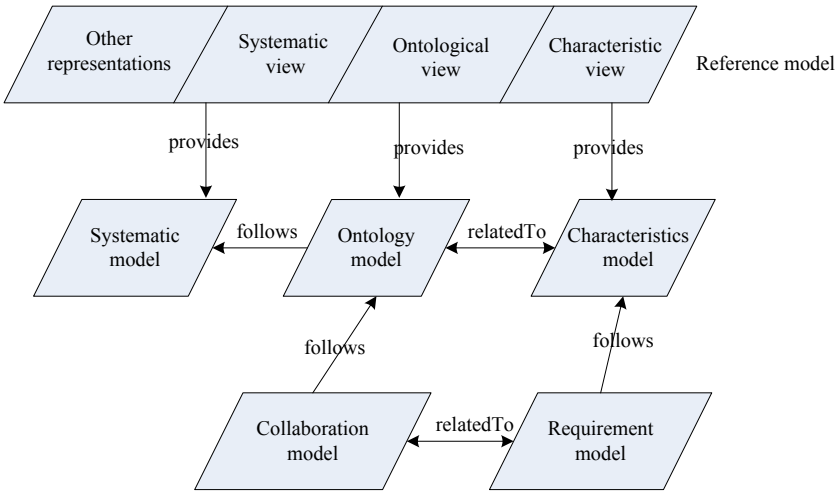


Figure 1.2 The big picture of our contributions

The Ontological Model: defines a common vocabulary that can be shared by various participants from different domains. It is composed of classes or generic concepts, individuals or instances, relationships between concepts, restrictions to impose constraints on these relationships, and axioms or rules to denote business facts/logics. The ontological model consists of a set of ontologies to denote concepts, relationships, restrictions and axioms for service system components and their underlying resources in order to provide service actors with a common understanding on the service system. As described by Noy and McGuinness (2001), the ontology allows to share common understanding of the structure of information among service actors and resources. It also enables the reuse of domain knowledge. Compared to

hard-coding assumptions, the ontology makes domain assumptions easier to understand and change.

Since a service system for KISB firms holds large quantities of collaboration between service actors, it desires for an ontological model to deal with their conceptual ambiguities, sharing of common understating of the structure of information, the reuse of domain knowledge, and the separation of domain knowledge from operational knowledge in order to facilitate knowledge exchanges and achieve interoperability.

The Characteristics Model: describes qualitative and quantitative features of IT-enabled services. At design time, a service system exhibits service characteristics to other actors. Service characteristics were coined to Lancaster (1966) who argues that both tangible goods and intangible services can be described by a set of characteristics that a good/service embodies. Saviotti and Metcalfe (1984) shows that manufacturers can obtain competitive advantages by offering users particular combinations of three sets of characteristics, one describing the technical features of the product (technical characteristics), one describing the services performed by the product (service characteristics), and one describing the methods of its production (process characteristics). In our work, we demonstrate how different service characteristics correspond to potential arrangements of its internal service system components. We generalize this concept to include resource characteristics, competencies characteristics, technology characteristics, and channel characteristics. Externally, a service system exposes its service characteristics to other service systems for potential collaboration and internally, the service system mobilizes its internal components to support its exposed characteristics. In this sense, describing services as a set of qualitative and quantitative features, and constraints provides benefits in the context of service actors by ensuring service quality at design time and measure service performance at the run time by evaluating if the delivered services achieve customer expected service characteristics. Thinking in terms of characteristics makes actors focus on their core businesses and their interactions rather than attempting to understand technical or domain specific terms related to other different disciplines.

The Requirement Model: deals with how to concretize unclear customer needs into specified and clear customer requirements in order to address the gap between service perception and service expectation. In addition, this model attempts to describe with inter-related structured languages (English-like language (SQML), business language (SBVR), technical language (SWRL)) to express customer needs and translate them to business rules and technical rules. As a result, different service actors (i.e., customers, business experts, IT specialists and developers) can understand

customer requirements and ensure their top-down propagation. The value of the requirement model also lies on its offering side-way propagation for IT specialists or other middle-level service actors. The requirement model is expected to deal with the third research question.

The Collaboration Model: As mentioned before, in traditional software systems or information systems, business processes are modelled as workflows of predefined activities each of which is implemented as Web services. Business processes are thus composed of Web services together by matching the output of one Web service to the input of its precedent Web service. However, the business process-driven modelling approach neither deals with dynamic and ad hoc collaboration processes nor achieves business interoperability. Conversely, our proposed collaboration model tends to represent exchanged business knowledge in terms of as business artifacts and to model dynamic and ad-hoc collaboration activities as a set of data-driven flows. Business artifacts are self-contained business records that comprise attributes, states and lifecycles (Nigam and Caswell, 2003). The artifact lifecycle describes possible transitions between artifact states. Changes in business artifacts are controlled by the artifact lifecycle. Each transition corresponds to the execution of one or more tasks that process the attributes and update the current states. From a business perspective, business artifact is manageable, analyzable, and easily understandable by business people.

The collaboration model is also extended with business rules (Business Rule Group 2001) to invoke computational resources and execute business logics.. In our context, business rules are introduced to invoke underlying software applications (i.e., Web services) and updating business artifact attributes and states. The benefit of the collaboration model lies on the fact that a service actor who needs to co-create value with other service actors will simply have to manage their collaborations by exchanging business knowledge via delivery channels rather than interconnecting processes and managing internal service system parts. The collaboration model is expected to deal with the fourth research question.

As shown is Figure 1.2, these various models are not isolated from one another but they are interrelated. Roughly speaking, the requirement model refers to the characteristic view to specify unclear and imprecise customer needs and transform them into clear and precise customer requirements. The collaboration model refers to the systematic view and the ontological view to enrich service systems and provide common understanding of collaboration among actors. The requirement model can be followed to identify service actors, business artifacts, and business rules to enable the collaboration model.

Due to the increasing complexity when more and more service actors take part in a service system, it becomes a tough task to design such complex architecture by integrating various IT-enabled services together and ensure semantic interoperability among different disciplines. In our work, adopt an system architecture by which service actors should focus on their relationships and model their relationships as interactions via interfaces to connect them rather than taking into account their internal structures or components. Gartner Group (2009) identifies a new approach to Enterprise Architecture (EA), which is aligned with our service system architecture. The EA has two characteristics: 1) "*architect the lines, not the boxes*": implying that focusing on the connection rather than internal complex components; 2) model relationships between businesses or components as a set of interactions via some set of formal or informal interfaces. In our context, the 'lines' are denoted as service characteristics and the 'boxs' are denoted as internal service system components. In addition, IT-enabled service systems are quite dynamic and collaborative and have system-of-systems structure, the hybrid thinking suggested by Gartner Group (2010a) helps to design such service systems by taking into account different service actors' views. Gartner Group (2010a) also advocates a hybrid thinking that "*integrates design thinking with other ways of thinking to produce successful outcomes to wicked problems by co-creating more meaningful, human-centered experiences*" in business transformation, innovation and strategy in order to design "*an ecosystem where variations abound, rather than an organisation, which is often monolithic*". We have developed our service system architecture with respect to the core idea of the Gartner Group (2009) to engineer IT-enabled service systems as follows:

1- At the design time, a service supported by an IT-enabled service system is exposed by its characteristics. Service characteristics express the utilities provided by the service and hide the complex internal parts of the service system. Customers refer to service characteristics to decide whether the expected service match their requirements.

2- At the runtime, various service actors collaborate with each other by exchanging business artifacts. The relationships among actors are concretized by interactions to exchange business artifacts, which make them focus on their delivery channels rather than dealing with their system internal structure and components.

1.4 Putting it All Together

Based on the big picture in Figure 1.2, we briefly and globally describe the service system reference model, requirement model and collaboration

model. These models are interconnected and help in building the general design process of creating a new service system.

The first step consists of specifying the IT-enabled service with its final characteristics to expose them to customers as a set of features and constraints on the service utilities. The service providers have then to organise their resources, technologies, competencies, and delivery channels in appropriate combinations to deliver the expected characteristics and match them with customer needs. The service reference model shows service actors how service components should be identified, accessed and connected together to create the added-value service. Following the requirement model, the features and constraints are specified with a formal structured language, called SRML (Service Requirement Management Language) to rigorously describe them and reason on service system components and their interactions. We conceive the SRML as an extension of the Quality Modelling Language (QML) (Frolund 1998), which is mainly used in software engineering to define Quality of Service (QoS) properties and allow software applications to replace old components with new components as long as the new components satisfy the QoS specifications. In a similar way, we argue that well-defined customer requirements would help service system designers to update their existing service systems, or replace old components with new components as long as the new components satisfy the customer requirements specification.

However, SRML is quite technical language since it applies constraints on service components making them difficult to be understood by non-technical actors (i.e., customers and requirement engineers). We enrich the SRML with a common structured language that has natural-like syntax to facilitate the collaboration of various service actors. To this end, we propose to translate all SRML expressions into SBVR (Semantic Business Vocabulary Rules) expressions.

SBVR (SBVR 2008, Hendryx 2005) is a formal structured language with natural-like syntax intended to be easily understood by customers. Business workers rely on the SBVR to identify and tune the requirements through collaboration with the customers and then validate them before forward them downward to the low level of the service system. The SBVR rules are also used to enrich service system ontology with facts about the expected service and identify business artifacts (business knowledge) that should be exchanged with customers during the collaboration phase. Based on the collaboration model, the SBVR rules are used later to generate domain specific rules, expressed in the Semantic Web Rule Language (SWRL) (Horrocks et al. 2004). These rules invoke computational resources (i.e., Web services) and update business artifacts. Each SWRL

rule encompasses knowledge about business artifacts and associates them with applications to process them based on their articat current values and their states. Since the SWRL rules are not embedded into information and communication applications, any changes in requirements are the customer level will propagate in a top-down manner and may result in changes in SWRL rules. This approach guarantees the adaptability of the delivered IT-enabled services in response to changes initiated by customers and/or IT-specialists.

As in KIBS firms, various service actors collaborate with each other by exchanging business knowledge (business artifacts). We thus rely on the collaboration model and the data-driven approach to design a collaboration framework for service actors by defining collaboration patterns at design time and develop context-aware rules to trigger them at the runtime. A collaboration pattern is composed of information describing a context, a generic problem, a goal, and general solution including business artifacts, ontology, and business rules updating artifact attributes and state transitions, etc). In our approach, the collaboration among service actors concerns each phase of the service lifecycle, which we divide it into five high-level phases: including the initialization phase, the situation assessment phase, the establishing agreement phase, the service design phase and the provisioning and evaluation phase. Each phase is also decomposed into various sub-phases that contain set of interactions each of which is modelled as a directed data flow from one service actor to another service actor and supported by information and communication applications. Supported by the requirement model, various service actors reach an agreement on the policies about their exchanged business artifacts and shared business rules.

1.5 Content Organization

As illustrated in Figure 1.3, the dissertation report is structured as follows:

Chapter 1 generally describes the research background, the research questions, the research approaches and our major contributions in this study.

Chapter 2 presents the state-of-the-art and covers research background related to our problematic. Firstly, it sheds the light on the term ‘service’ in different scientific fields. It depicts all challenges of designing IT-enabled services for KIBS. Secondly, the chapter discusses current approaches to systemize services and build a service system taking into account a requirement-based approach.

Chapter 3 presents our design method and a generic design process to create a service system. It also elaborates the service system refer-

ence model and explains its underlying different views namely, the systematic view, ontological view and characteristic-based view.

Chapter 4 presents a top-down requirement engineering approach and indicates how to refer to the service system reference model to concretize unclear, imprecise, and unstructured customer needs into clear, precise and structured customer requirements. Supportive to the top-down approach, the requirement model also intends to facilitate interactions among service actors belonging to the same discipline in order to concretize their requirements in an expressive language and propagate requirement changes throughout the top-down requirement approach.

Chapter 5 discusses how to model exchanged business knowledge to build a data-driven collaboration. It also discusses how to refer to the reference model to establish ad hoc and dynamic collaborations.

Chapter 6 demonstrates how to use our proposed models and approaches through a case study in the e-learning domain.

Chapter 7 illustrates the technical architecture and implements a prototype to simulate artifact-based collaboration among various service actors.

Chapter 8 summarizes our work and outlines future research directions.

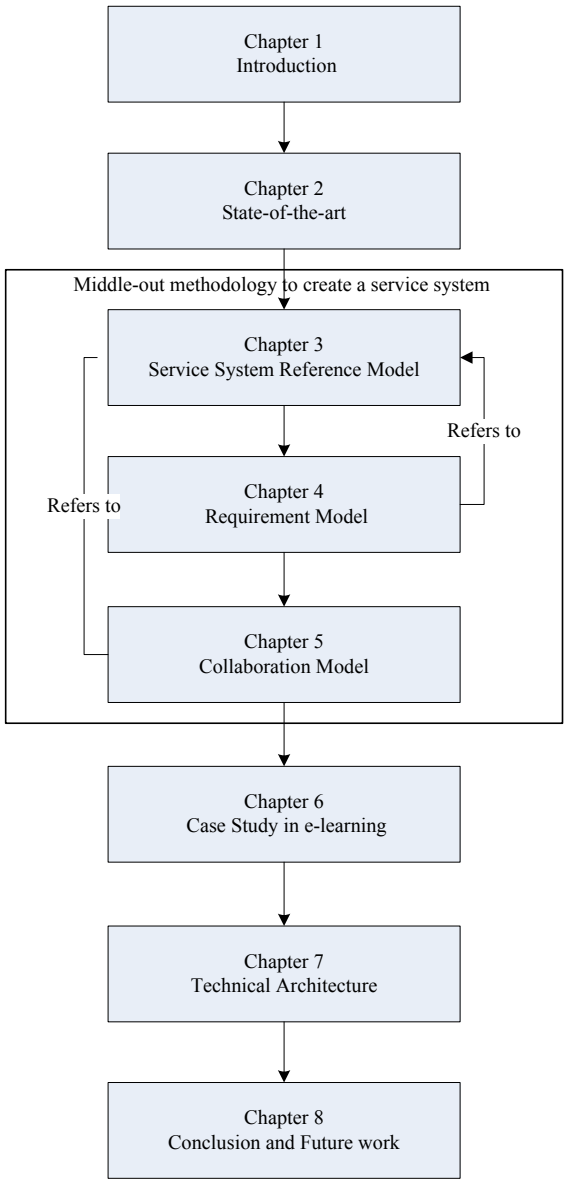


Figure 1.3 Content organization

Chapter 2

State of the Art

Abstract: Designing IT-enabled service systems requires multi-disciplinary design models and approaches due to their economic-social-technical nature. In this sense, this chapter seeks to explore the state-of-the-art regarding models and approaches used to design service systems. We at first discuss service interpretations in different disciplines to shed the light on the context by which we develop our contribution. We then discuss systematic approaches to design IT-enabled services and how current multi-disciplinary methods and/or models dealt with the challenges related to IT-enabled service systems. At last, we elucidate how these methods, models and approaches are not appropriate to answer our research questions presented in Chapter 1 and conclude with the need for a new design method convenient for IT-enabled service systems.

2.1 Introduction

In the service-centric and knowledge-intensive era (Vargo et al. 2008), service firms focus on their core business processes and increasingly propose their expertise as e-services to co-create value with their service partners.

This new business model intensively relies on ICT and has led to the emergence of virtual enterprises, e-marketplaces, e-services, online collaborations (B2B, B2A and B2C) and IT-enabled services. Also Technology-based KIBS, such as computer and engineering services, technological training and consultancy services, facilitate to generate innovation and improve service quality (Lengrand et al. 2002). In other words, today we are witnessing an extraordinary transformation of our economy into the service-centric logic that desires for models and approaches from multiple disciplines to facilitate the collaboration between customers and service providers for value co-creation throughout global interconnected service-oriented enterprises by applying ICT.

The service concept refers to the outcome that is received by the customer (Lovelock and Wirtz 2004). Roth and Menor (2003) list the alternative terms for the service concept including service offering, service package, and service or product bundle. The service contract is initially interpreted as the direct interactions between customers and providers and the design of a service may involve the front office where interactions happen and the back office that supports the services and interaction activities in the front office (Chase 1978) and then the concept is extended to customer participation (Shostack 1987) and finally extended to value co-creation during the whole service lifecycle (Sporher et al. 2008). Goldstein et al. (2002) indicate that service delivery systems include "*the structure (facilities, equipment, etc.), infrastructure (job design, skills, etc.) and processes for delivering the service*". Ponsignon et al. (2007) consider that a service delivery system is "*made up of multiple, interdependent service processes, which constitute a hierarchically-organized process architecture and an integrated design approach is required to ensure adequate coordination between processes within the whole service system*".

The design of IT-enabled services encounters challenges as we discuss in the previous chapter. In this chapter, we aim at providing models and approaches to create IT-enabled service systems for producing innovative and context-aware services. Due to multiple service definitions and the various use of the term 'service' in different domains, we discuss in Section 2.2 the need to agree on a clear understanding of this term in order to build up our contribution. In Section 2.3, we identify the main challenges related to designing IT-enabled for KIBS.

Designing IT-enabled services requires efforts, models and approaches from multiple disciplines. In this sense, we present in Section 2.4 a survey on service models, methods from and approaches from various disciplines and we show that the systematic thinking and designing of IT-enabled services (namely creating an IT-enabled service system) are quite

beneficial to deal with the main challenges related to IT-enabled services. As we will demonstrate later, a service system is an economic-social-technical system and its design requires multi-disciplinary efforts. Section 2.5 surveys current work to design service systems and concluding outlook are presented in Section 2.6. The outlined organization of this chapter is also illustrated in Figure 2.1, which shows our research scope gradually narrow down our research domain and finally focuses on the IT-enabled service system design.

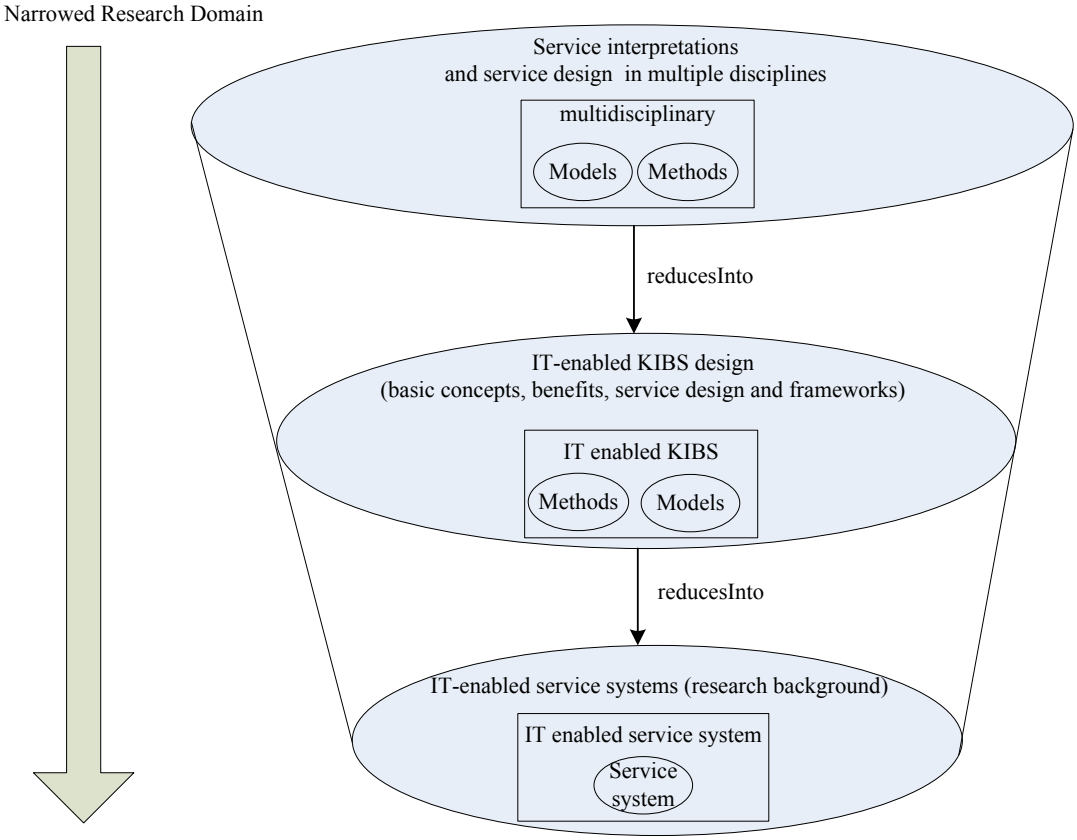


Figure 2.1 The outline framework of this chapter

2.2 Service Interpretations in Different Disciplines

The term 'service' exists for hundreds of years along human history. When a person or a group performs some tasks to benefit another, this act or performance becomes a service. In order to design an appropriate service, in current service economy, service providers have to collaborate with their customers to co-create services for high value (Vargo et al. 2008). The co-creation process may refer to interactions and participations of service actors from different disciplines, such as customers, business experts, IT specialists, and developers. However, these service actors may also use differ-

ent terminologies, and even as for the same term, they may have different understandings and interpretations with respect to their disciplines and domains of interest. In this sense, it is necessary for multi-disciplinary service actors to reach an agreement on respective terminologies used in service-related domains, and particularly, to agree on the meaning of the concept 'service' in the following three disciplines namely Business (including Marketing and Economy), Information System and Computer Science, which are highly involved in the design of services. For this reason, we will discuss in the following of this section different definitions and interpretations for the concept 'service' with respect to these three disciplines.

In the discipline of Marketing and Service Operations, the service concept is widely interpreted as economic activities or transactions. For example, Hill (1977) defines a service as "*a change in the condition of a person, or a good belonging to some economic entity, brought about as the result of the activity of some other economic entity, with the approval of the first person or economic entity*". Kotler and Bloom (1984) emphasize on exchanges between two or more parties and a transaction (potentially intangible) received by a customer. Fitzsimmons and Fitzsimmons (2004) view a service as a time-perishable, intangible experience performed for a customer acting in the role of co-producer. In the literature, the concept of 'service' is also used to differentiate from tangible goods. Sasser et al. (1978) view that a service is intangible and perishable and is created and used simultaneously or nearly simultaneously. Lovelock (2001) argues that goods can be described as physical objects or devices, whereas services are actions or performance. In the business domain, people often consider 'goods' as a synonym of 'product'. Edvardsson et al. (2005) define a service as a process, a performance or an action that reflects and results from the transformation of a company's resources (other than physical products) into value-adding benefits for the company's external and internal customers. However, Goldkuhl and Rostlinger (2000) consider services as products and also view goods as products. They also argue that both of them are results of actions and are aimed for customers. In this thesis, we consider the term 'product' as the synonymy of the term 'goods' and differentiate between the term 'product' and 'service'.

In addition, with the development of Internet and ICT, the concept of e-service is also introduced in the business discipline as new class of services enabled by ICT. de Ruyter et al. (2001) define an e-service as "*an interactive, content-cantered and Internet-based customer service, driven by the customer and integrated with related organizational customer support process and technologies with the goal of strengthening the customer*

service provider relationship." Rust and Kannan (2003) define e-services as "*the provisioning of services over electronic networks*".

In the Computer Science and Information Systems discipline, the interpretations of services used by scientists and professionals may include a large spectrum of services of different granularities such as business services, IT services, Web services, or e-services.

According to the definition of the IT Infrastructure Library (ITIL), a business service has two kinds of meaning: 1) "*an IT service that directly support a business process, as opposed to an infrastructure service which is used internally by the IT service provider and is not usually visible to the business"*, 2) a service that is delivered to business customers by business units. The successful delivery of business services often depends on one or more IT services (ITIL V3 2007). Thus, an IT service is based on the use of Information Technology and supports customer's business processes and involves people, processes and technology.

Based on these definitions, we argue that both of business services and IT services are used to support customers' business processes. Conversely, business services frequently denote results/output to be delivered to customers, whereas IT services are means to enable the delivery of business services.

The concept of Web service is also widely used in Computer Science and especially in the service-computing field. The W3C defines a Web service as a software system designed to support interactions between machines over a network (W3C 2004). Stencil Group presents Web services as "*loosely coupled and reusable software components semantically encapsulated discrete functionality and are distributed and programmatically accessible over standard Internet protocols*" (Stencil Group 2001). Tidwell from IBM defines a Web service as an extension of the Web and as "*a new breed of Web application and self-contained, self-described, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.*" (Tidwell 2000).

On the basis of these definitions, we generalize that a Web service could be a software system, a software component, an application or a function that is encapsulated and invoked through the Web service interface. However, some researchers also highlight the differences between a web service and an application. In fact, an application, also called application software, is defined as a subclass of computer software that helps users to perform a task and it provides functions required by IT services. Each

application might be "*part of more than one IT services and runs on one or more Servers or Clients*" (ITIL V3 2007).

To conclude, applications are generally used to help a user to perform a task, whereas Web services refer to software components or functions used by IT specialists or developers to design applications or IT services for users.

The Free Dictionary (2011) defines an e-service as "*an umbrella term for services on the Internet, which includes e-commerce transaction services for handling online orders, application hosting by application service providers (ASPs) and any processing capability that is obtainable on the Web*". Govindarajan et al. (2001) view that Web services and e-services are synonyms and both of them are either software or applications.

In this thesis, we focus on designing IT-enabled services, also called Web-enabled services, for KIBS firms. IT-enabled services are traditional services in which the integration of IT services in their design, the production, and the delivery processes increase the value for both providers and consumers. Information Technology that enables the business by improving the quality of services is viewed as IT-enabled services (Udaipur 2000). Compared to traditional services or business services, the most important aspect in IT-enabled services is the value addition thanks to ICT. The value addition could be obtained in different ways, such as in the form of customer relationship management systems, improved database, and improved graphical interfaces.

However, there are conceptual ambiguities between IT-enabled services and IT services. According to (Chau et al. 2011), IT services enable the digital transformation of traditional service activities into computational components, automating service processes, creating new service delivery channels, such as Internet and smart phones, and finally facilitating collaboration among various service actors. Du et al. (2005) view that services that address the acquisition, the management and the scaling of IT are IT services, whereas IT-enabled services refer to the business activities that either exist because of certain underling IT support or they specifically employ IT support for productivity improvement. Sudan et al. (2010) present that IT services mainly include IT applications and engineering services, whereas IT-enabled services involve a wide range of services delivered over electronic network, as shown in Table 2.1.

Table 2.1 IT service and IT-enabled service (source: BPAP 2007)

IT service		**IT-enabled services**
Application services	Engineering services	Business process services
Application development and maintenance • Application development • Application development	Manufacturing engineering • Upstream product engineering - Concept design	Horizontal processes • Customer interaction and support (including call centers) • Human resource management

integration and testing • Application maintenance System integration • Analysis • Design • Development • Integration and testing • Package implementation IT infrastructure services • Help desks • Desktop support • Data center services • Mainframe support • Network operations Consulting • IT consulting • Network consulting	- Simulation - Design engineering • Downstream product engineering - Computer-aided design, manufacture, and engineering - Embedded software - Localization • Plan and process engineering Software product development • Product development • System testing • Porting/variant • Localization • Maintenance and support • Gaming	• Finance and administration • Supply chain (procurement logistics management) Vertical processes • Banking • Insurance • Travel • Manufacturing • Telecommunications • Pharmaceuticals • Other Knowledge process outsourcing • Business and financial research • Animation • Data analytics • Legal process and patent research • Other high-end processes

In this section, we have briefly presented the concept of services from different perspectives and shed the light on its large spectrum of applications. As before mentioned, we focus in this thesis on how to design innovative IT-enabled services for KIBS firms. To this end, it is necessary to elaborate the reasons why we need IT-enabled services for KIBS firms. In the next section, we discuss benefits and challenges of IT-enabled services in our service economy.

2.3 A Glance at IT-enabled Services in KIBS Firms

In the service sector, KIBS firms are currently experiencing dynamic growth. Bettencourt et al. (2002) define KIBS as enterprises whose primary value-added activities consist of the accumulation, creation, or dissemination of knowledge for the purpose of developing customized services or product-solutions to satisfy customer needs. Such firms compete by the skills and knowledge of their employees to deliver services. KIBS are services and business operations heavily reliant on professional experts (i.e., scientists, engineers, and IT specialists) that provide their professional knowledge or skills to support for the business processes for other people or organizations (i.e., training services, legal and accountancy services, and IT services).

A common characteristic of KIBS firms is that clients/customers play a critical role by collaborating with their service providers to co-create value, which helps to improve operational efficiency and results into sustainable competitive advantages for service providers and customers. In addition, customers are emergent and highly collaborative since they themselves also possess much of (tacit or explicit) business knowledge and competencies a KIBS firm needs to successfully deliver its service. KIBS are mainly supplied to organizations, not to individuals (Miles et al. 1995).

Another characteristic of KIBS is the co-production of services between providers and customers (Biege et al. 2011).

In addition, KIBS firms are thought to be the most innovative firms in the service sector and they are more frequent to introduce new technologies (particularly ICT) for innovation. Barras (1986) underlines KIBS are innovative, especially through the use of ICT. The emergence of ICT offers KIBS new business opportunities by delivering customized services through the Internet. Miozzo and Grimshaw (2005) states that KIBS play an essential role in the processes of technological development of regional systems since they provide intangible assets that are key drivers of innovation. Also, in KIBS firms, new ideas and best practices obtained from certain projects can be used in other similar projects, which result into the diffusion and propagation of innovative ideas and practices (Smedlund and Toivonen 2007).

Den Hertog (2000) generalizes three key roles of KIBS in exploring innovation: 1) facilitator: supports a client in its innovative process but not directly generate innovation; 2) carrier: experiences in certain project can be used to other similar projects to produce innovation; 3) source: KIBS firms directly generate and elaborate innovative solutions for the clients. Martinez-Fernandez et al. (2004) view that "*KIBS provide a platform to study a group of services which are very actively integrated into innovation by joint knowledge development with their clients*".

Some researchers argue that the major drivers for KIBS growth are outsourcing, technology (especially ICT), growing demands for specialized knowledge for social, administrative and regulatory issues, factors associated with internationalization and globalization of business, an increasing emphasis on service and on intangible elements of service production, as well as issues regarding labour markets for knowledge workers (Sector Future 2005).

In terms of these initiatives and arguments, we identify the following advantages regarding IT-enabled services for KIBS firms:

1- IT-enabled services exploit ICT to improve the efficiency of an organization, which provides a wide range of services that include opportunities in contract centres, call centres, software development, medical transcription, back office operation, legal databases, web services, etc.

2- IT-enabled services can achieve cost reduction and functional specialization due to the dramatic growth of outsourcing services.

3- IT can create or deliver service functions more efficiently and effectively.

4- IT-enabled services are easy to help service actors to find an appropriate way to improve and innovate services than other kinds of services in the service actor.

5- Help to integrate various business services from different service actors and enable their collaboration to co-produce innovative and customized services with high efficiency.

6- Enable the survival for Small and Medium Enterprises (SMEs) in the global market since they provide quite professional services to be integrated into complex business value chains.

However, IT-enabled KIBS also raise challenges. In Chapter 1, we have listed four research questions related to creation of IT-enabled service systems for KIBS firms. In response to those challenges, it is necessary to survey current approaches on service design and service system engineering and evaluate if these approaches can help to deal with these challenges.

2.4 Service Models and Methods in Multiple Disciplines

2.4.1 Basic Concepts

In the previous section, we have articulated various interpretations of the term 'service' in different disciplines that demonstrate divergence and ambiguities when customers, business experts, IT specialists, developers and other service actors collaborate with one another. We have also discussed the challenges related to IT-enabled services for KIBS and identified the needs for multidisciplinary models and methods to design innovative and customized services that take into account customer satisfaction. To this end, it is necessary to understand how current models and methods have been developed to design services and how they deal with the aforementioned challenges. Before presenting this survey, we at first briefly discuss the definitions of the following concepts: 'method', 'model', 'service method', and 'service model' to clarify the context of our discussion.

Late Middle English (2011) defines a method as 1) "*a procedure, technique, or way of doing something, especially in accordance with a definite plan*; 2) *a manner or mode of procedure, especially an orderly, logical, or systematic way of instruction, inquiry, investigation, experiment, presentation*, etc. 3) *order to system in doing anything*, 4) *orderly or systematic arrangement, sequence, or the like*".

A model is "*a systematic description of an object or phenomenon that shares important characteristics with the object or phenomenon. Scientific models can be material, visual, mathematical, or computational and are often used in the construction of scientific theories*" (Science Diction-

ary 2005). Following the logic of this definition, we define a service model as a systematic description and abstraction of a service that shares important characteristics.

In the perspective of software system, service modelling is defined as "*the process of applying design principals and a context to the implementation of business capabilities" and it generally includes four phases: respectively service identification, service classification, service description, and service implementation*" (Manes 2011). Manes (2011) defines service models as "the *artifacts resulting from rich descriptions of a service that include metadata, such as interface descriptions, policies, dependencies, and consumer-provider agreements*".

In the literature (i.e., Chase 1978, Baida 2006, Donzelli 2004, Gallouj and Weinstein 1997, de Vries 2006, Mo et al. 2009, Tung et al. 2007, Qiu 2009, Fujii and Suda 2009, Issa et al. 2006, Sampson 2007, Tien and Berg 2003, Nof 2007, etc), various kinds of service models or approaches are presented, such as service contact model, service blueprint, requirement model, service description model, service reference model, service ontology model, client relationship model, collaboration model, service composition model, service delivery model, innovation model, service quality model, and customer satisfaction model.

In this thesis, we are not expected to discuss all these models but we restrict our scope to exploring systematic service paradigms to design innovative services, which deal with services as social-technical-economical systems and call for multi-disciplinary methods to produce innovative and value-added services. In this sense, it is necessary to present different ways to view and/or design services and to show their benefits.

2.4.2 Service Paradigms

A paradigm is "*a philosophical and theoretical framework of a scientific school or discipline within which theories, laws, and generalizations and the experiments performed in support of them are formulated*" (Merriam-Webster 2006). Sampson (2007) argues that "*a good services paradigm will need to be foundational, encapsulate common assumptions, and help specify advancement activities. At a minimum, a services paradigm should help those studying services decide what a service is and how services are distinct from non-services*".

In this study, we roughly classify current service paradigms into three categories: including service marketing, service operation, and service system engineering. It is worth noting that these three views do not

exclude one another and they may overlap and some service paradigms may be their integration.

Rust (2004) asserts that the service marketing perspective focuses on understanding and serving customers whereas the service operation perspective emphasizes the design of service delivery systems. Sampson (2007) assumes that the service marketing paradigm aims at what is provided to customers and how the customer receives benefits from the service, whereas the service operation paradigm highlights how services are produced and how value is delivered to customers. The systematic view seeks to introduce principals from multi-disciplines such as Computer Science, Management, Economy, Engineering, and Social Science to explore service economical-social-technical features and underline systematic approaches to explore value co-creation of various stakeholders (i.e., people, organization, agent, and system).

One of the most prominent service marketing paradigms may be the value model that views a service as a value-added deed, performance or effect to satisfy customers (Rathmell 1966, Gummesson 1994). Other scholars extend the value model by introducing new service elements, such as resource, competency, and technology. A service is the application of specialized competences (knowledge and skills), through deeds, processes, and performances for the benefit of another entity or the entity itself (Vargo and Lusch 2004). A service is also viewed as process, performance and actions that reflect and result from the transformation of a company's resources (other than physical products) into value-added benefits for the company's external and internal customers (Edvardsson et al. 2005).

One of the pre-eminent service operation paradigms is the customer contact model (Chase, 1978) that advocates physical presence of a customer in a service. Other researchers extend the model and consider a service process as a set of interactions between service providers and customers. For example, Gronroos (1990) defines a service as "*an activity or series of activities of more or less intangible nature that normally, but not necessarily, take place in interactions between customer and service employees and/or physical resource or goods and/or systems of the service provider which are provided as solutions to customer problems*". As a landmark article, Roth and Menor (2003) call on researchers to investigate service design issues through operations management lens. Zhang et al. (2007) views that a service represents a type of relationship-based interactions (activities) between at least one service provider and one service customer to achieve a certain business goal. Balin and Giard (2006) argue that it is impossible to present a service definition fit all services and they thus propose to interpret the service paradigm in an integrated view.

Eiglier and Langeard (1975) perhaps are the first researchers who define services from a systematic perspective. They argue that an elementary service is the result or the output of a service system. Sampson and Froehle (2006) views that the presence of customer inputs (i.e., customer-self, his belongings or tangible objects and information) is a necessary and sufficient condition to define a service process. Pinhanez (2008) extends Sampson and Froehle's model and asserts that a production process is considered as a service process when customers simply hold a small part of production resources and at the same time, their customer inputs are indispensable to the production process. Maglio and Spohrer (2008) view that a service system is a configuration of people, technologies, and other resources (including shared information as languages, laws, measures, methods) that interact with other service systems to create mutual value.

In terms of these arguments and initiatives, we argue that the service marketing view can provide a clear view about how to create value by designing an innovative service for customers and how to achieve customer satisfaction, but it rarely refers to how to design a service delivery system that may also lead to additional value and innovation. The operation view clearly discusses how various service actors collaborate with each other to co-design a service system by mobilizing respective resources; especially how customers take part in service production processes and facilitate the service design. The systematic view integrates these two perspectives and underlines collaboration, value co-creation, and customer inputs during the whole service lifecycle, as well as flexible strategies for the service delivery thereby being potential to create more value. In this sense, it is necessary to present how the systematic view creates and elaborate its benefits with respect to our research context.

2.4.3 Systemizing Services

A system is broadly defined as a set of elements or parts that are organized as a whole for a common purpose or goal. In addition, these various elements or parts could have interacting, interrelated or interdependent relationship when they work as a whole. A system is generally composed of input, output, process and feedback. System thinking is the process of understanding how things influence one another within a whole, which is good for complex problems to see their big pictures (Jackson 2003).

Kim and Nam (2009) figure that the systematic approach provides an effective way to help various service actors to identify the big picture of a service and its components and finally helps them to find an appropriate way to improve and innovate services through collaboration.

Mai and Bosch (2009) generalizes the main objectives of systematic approach: 1) to focus on the whole system and the constituent parts as well as their interactions, 2) to provide a framework for managing changes and complexity through the understanding of dynamic feedback embedded in complex systems, 3) to allow decision makers to anticipate the long-term consequences of their decisions and actions, and the unintended consequences of polices and strategies, and 4) to provide a common language for diverse stakeholders for deep dialogues and reaching consensus.

Additionally, compared to the traditional service operation paradigm that considers a service as the transformation of resources, the systematic view considers them as benefits/outputs of collaboration between various service actors (i.e., human, software, hardware, and organization) with different roles (i.e., customers, providers, etc.) by applying resources, technologies, competencies and delivery channels to co-create value and satisfy customer requirements. In other words, a service is the output of a service system in which various service actors collaborate with each other to co-create value and to achieve or exceed customer expectations. Service actor input is necessary throughout the whole service lifecycle. In addition, in the context of IT-enabled service systems, the systematic view guides enterprises to manage the connections/relationships between service systems rather than taking into account their internal components, and model all relationships between various service systems as interactions via some (informal and manual) set of interfaces (Gartner Group 2009).

We summarize that the service system paradigm offers the following advantages for KIBS firms to expose, bundle, and provide their services as IT-enabled services:

1- Emphasize on collaboration and value co-creation of various service actors.

2- Underline the necessity of customer input in each step of the service production process and assume that customers should take part in all decisions related to service production and delivery processes.

3- Help to understand the big picture of a service (front-office) and its complex internal components (back-office).

4- Separate the service utilities (exposed characteristics) out of internal service system components.

5- Help service actors to easily find an appropriate way to improving and innovating services through collaboration.

6- Take into account the social-organizational-technical nature of services.

As mentioned before, services are characterized by intangibility, heterogeneity, inseparability, perishability, and customization. As a result,

models or methods developed to tangible goods domain cannot be directly used in services due to their characteristics. In addition, systematic approaches for designing traditional service systems (i.e., information system, software system, social-technical system, etc.) cannot be directly used to design IT-enabled service systems. In this sense, on the basis of our presented big picture in Section 2.1, it is necessary to survey current systematic service design methods and/or models to illustrate their potentials and drawbacks for designing IT-enabled services.

2.4.4 Service Design Framework/Methodology

Service design refers to "*the design of the overall experience of a service as well as the design of the process and strategy to provide that service*" (Moritz 2005), which refers to "*activities of planning and organizing people, infrastructure, communication and material components of a service in order to improve its quality and the interaction between service provider and customers. The purpose of Service design is to design a service to satisfy customer requirements*" (Mager 2008).

The main development of service design is that in the earliest days, Shostack (1984) view the activities of designing services as part of the domain of marketing and management disciplines. In 1991, service design was first introduced as a design discipline by Michael Erlhoff at the Köln International School of Design (KISD). So far, various models are presented to design services. Morelli (2006) presents three main directions based on the overview of current service/product design methods: 1) Identifying service actors using appropriate analytical tools. 2) Elaborating possible service scenarios and use cases. 3) Representing services (including static components, and dynamic interactions). Morelli has also referred to some service design models and tools, such as IDEF0 (Integration Definition for Function Modelling) (Congram and Epelman 1995) and service blueprint (Shostack 1984), specific to these three aspects.

In addition to specific service design models and tools, more importantly, we need a service design framework to strategically provide a big picture about how to design services using service models. To this end, it is helpful to discuss service design frameworks in the literature and to analyze if they can be applied to IT-enabled services. A service design framework or service design process can be defined as innovative and creative process that helps companies to clearly define and design their services for customer satisfaction by applying service design models and tools. The International Organization for Standardization (ISO) presents general guidelines for service design processes in the light of three stages: respectively ser-

vices specification, service delivery specification and service quality control specification (ISO 1991). The service specification defines the service to be provided, whereas the service delivery specification defines the means and methods used to deliver the service. The quality control specification defines the procedures for evaluating the controlling the service delivery characteristics.

The corresponding service design process mainly includes two steps: 1) converting customer needs into specification for both the service and its delivery and quality control. 2) Design of the service specification and use flow chart to depict all activities, relationships and interdependences. Edvardsson and Olsson (1996) argue that the ISO principals simply consider a customer as a service recipient rather than a value co-creator. In addition, the ISO approach mainly conforms to logics of industrial product production rather than that of the service production. The sequential activity-based business process fails to describe the logic of IT-enabled service systems for KIBS since in IT-enabled service systems, various service actors mainly exchange explicit business knowledge in collaboration for value co-creation, which leads to their collaboration fairly ad hoc and dynamic.

Alonso et al. (1994) argue that companies should not only design innovative products and services, but also design efficient and effective service support systems to support the integration of products and services. This view is in accordance with the idea of systematic approaches to design innovative IT-enabled services.

Hax and Majluf (1991) view that before designing a new service or a product, a company should begin with a strategic assessment and they thus propose a strategic assessment framework to help managers to make decisions. The framework is mainly composed of four phases: 1) definition of mission including degree of concept and degree of market innovation in the light of the firm's long-term strategy, 2) external analysis of trend, threats, and opportunities, 3) internal analysis, such as how the new service or product fits in the firm's current offering, and how it will impact its operations, and 4) strategic analysis comprising marketing requirement, strategic requirements, and regulation requirements. We argue that this approach can not only be used to assess the strategy to design a service for a company before designing a new service, but also helps to evaluate if the designed new service reaches to the company's expectation.

Based on the strategy framework proposed by Hax and Majluf (1991), Bitran and Pedrosa (1998) propose a top-down design method and view that the service design process is similar to that of the hardware design and they present a design framework consisting of five stages in the

light of product development principals: 1) strategic assessment, 2) service concept design (e.g., customer need identification, and requirement specification), 3) system design, 4) component design, and 5) testing and implementation. The strategic assessment stage has been described beforehand in (Hax and Majluf 1991). Bitran and Pedrosa (1998) also present a framework to design service concept sequentially composed of marketing requirement, customer requirement identification, design requirement identification and ranking, concept generation, concept selection, and design attributes specification. Concept generation starts by making a rough sketch of the products, services, functions and attributes that the service provider believes are necessary and each design requirement is taken into consideration and translated into corresponding attributes. Concept selection refers to the process of evaluating generated concepts and selecting the most appropriate concepts for further development. We argue that the service concept design process provide a top-down approach to concretize abstract service concepts into precise and concrete service concepts and also take into account attributes of QoS.

This framework and its top-down approach are quite strategically useful to design different kinds of services ensuring their quality by taking into account QoS. However, since this framework overemphasizes the role of designers and neglecting the collaboration and value co-creation of service by all actors, it is not enough to IT-enabled services and requires adjustment to this new context. It also lacks of effective approach to deal with the gap between service perception and service expectation. In terms of this issue, Clark et al. (2000) propose to adopt the same representation model to vision services among various stakeholders, such as customer and provider.

In analogy to the development of a discount brokerage service, Shostack (1984) presents a service blueprint model for a new service development including ten steps: 1) value description of the basic service function, 2) information gathering process, 3) clarification of the service definition and boundaries drawn, 4) detailed service description, 5) internal review of the service and external market research, 6) documentation of service description & implementation plans, 7) operation functions put in place and tested, 8) implementation of communication strategy, 9) introduce the service to the market, and 10) review of service for further development.

We argue that this framework can be used as a common top-down method to design services. Actually, the service design process works in a cyclical and spiral way until service designers at last confirm the final service. Service blueprint displays services as "*interdependent, interactive systems*" (Shostack 1987). The service blueprint is used as a framework to

integrate processes, applications, and infrastructure to deliver integrated services to various customers, employees, suppliers, and portals (Kalakota and Robinson 2003). Southern (1999) shows that service blueprint facilitates the understanding of the way to process function within the overall service system. Zeithaml et al. (2006) consider service blueprint as a tool for simultaneously depicting the service process, the points of customer contact, and the evidence of the service from the customer's point of view. Ponsignon et al. (2007) discuss that service blueprint is particularly appropriate to analyze operations where the customer is directly involved in the service delivery process.

Similarly, Edvardsson and Olsson (1996) present a three-phase-based framework to design a service including: 1) service concept design, 2) service process design, and 3) service delivery system design. They also introduce a holonic view to design service delivery systems stating that a service system must be designed as a whole but requiring various sub-service-systems function separately to be integrated. This framework covers the service design and the service delivery systems design, which is in line with systematic service design principals. We argue that the service concept design can be described as the phase of identifying and specifying customer requirements. The service process design can be viewed as the design of ad hoc collaboration process in service systems. The holonic view of service delivery system also conforms to the principals of integrating various self-organized service actors to co-create value.

Kim and Nam (2009) finds out that clients care more about the service quality than an efficient information system that provides fast and convenient access based on past experiences. Kim thus presents a new IT-convergence service design framework including the following phases: 1) service concepts, 2) service design, 3) service simulation, 4) service platform, and 5) real service. This service design framework can be used for service providers to co-design, co-simulate and co-operate a service with clients, which confirms to the core idea of value co-creation of service systems to improve service quality.

We investigate that these traditional product or service design processes dominantly adopt a top-down method to design a service, which conforms to the way people do businesses (from abstract to concrete and from customer requirements to a technical requirements). In addition, in these frameworks, service designers play a dominant role in designing a service to satisfy their clients but clients rarely or partially take part in the design process (e.g., customer contract and customer involvement). We argue that the survey of service design frameworks provides a big picture about how to design a service from scratch regardless of the adoption of

systematic view or non-systematic view (i.e., marketing perspective or operation perspective). Although these frameworks are not specific for IT-enabled services, they provide guidelines to design IT-enabled services. For example, before designing an IT-enabled service, the strategy framework from Hax and Majluf (1991) can be used to guide the design process.

In addition, as mentioned before, different from traditional product or business services, IT-enabled services refers to large quantities of collaboration among service actors to exchange business knowledge enabled by ICT for value addition. Moreover, IT specialists play an active role to integrate input from service actors at business and technical levels to co-create value. In this sense, these traditional product or service design processes have to be adjusted to support the collaborative context in designing the service systems. To this end, a middle-out approach seems to be a convenient approach to simulate a context of peer-to-peer collaboration among all service actors instead of a top-down design approach.

Further, as mentioned before, the designing of a service from the systematic view provides more values than that simply from the service marketing perspective or operation perspective. Therefore, in the next section, we articulate current systematic approaches for service design.

2.5 Exploring Service System Engineering Approach

The Electronic Industries Association (EIA) defines System Engineering as: "*an interdisciplinary approach encompassing the entire technical effort to evolve and verify an integrated and life-cycle balanced set of systems, people, product, and process solutions that satisfy customer needs*." (EIA/IS 632 1994). Service system engineering is "*a multidiscipline that addresses a service system from a life-cycle, cybernetic and customer perspective*" (Tien and Berg 2003).

Tien and Berg (2003) also view that system engineering is a multidiscipline, including electrical engineering, mathematics, statistics, operations research, management science, computer science, decision science, human-machine systems, industrial engineering and bioengineering. By applying system engineering principals to services, Tien and Berg (2003) identify some unique characteristics of services, such as Information-Driven, Customer-Centric, E-Oriented, Productivity-Focused and service system engineering methods related to these characteristics, as shown in Table 2.2.

These four unique characteristics enabled by ICT and Decision technologies imply value addition compared to traditional service design approaches. Tien and Berg (2003) in their work simply provide the potential applications of ICT and Decision technologies to service system engi-

neering, but they do not elaborate how to create a service system with ICT and Decision technologies to achieve these four unique characteristics.

To this end, we will explore current service system design approaches from four different perspectives and discuss how they are used to create service systems.

Table 2.2 Service system engineering: unique Characteristics and Evolving Methods (source: Tien and Berg 2003)

Unique Characteristics	Evolving Service Systems Engineering Methods	
	Information/ Communications Technologies	Decision Technologies
Information-Driven • Creation • Management • Sharing	• Collaborative Software, Business Intelligence Software • Synchronization Software, Autonomic Computing • Peer-to-Peer Networking, Distributed Computing, Extensible Markup Language	• Data Mining, Decision Informatics • Data Mining, Decision Informatics, Index/Pointer Scheme • Index/Pointer Scheme
Customer-Centric • Co-Production • Customization • Satisfaction	• Intranet, Extranet, Internet • Software Agents, Synchronization Software, Peer-to-Peer Networking • Software Agents	• Demand Management, Decision Informatics, Adaptive Techniques, Artificial Intelligence • Data Mining, Decision Informatics, Adaptive Techniques, Artificial Intelligence • Demand Management
E-Oriented • E-Access • E-Commerce • E-Customer • Management	• Wireless, Internet-on-a-Chip • E-Procurement, E-Fulfillment, E-Supply Chain, E-Outsourcing, E-Auction • Customer Management Software	• Decision Informatics • Supply Management, Economic Value Added Analysis • Demand Management
Productivity-Focused • Efficiency • Effectiveness	• Customer Management Software • Customer Management Software	• Data Envelopment Analysis, • Reengineering • Demand Management, Life-Cycle • Analysis

2.5.1 Service System Definitions and Characteristics

Compared to traditional service marketing or service operations, the systematic view in Service Science perspective takes into account economic-socio-technical nature of services. Service systems comprise service providers and clients working together to co-produce value in a complex value chain or network (Tien and Berg 2003). A service system comprises service providers and clients working with each other in complex networks or chains to co-create value for all stakeholders, with value creation being realized by the transformation of something owned or controlled by the client (Spohrer et al. 2007, Krishna et al. 2007). A service system is "*any number of elements, interconnections, attributes, and stakeholders interacting in a co-productive relationship that creates value, where services are intangible*

activities customized to the individual request of known clients" (Grandison and Thomas 2008).

Service system is as a dynamic value co-creation configuration of resources, including people, organization, shared information and technology, all connected internally and externally to other service systems by value propositions (Spohrer et al. 2007). Service system is composed of people, technologies and other resources that interact with other service systems to create mutual value (Spoher et al. 2008).

In service systems, knowledge workers depend on their knowledge, tools, and social-organizational networks to solve problems, be productive, and continually develop, generate and capture value (i.e., IT outsourcing, call centres, database system, etc) (Magio et al. 2006).

Service system is a useful abstraction for understanding value and value co-creation and in which value can be defined simply in terms of an improvement in system well-being (Vargo et al. 2008). They also argue that value can be measured in terms of a system's adaptiveness or ability to fit in its environment.

IT-enabled service systems are quite complex and refer to large quantities of service actors from the global world interconnected in a system-of-systems value chain. IT specialists are quite significant to enable the integration of them to produce an innovative and customized service. Also, in IT-enabled service systems, customers do not wait for the delivery of services from their providers but they take part in all aspects and levels of decision makings to dynamically co-determine the appropriate strategy to co-design and deliver the service with their providers.

In order to create an IT-enabled service system, it is quite important to identify its internal components and their relationships and to present how various service actors collaborate with one another to construct a value network. The service system components include people, organization/enterprise, software, hardware, technology and shared information (Spohrer et al. 2007, Krishna et al. 2007). Krishna et al. (2007) view that complex service system compositions include people (e.g., individuals, teams, organizations), technology (e.g., information processing, communication, self-service), and business management processes all interconnected by value and information networks, resulting in observable dimensions or characteristics.

Besides well-identified service system components, the unique characteristics for service systems are also necessary to be discussed to differentiate service systems from other systems and to facilitate the understanding of them. As mentioned before, Tien and Berg (2003) characterize service systems by information-driven, customer-centric, e-oriented, and

productivity-focused. Qiu et al. (2007) list some essential features of a service system including people-centric, information-driven, e-oriented, and satisfaction-focused. They also view that a service system should encourage and cultivate people to collaborate and innovate. In terms of these initiatives, we identify the following characteristics for IT-enabled service systems:

Technology-driven: since in KIBS firms, service actors apply their professional knowledge and skills to facilitate collaboration and produce value-added services. Technologies, especially ICT technologies, are widely adopted in KIBS firms to improve the degree of flexibility and automation.

Customer inputs throughout the whole service lifecycle: in service systems, although service providers may play a dominant role, service customers also contribute to the service production. The context of customer input is different from that of customer contract or customer involvement. In the context of customer input, customers input their body, knowledge, belonging, resources, and assets to collaborate with service providers across the whole service lifecycle and customers are viewed as value co-creator (Sampson and Froehle 2006), whereas customer contact or involvement states that customers simply partially take part in the whole collaboration process.

Value co-creation: service provider collaborates with service customer to co-create value added services by using respective knowledge and technology (Vargo et al. 2008). For instance, in e-learning services, if a student studies hard, takes more time and efforts to his study and finishes all the assignments on time, he will command the course faster and better than other students who do not study hard. We can say that he creates more value than those who do not command well the course in the same period of study time and with the same cost.

High innovation: service systems are designed to improve productivity, quality, compliance, and innovation (Spohrer et al. 2007). In service systems, the whole service process is knowledge-intensive and various service actors exchange both tacit and explicit knowledge during their collaboration. Compared to non-knowledge-intensive activities, knowledge-intensive activities are much easier to produce innovation.

Human-oriented: human resources with professional knowledge play a dominant role in KIBS firms. They apply their professional knowledge to work with customers to co-create value. For example, a consultant uses his or her professional knowledge to solve a client's problems. Mo et al. (2009) view a complex service system as not a simple information system, but it includes non-software elements (i.e., people and hardware).

Service delivery via Web: in today's economy, service providers and their clients may be distributed in different locations. Technologies (ICT, Service Computing, Cloud Computing, Web 2.0, Web Service, etc.) make their collaboration possible even when they are scattered in the world. Now, service providers can provider services via Web and customers can also consume services via Web. Service delivery via Web not only reduces cost, but also improves efficiency for service actors.

Dynamic and ad-hoc collaboration: collaboration in the context of KIBS firms consists of interactions between service providers and customers, which define collaborative processes of assessment, planning, facilitation and advocacy for services to satisfy customer needs through communication and available resources. This collaboration process is an ad hoc and human-driven and does not follow a predefined workflow.

System-of-systems structure: the system-of-systems nature can help to interconnect global service enterprises for value co-creation.

IT specialists-driven: in IT-enabled services, IT specialists play an active role in integrating various service actors to work together for value co-creation.

Evolution: service systems will change over time as constituent systems are replaced or updated.

The main value for these characteristics is to differentiate between IT-enabled service systems and other systems (i.e., software systems and product systems). If a system has these characteristics, it can be viewed as an IT-enabled service system, urges new engineering requirements compared to traditional software system, information system, or product system and requires new design methods and models. In the next part we discuss engineering requirements for IT-enabled service systems.

2.5.2 Service System Engineering Requirements

Service systems are designed to deliver or provide services. Due to the diversity and complexity of service systems, engineering requirements. Stanicek and Winkler (2010) identify the following service system engineering requirements: including autonomous, co-operative, composable, and adaptive. Glushko (2010) argues that service systems "*combine and integrate the value created in different design contexts like person-to-person encounters, technology enabled self-service, computational services, multi-channel, multi-device, and location-based and context-aware services*".

Qiu (2009) generalizes three service system engineering requirements: 1) to timely capture end users' requirements, changes, expectation and satisfaction in a variety of technical, social, and cultural aspects; 2) to

efficiently and cost-effectively provide employees right means and assistances to engineer services while promptly responding the changes; and 3) to allow involved people consciously infuse as much intelligence as possible into all levels and aspects of decision-making to assure necessary system adaptiveness from time to time.

As mentioned before, Mai and Bosch (2009) also present some service system engineering requirements:1) to focus on the whole system and the constituent parts as well as their interactions, 2) to provide a framework for managing changes and complexity through the understanding of dynamic feedback embedded in complex systems, 3) to allow decision makers to anticipate the long-term consequences of their decisions and actions, and the unintended consequences of polices and strategies, and 4) to provide a common language for diverse stakeholders for deep dialogues and reaching consensus.

A service system is defined in terms of multiple and independent entities such as individuals, organizations, services, software and applications sharing one or several missions and focusing on the interactions and inter-relationships among them. The service system should be designed and delivered to achieve maximum customer satisfaction at an appropriate cost in a certain context. As a result, the service system is considered as a multidisciplinary engineering approach that addresses the service from a customer perspective.

The service system should be able to make an abstract of people (i.e. stakeholders), technologies (i.e. network, software, etc.) and exchanged explicit business knowledge (i.e. shared information) as elementary or atomic services. The process of combining these atomic services provides added-values that come from the skill, knowledge, time, and energy that the service providers invest in the service delivery to satisfy customer needs.

In many situations, several service providers collaborate to build a customized service. Service interoperability is a property referring to the ability of diverse services to work together. Interoperability does not only mean the ability of two or more services to exchange information and use the information that has been exchanged, but also the ability to semantically interpret information exchanged and unambiguously understand its meaning by service providers and consumers. To achieve semantic interoperability, both sides must refer to a common ontology reference model. In the context of knowledge-intensive firms, a business service is offered by one or more service providers. Service systems that are running on one or more service actors (human or machines) need to interact over the Internet to deliver service to customers.

Business services provided by KIBS firms over the Internet constitute a critical and growing economic sector. Service systems engineering is a powerful and productive multidisciplinary combination of individuals, organizations, technologies and businesses to deliver services in order to satisfy customer needs.

Based on these arguments, we identify the following engineering requirements for IT-enabled service systems:

Customization: service providers provide customized services for clients/customers. Customization plays an importance role in reaching customer satisfaction and producing innovation (Tien and Berg 2003).

Loosely-coupled and holonic structure: a service system must be designed as a whole but requiring various sub-service-systems function separately to be integrated (Stanicek and Winkler 2010). In addition, in a service system, various service enterprises interconnect and present a system-of-systems structure.

Propagate requirement changes: if sudden or constant changes happen in customer requirements, a service system should support these changes and make corresponding updating/changes to their services by re-mobilizing its internal components adapting to these changes.

Uncertainty: in service systems, a service provider has a variety of options to achieve customer satisfaction and the decisions made will gradually determine the appropriate strategy to deliver the service (Peng et al. 2009). In addition, the dynamics of both service customer and the environment make business behaviours and strategy uncertain.

Non-determinism and non-tractability: a service system is fundamentally non-deterministic. It is impossible to anticipate precisely the behaviour of such system even if we completely know its components and their functions (Tung et al. 2007).

Emergence: is the process of deriving some new and coherent properties in a complex system. Properties, which do not belong to any of the constituent parts/systems, will emerge from the combined system-of – systems (Tung et al. 2007).

Scalability: To be scalable, the system should be able to dynamically incorporate arbitrary numbers of additional components or service actors in its system-of-systems value chain.

Interoperability: denotes the ability of various components or systems to work together for a common goal. As mentioned before, interoperability does not only mean the ability of two or more services to exchange information and use the information that has been exchanged, but also the ability to semantically interpret information exchanged and unambiguously understand its meaning by service providers and consumers.

Self-organization: in service systems, although various constituent systems collaborate with each other to produce a service, each of them behaves autonomously and can independently achieve a task without a central authority or external elements imposing on it (Mo et al. 2007).

Adaptability or flexibility: this property denotes the capability to adapting to internal or external changes in service systems (Mai and Bosch 2009).

Semantics: due to the collaboration-driven characteristic of service systems, they need to collaborate with each other. But different service systems are implemented with different techniques or models. As a result, they confront vocabulary and conceptual ambiguity when they collaborate with each other. Semantics can help to deal with this problem.

Reusability: denotes the capability to reuse existing services or legacy systems in building larger or new systems.

The main value of these service system engineering requirements lies on that: 1) at design time, they can be used as the objectives by various service actors to co-design a service system, and 2) in the run time, they can also be adopted as the criteria to measure the performance of the service system. In terms of these requirements, in the following of this chapter, we will explore current approaches and evaluate if they can help to achieve them.

2.5.3 Service Systems Engineering

The service system engineering targets at making various service actors (customers, business experts, IT specialists and developers) at design time co-design an IT-enabled service system and in the run time supporting various service participants' (i.e., providers, customers, vendors, and supplier) decision making in changing businesses for value addition. This goal is quite similar to that of current Enterprise Architecture (EA) frameworks that focus on how to integrate business models (i.e., process model, and requirement model) and technical models (i.e., system architecture and data model) to support the changing businesses in enterprises.

Gartner Group (2010b) identifies four basic approaches to EA: traditional, federated, middle-out, and managed diversity and predicts that most organizations will support multiple approaches. The traditional approaches are top-down approaches that are supported by most current EA frameworks (i.e., ISA, TOGAF, FEAF, etc). As mentioned before, the top-down approaches are appropriate to relatively stable businesses. The federated approaches are appropriate to large, complex organizations, where decision making is often largely decentralized and business units have con-

siderable autonomy. Managed-diversity architecture is focused on defining choices or options. Managed diversity is "*focused on balancing the need for a set of standards with the need for a diversity of solutions to increase innovation, business growth and competitive advantage*" (Gartner Group 2010b). The advantage of this approach is that "*it enables users and project teams to select the right tool for the job, enabling innovation through diversity. The disadvantage of this approach is that users and project teams must accept more responsibility for their choices*" (Gartner Group 2010b). Middle-out architecture is "*an approach to EA whereby architects focus on managing the key dependencies among those parts of the organisation that have the biggest impact on the ability to change*" (Gartner Group 2010b). The middle-out approaches are quite effective to collaborative business ecosystem where business units, partners, and suppliers are not under the direct control of a central EA team but "*architect the lines, not the boxes*" (Gartner Group 2009). Gartner Group (2010a, 2010b) also suggests a blended approach of them for EA to meet the needs of dynamic businesses.

An IT-enabled service system is collaborative, dynamic system-of-systems value chain, in which various autonomous service actors collaborate with one another for value co-creation. Therefore, based on the aforementioned arguments, the middle-out approaches allow the design of innovative and adaptive IT-enabled service systems.

After identifying the appropriate approaches or strategies for EA, we have to identify the detailed business models and technical models for designing an IT-enabled service system. Since the discipline SSME is quite new (proposed in 2004), the well-developed system engineering methods specific for service systems so far are quite limited. In the following, we roughly classify current engineering approaches for service systems into four categories: respectively socio-technical systems (e.g., work system), ecosystems, holonic systems, and information systems and we will respectively explore their characteristics and identify their potentials/value to create IT-enabled service systems.

2.5.3.1 Service Systems as Socio-technical systems

Yass et al. (2010) view that socio-technological systems are combination of natural and manufactured systems which refer to any number of elements, interconnections, attributes, and stakeholders interacting to satisfy the request of a known client and co-create value. So far, various well-formed specifications have been presented to deal with software-human behaviours, such as WS-HumanTask (2007) and BPEL4People (2007).

Bostrom and Heinen (1977) present a socio-technological system model, which hypothesizes the presence of two subsystems in every organi-

zation or corporation: respectively technical sub-system and the social sub-system. Cartelli (2007) argues that Bostrom and Heinen's model (1977) just takes into account the individual and society element in the social sub-system, which can not completely represent the social structure. In this sense, Cartelli in this work presents an extended socio-technological system model by introducing the concept of community to the social sub-system that is viewed as the middle level between individuals and the society.

Qiu (2009) views a service system as a social-technical system, focusing on engineering and delivering services using all available means to realize respective values for various service actors. In this work, Qiu presents an approach to modelling the dynamics and adaptiveness of service systems taking into account human's physiological and psychological issues, cognitive capability, and sociological constraints during the whole service lifecycle. He presents a computational model (C^2S^2) revised and adapted from Yet Another Workflow Language (YAWL) (Van Der Aalst and Ter Hofstede 2005) to look into systemic operations, behaviours, and interactions in a service system by taking into account its operational dynamics and adaptiveness. The C^2S^2 model advocates that a service system is conceivably constructed in terms of the extended Business Process Management (BPM) model as a process-based service system and introduces a set of symbols from YAWL into BPM for incorporating human task and human interactions.

In addition, in order to measure the viability of dynamic service systems and take into consideration people's psychological behaviours and cognitive capability, and sociological constraints, Qiu in this work also proposes to analyze dynamics in a service system, including exploring, capturing, understanding, and managing systemic behaviours and interactions, connections, complex relations, and interdependencies, from a social-technical view by using Structure Equation Modelling (SEM)) (Kaplan 2000) technique. YAWL is developed to directly support all the control flow patterns required in a workflow system (van Der Aalst and Ter Hofstede 2005). SEM is a statistical technique widely used to study social and/or economic behaviours of organizations (Kaplan 2000).

This model presents a conceptual model for social-technical service systems taking into account dynamics. It also infuses human's influence to all levels and all aspects of decision-making to ensure necessary system adaptiveness to manage runtime operations. In addition, it presents a model to measure the viability of a service system taking into account profits and customer satisfaction for guaranteeing smart decision-makings. However, this model is still process-centric and seeks to compose a set of hierarchical tasks into complex tasks, which lack of flexibility to enable

business people managing their day-to-day businesses. It does not provide an effective mechanism/approach to enable the collaboration of various service actors who use different models or languages.

Another initiative in this domain considers service systems as work systems (Alter 2002, 2008). A work system is defined as "*a system in which human participants or machines perform tasks or work by using their information, technology, and other resources to produce services for internal and/or external customers*" (Alter 2002). Alter (2008) proposes three frameworks that provide a foundation for understanding and analyzing service systems: respectively work system framework, service value chain framework, and work system life cycle model. The work system framework presents the static components in a service system composed of nine elements: respectively customers, products and services, processes and activities, participants, information, technologies, environment, strategies, and infrastructure. More details of these concepts can refer to the work (Alter 2002). The service value chain framework introduces functions that are associated specifically with services and illustrates how service providers and customers co-create value via collaboration. The work system lifecycle model presents how work systems change and evolve with them by treating the lifecycle of a system as a set of collaborations involving planned and unplanned changes. The three frameworks are jointly used to describe and analyze how service systems are created and how they produce adaptive services. We argue that these three frameworks for work systems can also be appropriate to service systems and they facilitate to understand both the static components and dynamic collaboration in service systems. However, these frameworks are quite conceptual and lack of details about how to design a service system and how to represent ad hoc collaboration and to facilitate service actors' decision makings in collaboration.

Mo et al. (2009) present a virtualization-based service system development method. The general view of their approach lies on that: key service elements (such as service actor, resource, and behaviours) are at first packaged as standardized service components. A service component is defined as a formal package of a service actor, behaviours performed by this actor and all supportive resources to support the actor behaviours, which is a basic unit of a service system that offers a predefined functionality to collaborate with other service components. A service component is a document for provider to publish his services and for customers to query easily. Then service components are virtualized as web service or software and deployed on a service platform.

They distinguish three kinds of service components: including human component, software component, and hardware component. Differ-

ent from the other two components, human components represent the behaviour that is executed by people or mainly depends on people, which may lead to uncertainties. In this work, they consider the software of human components as a software client of the real people that includes three main parts: including tasklist, log, and interface. The tasklist denotes a task schedule for a human component, which records when, where, to whom that the component need to provide services. Log records the using history of the component. The interface is divided into common interface corresponding to the general functions of all software service components and service function interface representing a service function of a component. Since these three service components can be virtualized as web services, they argue that they can be composed together to build a service process to satisfy customer requirements with existing standard used to build/compose a large size web service.

In addition, they also present methods to design a social-technical system. If a customer puts forward his/her demands that refer to constraints on total or local quality indicators, such as functions, time, cost, and credit, on this service platform, the service platform will then collaborate with the customer to obtain a service flow model denoted as a XML file parsed from customer demands to show a set of simple parts the complex service include and describe the sequential logic of these simple parts. Then the platform will choose appropriate service components and organizing them correctly to satisfy the customers. It mainly outputs four parts of information for the customer: including service components set, result of evaluation, and scheduling plan. The service components set are the components selected for each simple part of the complex service. Result of evaluation is the ranking list of service components set for each part of the complex service process based on customer-defined policies, and the scheduling plan is a set of information for calling the service components. After that, the customer can submit the result about his/her satisfactory service component for each simple part to the service platform. At last, the platform will dispatch tasks to those components to execute service interactions with the customer. The advantages of this work lies on that: 1) it takes into account customers' voices when the service platform specifies service flow model based on customer demands and searches for appropriate service component candidates for each simple part of the complex service and ranks them based on customer-design policies, 2) it abstracts service elements into service components by integrating service actors, their behaviours and supportive resources to support these behaviours, 3) it virtualizes complex service elements into service-based service components and represents them as documents, which facilitates providers to publish their services in the Web

and customers to query. In other words, when collaboration, various service actors do not have to take into consideration complex service elements and how they are used to package a service component but simply focus on exposed service information, such as interface and functional and non-functional information about a service for a service component. This logic partially confirms to the idea of "*architecture the line not boxes*" proposed by Gartner Group (2009) to facilitate the integration of global service enterprises interconnected in a system-of-systems structure.

Although this method takes into customers' voices to specify customer demands into XML-based sequential web-service flow model, it fails to propagate requirement changes due to the lack of semantics of the XML-based web service model. Additionally, this approach seeks to virtualize human components, software components, and hardware components as web services and then bundling/composing various web services together to build a sequential work flow for interoperability. However, this approach is simply appropriate to model relative stable activity-based business process but not flexible enough to model dynamic collaboration in service systems.

2.5.3.2 Service Systems as Ecosystems

According to the similarity between service systems and natural ecosystems, Mo et al. (2007) present their service system understandings to describe relationships between various service actors or service systems with an analogy to that in ecosystem, such as mutualism, and parasitism. We argue that their approach helps to understand dynamic behaviours in service systems. However, their model is quite conceptual and descriptive and requires more details to specify the practical design of service ecosystems.

Tung et al. (2007) introduce an intelligent service systems design framework (iDesign) to explore value co-production and service innovation by imitating the ecological symbiosis relationship. They distinguish two kinds of symbiosis relationship: 1) mutualism, and 2) communalism. They also classify their iDesign framework from two dimensions: 1) the continuity of value co-production including levels: respectively mutualism, collaboration, and commensalisms from high to low, 2) the degree of mutual adaptability: respectively one-sided and two-sided from low to high. They classify four kinds of e-service system (mass services, transactional marketplace services, collaborative alignment services, and collaborative personalization services) depending on different compositions of these two dimensions.

In addition, they also use analysis-design methods (i.e., Simple Service Machine (SSM) and Intelligent Service Machine (ISM)) for modelling service systems based on the social-technology systems. SSM is used

to explore why and what underlie the service systems design and it includes two phases: 1) investigation to identify why the firms need the service system for business strategy, and 2) analyze what functions and procedures the service system needs for the business strategy. ISM focuses on how the service systems design could be unfolded and it also includes two phases: 1) respectively design phase about identifying how to design the service system based on the analysis results, and 2) implementation phase about how to implement the service system.

At last, based on the three criteria (proximate response, ultimate response, and evolved dependence) for testing the performance of mutualism proposed by de Mazancourt et al. (2005), they provide a new service system evaluation model for measuring the experience quality and validating value co-production in collaborative service systems.

We argue that their top-down approach provides strategically guidelines to design the service system at design time and to implement it based on the analysis result of SSM and ISM. However, since the service system is quite dynamic and collaborative, their top-down approach is not enough and lacks of common understanding of service systems and can not deal with the problem of the collaboration.

Tian et al. (2008) design service ecosystems using models of value-network, game theory and multi-agent systems (MAS). The value-network modelling is used to describe relationships between business entities by three types of value transactions: respectively goods, services, and revenue. They use the game theory to analyze the behaviours of multiple service actors and how they make a decision in the case of information asymmetry or a decentralized decision-making environment in supply chains. They use MAS principals to model how various autonomous service actors collaborate with each other to solve a problem. We argue that this model is not simply appropriate to model a service ecosystem, but it can be used to model IT-enabled service systems where the value network model can be used to describe roles and relationships of various service actors in collaboration and the decision-making model can be used to model dynamic and ad hoc behaviours of multiple service actors even with the participation of human. However, this model lacks of a side-away approach to discuss the role of IT specialists and lack of models to formalize various knowledge on services in order to facilitate the collaboration of service actors that use different models or languages.

2.5.3.3 Service Systems as Holonic Systems

A holon generally denotes an element that is a part but at the same time a whole (Baina et al. 2006). With this logic, a holonic system can be defined

as a system that works as a whole is simultaneously a part of another bigger system, which means that a holonic system presents the system-of-systems structure.

Maglio et al. (2006) present the service systems characteristics of connectivity and recursivity, which implies that a complex service system is composed of a set of simple service systems interconnected with each other to co-create value. In this sense, a service system can be viewed as a holonic system.

Today, a key characteristic of the service industry is its high information density. Groeneweg and van de Kar (2007) propose to introduce systematic view to service design and they assert that "*Systems thinking includes holistic thinking, i.e. understating the system as a whole in time and space, and all the components and organization involved in a service oriented system have to work together to provide a service that adds value to users*".

Van de Kar (2004) presents a service systems design model that describes the elements of ICT enabled services, which is composed of three components:

- **The service formula** describes the differentiating value proposition demanded by the end-user.
- **The enabling technology** stands for the service architecture providing the necessary technical functions to realize the service.
- **The value network** is the configuration of activities between organizations and the correlated relationships, revenue models and cost structures.

Each two of these three components has trade-offs and negotiates with each other. The trade-offs between the service formula and the technology concern "*how to get the enabling technology to support a service that fulfils the demand of the users?*" The trade-offs between the technology and value network refer to "*how to get a value network in place to provide the necessary technology?*" The trade-offs between the value network and the service formula concern "*how to create a network of actors and coordinate the activities of these different actors to deliver value to the user?*" This method involves large quantities of trade-offs and negotiation activities between service provider and customers to co-design a service for value co-creation. One issue of this model is that it does not discuss how to capture the requirements at the customer level and how the underlying service architecture takes these requirements in account to adjust the functions that realize the service. .

Stanicek and Winkler (2010) provide a framework to explore service systems in terms of context and goal. They define a service system as "a composite of agents, technology, environment, and/or organization units

of agents and/or technology, functioning in space-time and cyberspace for a given period of time". Their proposed modelling framework includes three models: 1) provider-client-target triangle, 2) the conceptual model to denote service systems behaviours, 3) holonic view of service systems. The client-provider-target triangle illustrates the nature of service system and helps to understand a service system. The conceptual model for service systems behaviour is denoted by a 7-tuple: <goal, requirement, service, agent, model, use-case, context>. At last, they discuss the holonic nature of a service system and they argue that a service system is characterized by features, such as autonomous, co-operative, composable, and adaptive. We argue that their presented conceptual reference model provides a clear view to understand dynamic behaviours in service systems and also clearly illustrates how actors collaborate with each other to satisfy a goal in a certain context. However, this approach does not consider the collaboration in establishing the service despite the fact that is goal oriented.

Baida (2006) proposes a service ontology model taking into account the service offering perspective and service value perspective. The service offering perspective describes service elements including their input and output resources, as well as bundling requirements in supplier-oriented terms, whereas service customers can simply express their needs in their own terminologies. To address the gap between the two views, Baida proposes to use a 6-tuple <*need, want, demand, sacrifice, requirement, resources*>. They also present a formal structured language to describe clear and precise customer requirements. The general process of Baida's method lies on that customer needs are firstly concretized into customer wants and customer wants are then specified into customer demands and finally into customer requirements that are composed of customer demands and customer sacrifices. Customer requirements are described by service properties mainly comprising required resources, which are customer benefits in service provider terminology. Accordingly, services that offer the requested resources will satisfy a customer's demand.

We argue that the main value of this work lies on that: 1) provides a common understanding to address conceptual ambiguities by introducing ontology. 2) Provides a top-down approach to concretize customer requirements considering the trade-offs and negotiation between service actors to deal with the gap about service perception and service expectation. 3) Baida provides a production rule to specify customer requirements regulating that "*if customers have demand X with service property L, M, …, N, then offer them a resource described by service property A, B,… C*". The core idea of this production rule is that resources are used to model the relationship between two service systems, which facilitate the integration of

various service systems for value addition at design time. However, this work highlights the automatic bundling of services but lacks of a side-way requirement engineering approach to facilitate the collaboration of various service actors from the same discipline and to propagate changes to customer requirements throughout the top-down approach. It neither explores dynamic changes when bundling occurs during the run time and how it impacts the overall service.

2.5.3.4 Service Systems as Information Systems

Information systems are "*implemented within an organization for the purpose of improving the effectiveness and efficiency of that organization. Capabilities of the information system and characteristics of the organization, its work systems, its people, and its development and implementation methodologies together determine the extent to which that purpose is achieved*" (Silver et al. 1995). An Information System includes people, procedures, data, process software, hardware, and ICT technology (Sue et al. 1999) to enable humans to perform complex tasks that can not be done by human brain, such as handling large amounts of information, performing complex calculations, and controlling many simultaneous processes. As mentioned before, collaborative service systems encounter interoperability issues in collaboration. In order to deal with this issue, we explore approaches from Information Systems.

Elvesæter et al. (2006) introduce an interoperability framework based on Model-driven Architecture (MDA) dealing with how model-driven development is applied to address interoperability, which is composed of three parts: respectively conceptual, technical and applicative integration. Conceptual integration focuses on concepts, meta-models, languages and model relationships to systemizing various aspects (service aspects, information aspects, process aspects, and non-functional aspects) of software model interoperability. Generally, meta-models and ontologies are used to define model transformations and model mappings between the different views of an enterprise system. Technical integration centres on developing software models and execution platforms by providing tools. Applicative integration focuses on methodologies, standards, guidelines, principals, patterns, and domain models to deal with software interoperability issues. They present three reference models to deal with these three kinds of integration for interoperability. The reference model for technical integration is developed from a service oriented view where a software system provides a set of services required by the businesses and users of the enterprises and service implemented by software should provide functionality to support business behaviours or operations of service actors. The ref-

erence model for applicative integration advocates that enterprise and software models can be related in a holistic view, regardless of modelling language formalisms, through the use of meta-models.

The middleware technology has evolved to support interoperability and communication for distributed architectures. In fact, middleware is a software-based enterprise application integration platform that exhibits components to connect two or more software applications, allowing them to exchange data. Issa et al. (2006) present the AgFlow middleware platform enabling a quality-driven composition of web services representing service system functionalities. The AgFlow system architecture comprises several components (i.e. web services, a service broker and a service composition manager) communicating an intermediary service bus. The salient features of this architecture are its multi-dimensional QoS model captures non-functional properties and the adaptiveness to changes during the execution of services.

Another salient middleware framework is proposed by Fujii and Suda (2009). They propose a semantic-based context-aware framework composed of Component ServiceModelwith Semantics (CoSMoS), Component Runtime Environment (CoRE), and Semantic Graph based Service Composition (SeGSeC). CoSMoS models the semantics of components and contexts of users. CoRE is a middleware to support CoSMoS on various distributed computing technologies. SeGSeC is a mechanism to compose an application by synthesizing its workflow based on the semantics of components and contexts of users. The core inspiration of this framework lies on that it facilities the assembling of various applications based on customer requirements expressed in natural language and adapts to different users by utilizing context information when composing applications, and adapts to dynamic environment by autonomously composing a new application and migrating onto it when context changes. It also allows both system designers and users to specify rules on how to compose context-aware applications and as well as model functional, semantic and contextual information and user-specified rules in the same semantic graph format to facilitate their integration.

Gou et al. (2008) present an integration model for business systems and propose to adopt Service-oriented architecture (SOA) as the conceptual architecture to integrate various digital service systems. The principals used by them are based on componentization of business (taking components as the fundamental building blocks and each component has well-defined interfaces and serves a unique purpose). Generally, traditional way to use functions of legacy system is through packaging their function interface as a web service. Now a new solution, namely Software as a Ser-

vice (SaaS) (SaaS, 2008) is proposed to deploy software by hosting an application as a service.

To conclude, most work in designing digital service information systems to ensure interoperability refer to modelling activities in business processes as Web services. These traditional approaches achieve interoperability by matching the outputs of a Web service to the input of its precedent Web services. However, they fail to achieve semantic interoperability since various Web services may have different descriptions of their respective input and output. Description languages (i.e., WSDL 2.0 (WSDL 2.0 2007), WSML, and ontology just to mention a few) are introduced to add semantics to web services, but their composition into activity-based business processes render the system unstable in response to contextual changes (business rules, structural organisations, …).

2.6 Concluding Outlook

Now we are witnessing a notable transformation of our economy into the service-centric, knowledge-intensive, and technology-driven (especially ICT) logic that enables various global service enterprises to interconnect with one another in system-of-systems networks in order to create new services and generate benefits. In such context, various service actors have to collaborate with each other to co-design, co-simulate and co-operate services.

This transformation implies an environment in which traditional design methods are not convenient to understand and conceive new service systems supported by ICT. This environment is characterized with the following facts 1) collaborative processes are quite ad hoc and dynamic. 2) In collaboration, various service actors focus on what to do in response to the exchanged business knowledge instead of how to do it. 3) All service actors (i.e., IT specialists business experts, customers, developers, …) play crucial roles in service innovation and work together to co-design IT-enabled services. The innovation can be initiated by any actors at any level. 4) Various service actors may use different languages or models, which makes the collaboration difficult without a common understanding of the service system and its components. 5) In services, customer inputs are integrated to the whole service lifecycle (design, production, consumption, ...) and any changes in their requirements have to be propagated and consequently adapt the service to ensure customer satisfaction. 6) When collaboration occurs, the activities, which have to be executed, and the required sequence are not often predetermined. Human workers, especially knowledge workers, do require flexibility in their collaboration that cannot be expressed in terms of predefined processes. The service provider, that is to say the

knowledgeable worker, first has to assess customer needs after which further action has to be scheduled accordingly. 7) Customer satisfaction is the ultimate goal to be reached. However, the strategy to follow is unclear in the beginning. The service provider has a variety of options to reach the goal and the decisions made will gradually determine the appropriate strategy to deliver the service. Rather than being predefined, the process of collaboration should evolve. 8) Due to the complex structure of service systems, it is quite difficult to integrate various service systems in a system-of-system value network due to lack of semantic interoperability, and complex and dynamic businesses.

Based on these facts and on the basis of our survey, current top-down approaches for Enterprise Architectures are quite effective to design enterprise information systems that have stable and well-defined business functions (i.e., Sowa and Zachman 1992, TOGAF 2009, FEAF 1999, etc). Additionally, current multi-disciplinary methods and/or models found in the literature and used by practitioners present interesting initiatives to take into account the social attributes of service systems and deal with how human take part in services and how they collaborate with other service actors (human, organization, system, etc.) (i.e., Qiu 2009, Alter 2002, Alter 2008, etc). Other initiatives on service systems provided conceptual models to identify service system components and their relationships (i.e., Tung et al. 2007, Mo et al. 2007, Stanicek and Winkler 2010, etc). In addition, requirement-based approaches rely on trade-offs and negotiations to identify customer requirements (i.e., Rolland 1993, Schmitt 1993, Pohl 1993, Lee 1996, Toffolon and Dakhli 1999, Donzelli 2004, Baida 2006, etc) and dealing with dynamic decision-making in collaboration (i.e., Nigam and Caswell 2003, Kumaran et al. 2008, Muller et al. 2008, Narendra et al. 2009, etc).

However, most of these approaches are top-down approaches and rarely underline the active role of all actors in co-creating the service through collaboration and eventually their participating in all phases of the service lifecycle (design, production, delivery and consumption, …). In addition, current approaches reduce the participation of customers to the preliminary phase (analysis phase) and often do not provide them with appropriate requirement languages close to the natural language to express their needs at each phase of service production and consumption. Due to the system-of-systems structure of services, approaches (e.g. Mo et al. 2009, Baida (2006), etc) that attempt to reduce the complexity of designing these systems mainly focus on modelling the system themselves instead of modelling the interactions among them. This approach is particularly criticized by the Gartner Group (2009), which advocates a new enterprise architecture

("*architect the lines, not the box*"). However, approaches studied in the beforementioned surveys lack of details and reference models to inform all actors at design time how a service system looks like and how it is integrated with its other systems without taking into account their internal and complex structures.

In response to these challenges, we advocate a new thinking way to design IT-enabled services by considering requirements and collaboration. We view service systems as social-technical-economic systems where various service actors including service providers, their competitors, service customers, and other service participants, co-evolve through the relationship of collaboration. In our contribution, in order to deal with these challenges, we introduce a middle-out methodology for modelling and designing IT-enabled service systems driven by requirements and collaboration. The methodology mainly refers to a set of interconnected models namely service system reference model, requirement engineering model, and collaboration model. In the following chapters, we will elaborate the middle-out methodology and demonstrate how to deal with these challenges.

Chapter 3

Service System Design Framework and Middle-out Methodology

Abstract: Traditional top-down approaches for Enterprise Architecture (EA) are quite efficient to deal with enterprises with stable businesses. Nowaday, enterprises have to collaborate with one another to co-create new services and products. They also have to be intelligent enough to adapt their businesses to changes. However, current approaches for EA dominantly focus on the integration of various enterprise business processes, which make them spend too much time, effort and investment on building common businesses rather than focusing on collaboration and co-evolution. This also makes difficult for business people to manage their day-to-day businesses and to adapt them to changes. In addition, there is not a common understanding on service systems, which makes it difficult to formalize knowledge related to business domains and technical domains and support collaboration. In order to deal with these issues, in this chapter, we present a design framework for IT-enabled service systems including a middle-out methodology to indicate how to create/design service systems driven by requirements and collaboration. The framework mainly involves three mod-

els: service system reference model, requirement model, and collaboration model.

3.1 Introduction

In the previous chapter, we discussed the four basic approaches for EA identified by Gartner Group (2010b) and argued that the blended approach of traditional top-down approaches for EA and middle-out approaches are quite effective to design collaborative and dynamic IT-enabled service systems. As a whole, an IT-enabled service system exposes its computational features as services to collaborate with each other in order to co-create value. Internally, IT-enabled service system may be bundled by various autonomous sub-systems through collaboration to produce and expose an innovative and context-aware utility.

In terms of the idea "*Architect the line, not the boxes*" proposed by Gartner Group (2009), an IT-enabled service system has to mobilize its complex internal IT resources and service system components (i.e., resources, competencies, technologies, and delivery channel) to realize its exposed services. When two enterprise service systems collaborate to deal with complex problems, they should not taking into account the integration of respective internal complex businesses but focusing on their collaboration and the delivery channel to deliver services. In other words, a service system expects to consider its partners' internal businesses as black box and it does not have any interest to know how its partners support their exposed services. For example, when a student collaborates with an e-learning service provider, he/she does not care how the provider implements the e-learning services for him/her. More importantly, the student expects the providers to satisfy his/her requirements and as per his/her requirements, the provider will tell him/her how to improve study efficiency step by step through a collaborative way. This could greatly reduce the complexity of their collaboration and facilitate their bundling for value co-creation.

In this sense, supportive to traditional top-down approaches for EA, a middle-out approach is required to connect high-level service actors (e.g., customers and business experts) and low-level service actors (i.e., IT specialists and developers) for value co-creation. To this end, we introduce a service system reference model for designers to identify service system components and their relationships, understand the dynamic behaviours of service actors, and clarify how to connect various service systems through collaboration for value co-creation. So far, the reference model mainly includes three different views of an IT-enabled service system: respectively

the systematic view, the ontological view, and the characteristics view. We will elaborate the reference model in this chapter.

As discussed in Chapter 2, we argued that a service system can be viewed as social-technical system, ecosystem, holonic system, and information system. We also discussed that the widely accepted service system definition relies on the collaboration between the providers and the customers to co-create value (Vargo et al. 2008) and customer inputs as indispensable inputs to a service system (Sampson and Froehle 2006). However, in the context of IT-enabled services, a service system should encompass different design contexts (such as formalizing various kinds of knowledge (i.e., business knowledge and technical knowledge) on service systems, human-to-human collaborations, human-to-system interactions, system-to-system integration and bundling, IT enabled self-services, computational services, context-aware services, and dynamics and evolution), to present its social-technical-economical-systematic features. It thus requires a new understanding specific to IT and service systems to cover service features. It also desires for a big picture to illustrate how to create the IT-enabled service system.

In order to address these issues, in this chapter, we at first agree on an appropriate definition for the IT-enabled service system. After that, we present the service system design framework to provide a global picture for creating IT-enabled service systems. We also introduce the middle-out methodology to globally describe how to create an IT-enabled service system as per requirement and collaboration models. In this chapter, we focus on the reference model and the requirement and the collaboration models are respectively discussed in Chapter 4 and 5.

3.2 IT-enabled Service System Understanding

As discussed in Chapter 2, the definition of service systems are either quite generic to be defined as the value co-creation network between service providers and customers by mobilizing or configuring resources or quite specific for software systems, information systems, ecosystems, or work systems. It lacks of definitions specific for IT-enabled service systems that are characterized by economical-social-technical-systematic nature. To this end, before presenting our work, we at first present our understanding of the IT-enabled service system based on its economical, social, technical and systematic features. We view that the economical feature should be unfolded by value co-creation. The social feature should explore how a person's behaviours influence his or other service actors' decision makings when collaborating with other service actors (human or system). Especially, customers take part in all levels and aspects of value co-creation and deci-

sion makings during the whole service lifecycle (from requirement analysis to service design, then to service delivery and finally to service evaluation). The technical feature should be illustrated through the adoption of technologies, especially ICT, to digitalize service systems and to facilitate various service actors' collaboration of. At last, the holonic/systematic feature should state that a service system can be easily bundled by a set of relatively simple service systems interconnected in a system-of-systems value network without taking into account mutual internal structure/components for value addition.

In this sense, we define the IT-enabled service system as an evolving system-of-systems value chain, in which various networked, interconnected and autonomous service actors (such as human, enterprise, organization, software, and hardware), located in dynamic digital environment (in space-time and cyberspace), collaborate with each other based on various roles (such as provider, customer, and partner) to co-create value for a given period of time. In collaboration, various service actors mainly exchange explicit business knowledge and their collaboration is enabled by respective supporting inputs (i.e., resources, competencies, technologies, and delivery channels). The main goal of collaboration is to bundle a set of autonomous and self-organizing component service systems into a new service system producing customized and innovative IT-enabled services, without taking into account mutual internal businesses. The whole collaboration requires customer inputs as the indispensable inputs to achieve value addition and the co-produced IT-enabled services will be delivered to customers via multiple delivery channels for dynamic customer requirements and satisfaction.

This definition illustrates the following aspects of IT-enabled services:

1- Integrating the following two widely accepted generic service system definitions in the literature: a) value co-creation through collaboration (Vargo et al. 2008); b) customers as indispensable inputs to a service system (Sampson and Froehle 2006).

2- Presenting the economic-social-technical-holonic nature.

3- Emphasizing on the system-of-systems structure.

4- Stating the exchanging of explicit business knowledge in collaboration.

5- Illustrating the basic service system components (i.e., resources, competencies, technologies, channels, customer inputs, service actors, explicit business knowledge, etc).

6- Highlighting the dynamics and evolution of a service system and its value co-creation is just realized in a period of time.

7- Highlighting the adaptability of services to customer requirements, changes and their satisfaction.

8- Presenting that customers take part in all levels and aspects of value co-creation and service lifecycle.

9- Involving the value-added design policy to make service actors focus on collaboration rather than mutual internal businesses.

Since this definition refers to various aspects such as service system components, dynamics, and economic-social-technical features, it is not easy to be understood by all levels of collaborative service actors in service systems. Detailed models and approaches are needed. To this end, we at first present a service system design framework to provide a global picture of our models and approaches to be elaborated to enrich this definition. We also discuss how to create a service system through a middle-out approach.

3.3 Design Framework and Middle-out Methodology

3.3.1 Design Framework

A framework is a group of components that interact and work with each other to generate a consistent output. The service system design framework refers to the design processes or steps to create a service system by using models and methods that are encompassed in the framework. A design framework provides a static view to indicate models and methods for designing a service system and also presents a global view to illustrate the design processes or steps of how to use them to create a service system.

Generally, when designing a system, designers do not have to start from scratch and they always follow the global policy of referring to other projects that have similar problems to their current project. They also expect to reuse current systems to the extent as great as possible for the sake of cost efficiency. In this sense, we view that, before designing a service system, designers can strategically analyze various design frameworks and choose one that is most appropriate to their context. At design time, the chosen design framework can provide them guidelines to create an appropriate service system. To this end, before articulating detailed models, methods, and steps to create a service system, it is quite necessary to present our service system design framework to provide a global view.

As shown in Figure 3.1, our proposed framework is mainly enabled by three models: 1) service system reference model, 2) requirement model, and 3) collaboration model, and it is based on a middle-out ap-

proach to design IT-enables services. In the following paragraphs we briefly introduce them and provide more details in the remaining chapters.

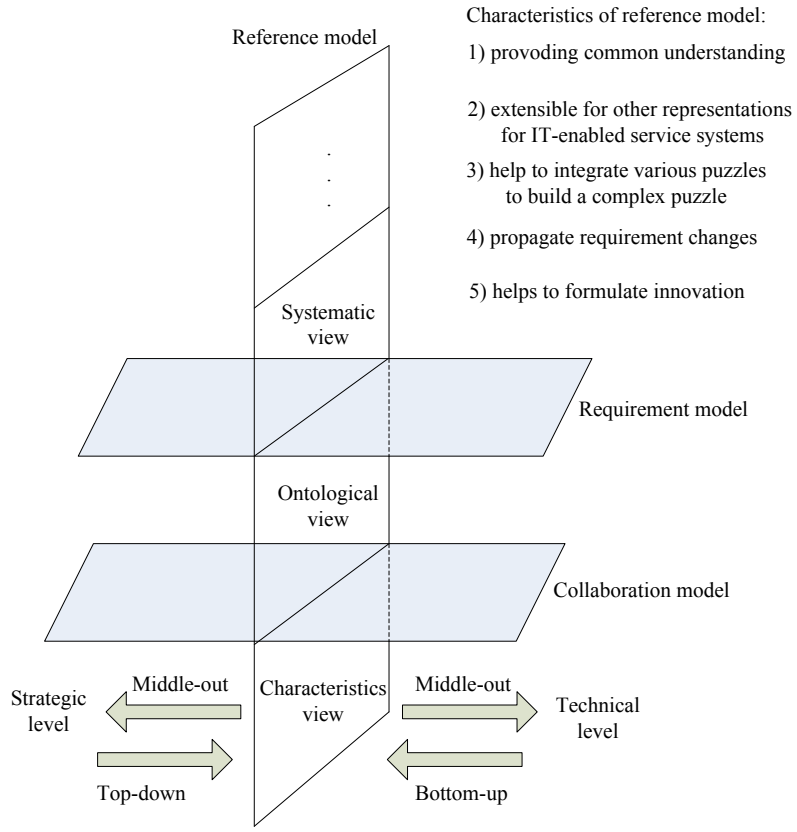


Figure 3.1 Global view of service system design

3.3.1.1 Service System Reference Model

A reference model generally embodies the basic entities that exist in a system and tells how they interact with one another. It also indicates its specific environment or context and involves a clear description of problems in such environment to be addressed (Ramesh and Jarke 2001). Most importantly, a reference model should clearly indicate to its users how to refer to the model to deal with their problems or to achieve their goals.

A reference model is intended to be used by service actors (i.e., customer, business experts, IT specialists, and developers) to identify service system components and their relationships, to develop service system ontology, to specify unclear and imprecise customer needs into clear, precise and structured customer requirements. The reference model can be also used to represent the IT-enabled service system as a set of service characteristics and internal service system components, and to provide guidelines to support the collaboration of service actors to co-design and co-innovate an IT-enabled service to satisfy dynamic customer requirements.

In our context, we introduce the Service System Reference Model including three different views to describe service systems: the systematic view, the ontological view, and the characteristics view. The systematic view mainly describes service system in terms of components and their relationships. The ontological view presents a formal presentation describing service systems, which is mainly used to address the conceptual ambiguities between service actors from different disciplines. In addition, the ontological view formalizes knowledge in the business domain and in technical domain allows knowledge-based reasoning. As a result, the ontological view aims at facilitating and automating the collaboration of various service actors. The characteristics view depicts service features and utilities and makes service actors focus on connections of various businesses rather than mutual dealing with complex internal businesses.

In addition, this view considers services exposed by an IT-enabled service system as a set of service characteristics that encapsulate the qualitative and quantitative features. For example, an e-learning service can be described in terms of service characteristics such as 'knowledge-intensive, web-based, study any time and any location, high efficiency, low cost, freely access to large quantities of digital study resources, flexible teaching plan, multiple teaching methods, multiple interaction ways, …'. Each service characteristic can further be measured by a set of features and constraints on them. For example, the service characteristic 'flexible teaching plan' can be described by 'updatingFrequency, updatingStyle, isShared, …'. The feature 'updatingFrequency' denotes the frequency to update a teaching plan, and the feature 'updatingStyle' denotes who will dominate or drive the changes to the teaching plan (i.e., the student, the teacher, or their joint efforts), the feature 'isShared' denotes if the teaching plan is shared between the student and the teacher. If another service actor or service system (i.e., a student) looks for e-learning services to establish collaboration, he or she are not necessarily interested by the internal structure of the e-learning service system and how it mobilizes its internal components to support the exposed characteristics.

3.3.1.2 Requirement Model

At design time, the service actor at first collaborates with the e-learning service provider to set requirements to the exposed service characteristics of the service system and to generate the expected service characteristics. Then they collaborate with each other to bundle their service systems together for co-designing a value-added service to satisfy the expected service requirements. In addition, the characteristics view helps to improve the service quality at design time since services are described by a set of quan-

titative and qualitative features that customers applied their constraints on them. In addition, the characteristics view also potentially helps to measure customer satisfaction at runtime by evaluating if the services delivered satisfies customers and exceeds their expectation. So far, we have briefly and broadly discussed the service system reference model and we will elaborate it in Section 3.4.

The Requirement Model involves a top-down and sideway requirement engineering approaches. The top-down approach is used to concretize unclear customer needs expressed by customers into specified and clear customer requirements through collaboration in order to address the gap between the service as perceived by the provider and the service as expected by the customer. The sideway approach seeks to represent requirements in a common structured language understandable by all actors belonging to the same discipline. This could facilitate their collaboration. The top-down approach makes possible propagation of requirements from the high level (i.e., customer level) to low level (technical level). These two approaches jointly allow various service actors from different disciplines to co-design the IT-enabled services. In other words, if customer requirements are specified with expressive languages, they should be understood by customers, business experts, IT specialists, and software developers as well.

Therefore, by referring to the characteristics view of the service reference model, customer requirements can be specified as a set of constraints on the expected service characteristics. Since the expected service characteristics are related to IT resources and service system components, whenever customers change their requirements, changes can be propagated and satisfied by re-mobilizing internal service system components.

3.3.1.3 Collaboration Model

As mentioned in Chapter 1, when collaboration occurs, the activities which have to be executed are uncertain and the providers have a variety of options to achieve customer satisfaction. In addition, the customers take part in all levels of decision makings to help the provider to gradually determine the appropriate strategy to co-design and deliver the expected service. Therefore, rather than predefined workflow of activities, the collaboration process will evolve in response to customer input. Further, in the service delivery channel, the exchanged data or explicit business knowledge are sometimes not well structured (i.e., documents and email). As a result, IT-enabled service systems urge a model to represent the delivery channel and exchanged explicit business knowledge and model the dynamic and ad hoc collaboration processes to provide guidelines for service actors to decision

makings and to facilitate the runtime bundling of various service systems for value co-creation.

To meet these requirements, we introduce the Collaboration Model in which we present exchanged explicit business knowledge as Business Artifacts (Hull 2008) and introduce ontologies to provide a common understanding of concepts, terminologies and vocabularies for various service actors to facilitate their interactions. Business artifacts are viewed as self-contained business records that are composed of a set of attributes, some states, and the lifecycle model that reflect the changes among these states.

By referring to the ontological view, we have developed a high-level ontology to deal with service actors' vocabulary ambiguities in collaboration. We also propose to model collaboration processes as a set of interaction activities each of which is modelled as a directed data flow service actors. In the delivery channel, the principal content of a directed data flow is explicit business knowledge encapsulated as business artifacts.

3.3.1.4 Middle-out design approach

Compared to traditional top-down approaches that create a system from high-level organization strategies downward to the implementation level and bottom-up approaches that start from the low-level (i.e., implementation level) and then gradually extends the design process to the whole organization, we adopt the middle-out approach, which starts off with an intermediary level of details to help designers and guide their design process upward stream and downward stream. In other words, the middle-out approach is much higher level than the implementation level but it is still lower than the business level). Any languages or models based on this middle-level will be understandable by actors at the business level as well as actors at the technical level with a certain level of details. Business people could establish their business strategies and design their service utilities according to models and languages provided by the middle-out approach and technical people could specify and implement their underlying information and communication applications to support the perceived services. All actors at different levels could discuss their contribution, exchange their knowledge and participate in the innovation of the perceived services through languages and models provided by the middle-out approach.

Compared to bottom-up approaches, the middle-out approach starts off from a much higher level, which reduces the technical complexities and makes possible collaboration with non-technical service actors. Compared to top-down approaches, middle-out approach is not too strategic / abstract which makes it possible to be understood by technical actors. The

main challenge is the choice of the granularity of details to be considered in the middle-out approach to be flexible enough to adapt to changes in the upper-level or the lower-level requirements. It is worth noting that the middle-out approach is not expected to take over top-down or bottom-up approaches but supportive or complement to them.

3.3.2 Middle-out Design Methodology

The middle-out design process of IT-enabled services refers to the set of steps to follow in order to design service systems and relies on a set of models (service reference model, requirement model and collaboration model) with appropriate level of details to describe service utilities and service system component. The granularity of these models serves as blueprints for all actors to respectively specify their requirements and guide their collaboration. In general, the middle-out design provides strategic guidelines for service providers to identify, specify and build IT-enabled services. Based on these models as illustrated in Figure 3.1, we briefly introduce the middle-out design methodology for IT-enabled service systems that includes the following steps, as shown in Figure 3.2:

Step 1: Designing Generic Service System Ontology

Independent from any specific service, service provider at first identifies fundamental elements that build up their service based on the Systematic View of the Reference Model (i.e., competencies, tangible and intangible resources, processes, technologies, codified knowledge, ..) and then create the high-level service system ontology using the Ontological View (explained later). The high-level ontology describes the general service concepts that are independent of a particular domain.

Step 2: Designing Domain Ontology

In response to a request for a specific service, all actors should be identified with their roles in order to include their participation in various processes such as service innovation, co-design and co-production. After that, service provider design and build domain ontologies to describe domain-specific concepts (i.e., e-learning domain).

Step 3: Concretizing Customer Needs into Requirements

Based on the Collaboration Model, service providers (i.e., service engineers) and customers at first collaborate with each other to concretize customer needs. At this step, customer needs are rewritten with Semantics of Business Vocabulary and Business Rules (SBVR), which is a structured English language. SBVR facilitates the validation of needs with customers. In addition, customer needs are specified as a set of characteristics describing the expected service. Each characteristic is composed of features and

constraints. Business experts and IT specialists reformulate the customer needs with the Service Requirement Modelling Language (SRML) to specify the service characteristics by representing their quantitative and qualitative features and ensure quality (QoS). However, SRML is quite technical and it may be difficult to be understood by other non-technical service actors (i.e., customers and business experts) but it helps IT-specialists and developers to specify the underlying infrastructure with respect to customer requirements.

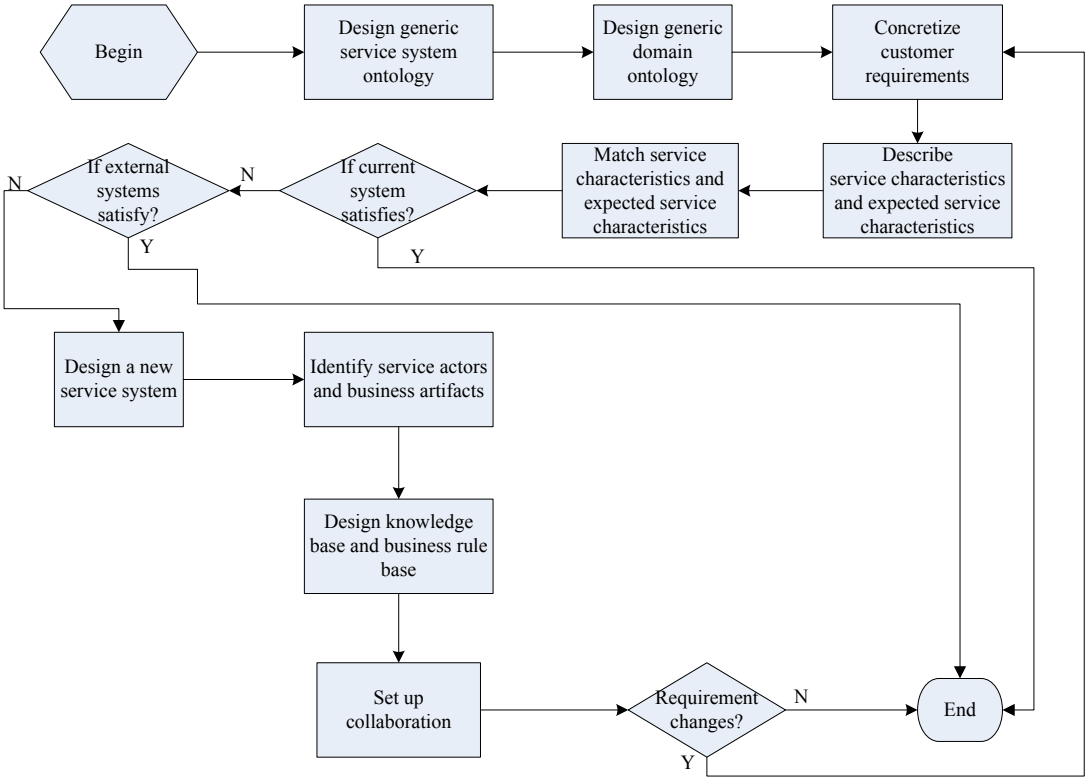


Figure 3.2 General design process for IT-enabled service systems

Step 4: Evaluating whether current services and/or third party services satisfy customer requirements

On the basis of customer requirements, the service provider will assess whether existing internal business services can satisfy customer requirements by matching the exposed service characteristics with the expected service characteristics. If there is no match, then service providers may outsource or inquiry third party services. Since services favourite customization to provide tailored solutions for customer problems, service providers usually adopt an incremental innovation approach, consisting of modifying existing service systems by collaboration with customers to create new services.

The first choice of incrementing existing service will be preferentially taken into account since it can reuse current service system components to the greatest extent of producing new services. If this choice is appropriate, then the service provider will select partners (if exist) and after that, various service actors will set up collaboration based on the collaboration model. Otherwise, a complete new service system has to be created. In this sense, IT specialists identify required service system components to satisfy the customer requirements. They also enrich their designed high-level ontologies and the domain ontologies. This approach promotes an incremental innovation process to create innovative and customized IT-enabled services.

Step 5: Identifying Exchanged Knowledge.

In IT-enabled service systems, service actors exchange explicit business knowledge during their collaboration that we have represented as business artifacts. As mentioned before, a business artifact is a self-contained business record composed of attributes, states, and lifecycle that illustrates how an artifact transits from one state to another state. In general, each concept in customer requirements can be viewed as a business artifact. But in fact, designers have to identify an appropriate granularity for each business artifact. If the granularity of a business artifact is too large, it may lose some aspects or levels of information and, on the contrary, it may lead to redundancy and additional complexity and low efficiency to process them. This knowledge-based approach has the advantage of focusing on exchanged information and knowledge instead of how each actor is processing them.

Step 6: Identifying Business Rules.

IT specialists collaborate with business experts to enrich the generic and domain ontology based on based on identified customer requirements and then to map business rules in natural-like formal language (i.e., SBVR) into implementation rules (i.e., SWRL rules). This rule-based approach has the advantage of excluding the business logic from software application and communication devices into set of rules easy to manipulate and update.

Step 7: Setting up Collaboration

Service providers rely on their collaboration with customers to guide the decision on involved service actors and identify business artifacts and business rules. The collaboration is also extended to cover service design, production and consumption alongside with customers. For example, developers collaborate with IT specialists to implement the delivery channels to provide service utilities over Internet by developing Web services

that can be invoked based on business rules to update exchanged business artifacts.

Since artifacts are self-contained, each partner can select an appropriate computational resource (e.g., Web service or SaaS) to update their attributes and to make them transit from their respective current states to new states with respect to their respective lifecycle. In such a way, partners are not necessary to reveal details about their internal applications or Web services/SaaS but simply invoking appropriate services to updates artifacts. In other words, a service actor sends an artifact with current states to another actor, who receives the artifact and decides, based on current states and shared business rules, which operations should be invoked and what states should be updated before returning the processed artifact to the sender. Building data-driven collaboration based on business rules and exchanged business artifacts has the advantage of being interoperable and stable since the business artifact structures are often stable and applications do not need to interconnect and adapt their interfaces (operations, input messages, out messages, binding protocols, …) .

Step 8: Requirement Changes

An important aspect of our design approach is taking into account customer satisfaction by ensuring that any changes to initial requirements will be top-down propagated throughout the system, namely from unclear and informal customer needs to formal and structured customer requirements and enabling collaboration among customers, business experts, IT specialists and developers. Accordingly, based on the new customer requirements, IT specialists will identify new business artifacts and business rules and the developers will implement the collaboration with different web services or SaaS.

The middle-out design method seems to be similar to traditional methods of designing Information Systems. Nevertheless, this method is somehow differ in many aspects such as the context in which we apply the method that takes into consideration service characteristics (by intangibility, heterogeneity, inseparability, perishability, and customization), the multidisciplinary in collaboration in order to co-create added-value services, and customer satisfaction driven by requirements, which guide how the competencies, resources, technologies, and delivery channels combine to achieve these requirements. In our approach, services evolve spirally throughout the service lifecycle (i.e., from customer needs to the service design and at last to implementation and evaluation) by considering customer input at each phase. However, different from traditional methods, our approach is neither quite strategic nor technical and it integrates various service actors and concepts from various levels (generally including cus-

tomer level, business level, IT specialist level, and technical level) to support the whole service design process co-create a new IT-enabled service system. It is worth noting that our approach is also not universal and effective for all services and it is exactly appropriate to IT-enabled services for KIBS firms in which various service actors collaborate with each other by exchanging explicit business knowledge and IT specialists play an active role to integrate various inputs from service actors.

In the following sections, we elaborate those referred models in our middle-out methodology and particularly focus on the service reference model. We respectively discuss the requirement model and the collaboration model in Chapter 4 and Chapter 5 and present in Chapter 6 a case study to discuss how to use our middle-out methodology in order to create an e-learning service system driven by requirements and collaboration.

3.4 Service System Reference Model

As mentioned before, the reference model mainly includes three different views for a service system: including systematic view, ontological view and characteristics view. In this section, we will elaborate these three views.

3.4.1 Systematic View

In Section 3.2, we have presented our understanding for IT-enabled service systems, which refers to some basic concepts and components used in service systems. The structure of service systems and the interactions among them is depicted in Figure 3.3.

Before presenting how the systematic view helps to understand interactions between providers and customers, we introduce the main components that compose the system:

Front office: refers to the external and available service features that come in direct contact with customers.

Back office: refers to activities that support services delivered to customers and their interactions.

Internal service system component: refers to the composition and mobilization of internal service actors, resources, competencies, technologies and delivery channels to support the services provided by a service system.

Service actor: refers to participants in service systems that co-create value. In service systems, a service actor is an autonomous entity capable of performing tasks in a self-organizing environment. We generalise the concept of actor by distinguishing the following:

1) Human service actor: refers to human being or employees with professional skills and knowledge in service systems.

2) Organizational service actor: refers to an organization or enterprise that works as a whole and operates in an autonomous and self-organizational way in service systems and emphasizes cooperativeness and proactiveness in collaboratively pursuing their common goals.

3) Digital service actor: refers to software systems or platforms that work as a whole and operate in an autonomous and self-organizational way in service systems.

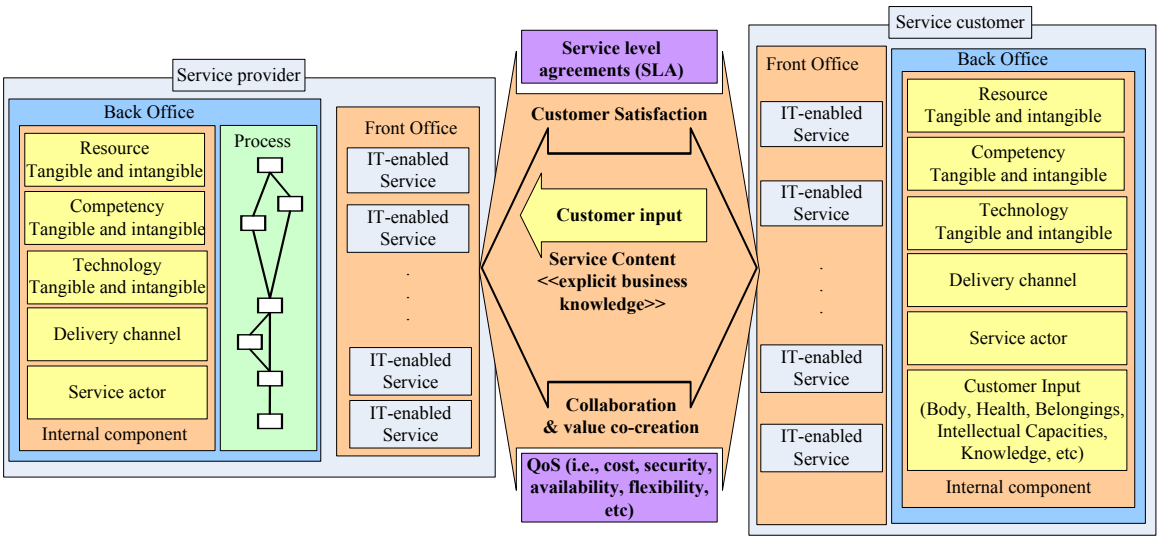


Figure 3.3 Systematic view of a service system

Resources: are consumed to facilitate collaboration between participants for the sake of value co-creation. We identify the following kinds of resources:

1) Computing resources: include hardware, software and data that are designed, built, operated and maintained in order to collect, record, process, store, retrieve, display and transmit information.

2) Capital resource: refers to any asset used in service production. The pivotal capital resources may include building, monetary, equipment, infrastructure, facility, warehouse, etc.

3) Human resources: consist of manual labour and mental labour. In our context, employees provide their time and competencies as input during the whole service lifecycle.

4) Intangible resources: refer to information, experience, knowledge, idea, brand name, skill, intellectual resource, reputation, and relationship.

5) Physical resources: comprise natural resources, such as materials, and water, and physical goods, such as tools, desks, and computers.

Competency: represents the ability to perform a task in a specific context by using knowledge, expertise, skills and mobilizing various resources. We identify the following categories of competency:

1) Tangible competency: derives from codified knowledge, such as knowledge about mathematic and scientific competence. The operation competency refers to specific skills to cope with a problem (i.e., communication skills, team member skills) whereas the technical competency denotes technical abilities.

2) Intangible competency: refers to tacit knowledge (e.g., relational competence and creative competence) and is classified into the following categories:

a) Creative competency: refers to the ability to innovate.

b) Relational competency: refers to the ability to cope with relationship (e.g., social ability).

c) Strategic competency: refers to the ability to formulate strategy, regulation or rules.

d) Management competency: refers to the ability to management.

e) Learning competency: refers to the ability to learn new technologies and skills.

In addition, competencies are in close relationship with explicit business knowledge and they are derived from various sources, e.g., initial education, continuing training, skills, experience interaction or cooperation.

Technology: in service systems, technologies are used to transform traditional service activities into computational components, to automate service processes and create new delivery channels, and to facilitate collaboration among various service actors. In addition, the adoption of technologies fosters innovation and increases business value and benefits for both service customers and providers. Technology can be divided into six categories:

1) Scientific technology: refers to technologies related to sciences such as Chemical technology, and Physical technology.

2) Information and Communication Technology (ICT): includes any communication devices, applications, software or hardware to help users to access, collect, record, process, retrieve, display, create, store, transmit, and manipulate information, such as Web 2.0, SaaS, and Web service.

3) Method: refers to a mean or manner to deal with a problem or an issue, or accomplish a goal, especially in a regular and systematic way, such as system design methodology, ontology design method, and study method.

4) Model: refers to mathematical abstraction or graphical representations of a system, theory, or phenomenon that account for its known or referred properties, such as architectural model, service composition model, and service system model.

5) Mathematical instrument: tools or devices used to build a model, e.g., UML.

6) Organizational technology: technologies used to configure people and organizations for positive growth, e.g., customer satisfaction evaluation, training, and performance management. Organizational technologies provide systematic approach to aligning people, organization and improve how they support each other.

Delivery channels: represent the means by which IT-enabled services are delivered to customers. We categorize three kinds of service delivery channels:

1) Organizational channel: to provide services by virtue of the delivery channel of other organizations or departments;

2) Physical channel: to provide services by using buildings and people as locations and resources, such as letter, newspaper, codified document, and face-to-face;

3) Electronic channel: to provide services via utilizing electronic interaction, such as phone, online, fax, e-mail, and video.

In practice, service actors may introduce an integrated delivery channel that is composed of at least one of these channels. They may start in one channel and then change to another to deliver services, which is quite important to develop a service delivery strategy for service actors.

Customer input: customers input their body (i.e. health care service), knowledge, belonging (i.e. mortgage services), resources, and assets to collaborate with service providers across the whole service lifecycle in service systems. We classify customer input into the following categories: respectively customer competencies, resources, technologies, and delivery channels. Customer input is indispensable for a service system for value co-creation.

Quality of Service (QoS): denotes parameters associated with various aspects of service systems. They are normally considered as non-functional parameters, such as performance, dependability, security and cost. A taxonomy of QoS specifications presented in the work (Quinn et al. 1987) is widely used for discussing QoS parameters and it classifies QoS attributes into metrics (i.e., performance, availability) and policies (i.e. service management). In service systems, we adopt QoS metric to evaluate service behaviours in terms of qualitative and quantitative performance and to measure customer satisfaction.

Service Level Agreements (SLAs): record QoS that govern the way services are delivered to customers by providers and also illustrate the obligation or customer inputs to be fulfilled by the customers to obtain the service (Nitto et al. 2008). In other words, SLAs record the common understanding about QoS, the format of exchanged messages between service actors, the semantic of their exchanged explicit business knowledge, and delivery channels.

Service content: refers to the explicit business knowledge exchanged between service actors during their collaboration.

Customer satisfaction: refers to measures of how services meet customers' requirements. Parasuraman et al. (1991) propose to evaluate customer satisfaction by measuring the gap between service expectation and perception. Berry and Broduer (1999) proposes to use QoS to measure customer satisfaction. In this thesis, we will integrate their ideas and use service characteristics to describe services (service perception) provided by service providers and then they can collaborate with customers to obtain the expected service characteristics or requirement characteristics (service expectation) by setting requirements on the service characteristics.

Processes: in our context, they refer to collaboration processes that bundle various service systems together in order to achieve innovative services by mobilizing various internal or external service system components to support the exposed services.

After presenting the main components in the systematic view, we begin to present how a customer interacts with a provider to co-create value. We view that when a customer is bundled into the process of the service provider, their relationship is modelled as a set of collaboration activities and both of them will focus on these collaborations. Before their collaboration, they have to reach an agreement (SLAs) involving the format of their exchanged messages, the semantic of the exchanged explicit business knowledge and delivery channels. Each service actor reacts autonomously and their collaboration activities are supported by respective internal system components (i.e., resources, technologies, competencies, and delivery channels). If their collaboration goal, actors just have to remobilize respective service system components to achieve the new goal. This will greatly reduce the complexity of bundling services in a system-of-systems network for value co-creation and by focusing on exchanged knowledge to achieve collaboration goals.

The main value of the systematic view is its ability to provide a static view of the service system, illustrates its basic components, and depicts a graphical model representing interactions between service providers and customers. However, the systematic view allows a common under-

standing on the service system and its components but it does not show to process them by information and communication technologies and how to reason on their interactions since they do not have structures and semantic descriptions exploitable by computers and software. In this sense, we propose the ontological view to address this issue and reason on the service system components.

3.4.2 Ontological View

An ontology defines a common vocabulary and allows machine-interpretable definitions of basic concepts in a domain of interest and relationships among them. Ontology is widely used in multiple domains and various disciplines, such as Artificial Intelligence, Computational Linguistics, and Database Theory, Knowledge Engineering, Knowledge Representation, Database Design, Information Modelling, Information Integration, Object-oriented Analysis, Information Retrieval and Extraction, Knowledge Management and Organization, and Agent-based Systems Design (Smith 1998).

Jasper and Uschold (1999) list the following reasons to use ontologies: 1) to assist in communication between human beings, 2) to achieve interoperability (communication) among software systems, and 3) to improve the design and the quality of software systems.

In general, an ontology is composed of "Classes" which describe generic concepts, "Individual" which represent instances of classes and "Properties" which describe relationships between concepts. For example, the relationship 'is-a' denotes the inheritance of two classes whereas the 'has' relationship represents the property relationship between two classes. Finally, the "Axiom" is defined as a set of rules to express facts and logics formula on classes, instances and relationship.

In our context, the reasons why we adopt ontologies are summarized as follows: 1) provide service actors common understandings of concepts and a vocabulary that allow them to share information; 2) enable the reuse of domain knowledge; 3) separate the domain knowledge from the operational knowledge; 4) address semantic ambiguities during in collaboration and improve interoperability; 5) help in creating business rules to update exchanged knowledge and invoke the underlying software.

Our generic service system ontology mainly includes: 1) service system component ontology, 2) service actor ontology, 3) requirement ontology, and 4) collaboration ontology.

Due to the lack of formal common understanding of service system components, we present the component ontology, as shown in Figure

3.4. The component ontology includes three levels: component level, system level, and the system-of-systems value network level. A service system mainly involves internal service system components, customer inputs and exposed services. Internal service system components include resources, competencies, technologies, delivery channels, and service actors. Service actors include digital service actors (i.e., software and hardware), and economic service actors (i.e., organization and enterprises), and human service actors. In addition, a service system is bundled by a set of relatively simple and autonomous service systems in which various services are integrated together to provide new value-added services.

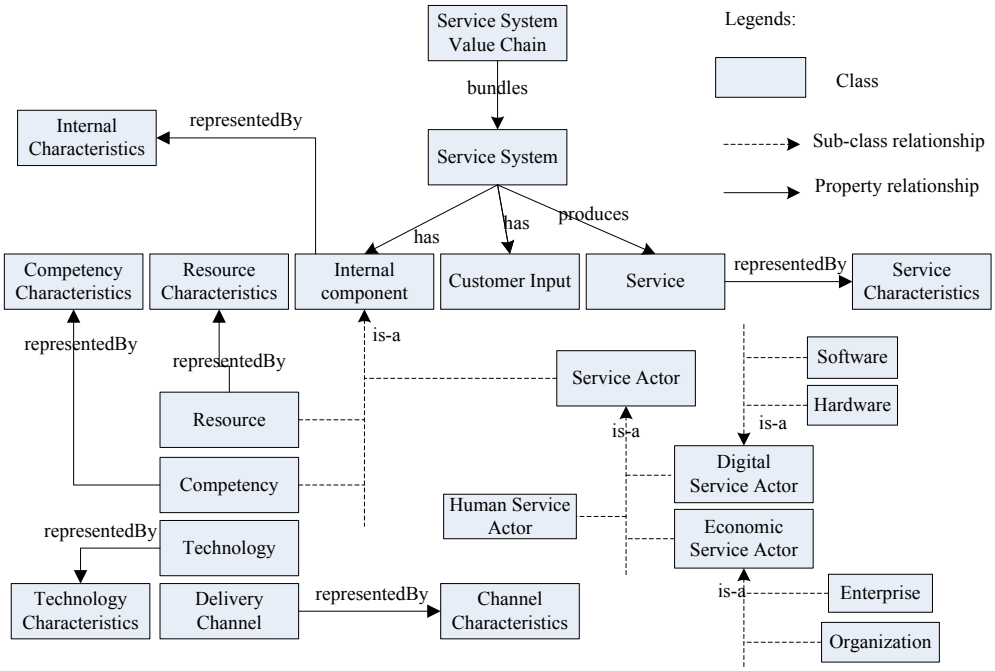


Figure 3.4 Service system component ontology

In order to formalize internal service system components and exposed services in IT-enabled systems, share knowledge, and reason on knowledge and collaboration between service actors, we represent main service system components, services, and customer requirements as sets of characteristics. As mentioned before, Lancaster (1966) argues that both tangible goods and intangible services can be described by a set of characteristics that a good/service embodies. Saviotti and Metcalfe (1984) propose that manufacturers can obtain competitive advantages by offering users particular combinations of three sets of characteristics, technical characteristics, service characteristics, and process characteristics. Based on Saviotti and Metcalfe' characteristic model, Gallouj and Weinstein (1997) consider that product (good or service) can be represented by a set of final characteristics, which can be achieved by participants using their

competencies and technology. They present a formal model composed of four vectors: service provider competency characteristics vector C, service customer competency characteristics vector C', service technology vector X, and final characteristics vector Y. They also indicate that the service innovation can be achieved by remobilizing component characteristics (i.e., competency characteristics and technology characteristics).

De Vries (2006) views that the model proposed in the work (Gallouj and Weinstein 1997) just takes into account the case of one service provider and one service provider. The author thus extends the model to the case of one service customer with multi-service providers and introduce customer technology vector X'. Baida (2006) proposes to formalize resource components, services, and requirements as a set of properties for automating the service bundling.

In terms of these initiatives, the characteristic-based representations have the advantage of enabling innovation by offering actors a simple way to think of services and all components in terms of their characteristics and their utilities instead of technical details (Gallouj and Weinstein 1997). They also allow ontological presentations to formalize knowledge as well as separate internal component functionalities from external service utilities (Baida 2006). The separation of concerns is very useful from customer perspective.

In this sense, we generalise this approach and represent internal components as a set of characteristics, respectively resource characteristics, technology characteristics, competency characteristics and channel characteristics. Since customer input could be resources, technologies, or particular competencies, we denote them with a set of characteristics, namely customer input characteristics:

Technology characteristics: refer to features embodied in tangible or intangible technologies, which describes the utilities obtained from these technologies. A technology characteristic is composed of a set of features and constraints. We list the following features as examples: including name, type, description, etc.

Example: the video conference technology can be used to provide audio and visual conference calls for online discussion groups, which can be represented as {name, type, description, Resolution Quality, Delay, connectionFailureRate, maxConnection, InternetBandWidth, …}.

Resource characteristics: describe the characteristics of a source from which benefit is produced. A resource characteristic is composed of a set of features and constraints and may have features such as name, type, description, comparable, etc.

Example: the credit card resource {name (credit card), type (tangible resource), comparability (false), location (worldwide), holder name (John Zhang), expired date (March 2013), credit limit (5000), valueUnit(euros)}. Yet another example, a teacher can be viewed as a human resource that could be represented by the following characteristics {name, type, age, major, workingYears, title, affiliation, availableTime, …}.

Competency characteristics: describe the characteristics of the competencies used to obtain the service characteristics. A competency characteristic is composed of a set of features and constraints and may have the following features: name, type, description, and proficient level.

Example: in the e-learning case study, the ability for a student to upload online resources can be represented as the following characteristics: {name, type, contentFormat, contentSize, waitingTimeToValidate, creditAward, …}.

Channel characteristics: describe the characteristics of channels used to deliver services and are composed of a set of features such as name, type, and description, ….

Service characteristics: encapsulate domain knowledge that describes a service qualitatively and quantitatively. QoS parameters are associated with various aspects, normally considered as non-functional parameters, such as performance, dependability, security, and cost. As mentioned before, Quinn et al. (1987) present a taxonomy of QoS specifications that are widely used to discussing QoS parameters. That taxonomy classifies QoS attributes into metrics and policies. Metrics include security (confidentiality and integrity), performance (timeliness, precision, accuracy and combinations) and relative importance; policies include management and levels of service. In service systems, we adopt QoS metric to evaluate qualitative and quantitative knowledge in services. In other words, non-functional service characteristics are composed of a set of QoS characteristics and each service characteristic can be represented by a set of features and constraints.

Example: we list the following features as examples: including name, value, value type, unit to measure the value, and description. For example, availability of a service can be denoted as: {name (service availability), value (12), value type (numeric), unit (time), description (describe when this service can be used)}. In addition, in Section 3.3, we have also presented an example from the e-learning domain to describe service characteristics as {knowledge-intensive, web-based, study any time and any location, high efficiency, low cost, freely access to large quantities of digital study resources, flexible teaching plan, multiple teaching methods, multiple interaction ways, …}.

Internal characteristics: represent the features of internal service system components that can be mobilized to support a service characteristic.

Example: the service characteristic 'online lecture' can be satisfied by two internal components: 'Video Conference Technology' and 'Teacher Resource'. Both the 'Video Conference Technology' and 'Teacher Resource' can be represented by a set of features and constraints. For example,

TeacherCharacteristic = {Age, Name, Nationality, Title, Major, Availability, workingYears};

VideoConferenceCharacteristic = {response time, delay, connectionFailureRate, connectionBlockRate, ConnectionNumber, internetBandwidth};

In a service system, the most important component is service actors. Their behaviours are quite complex and they collaborate with each other to co-create and co-innovate service. In order to clearly illustrate their behaviours and collaboration, we introduce the service actor ontology.

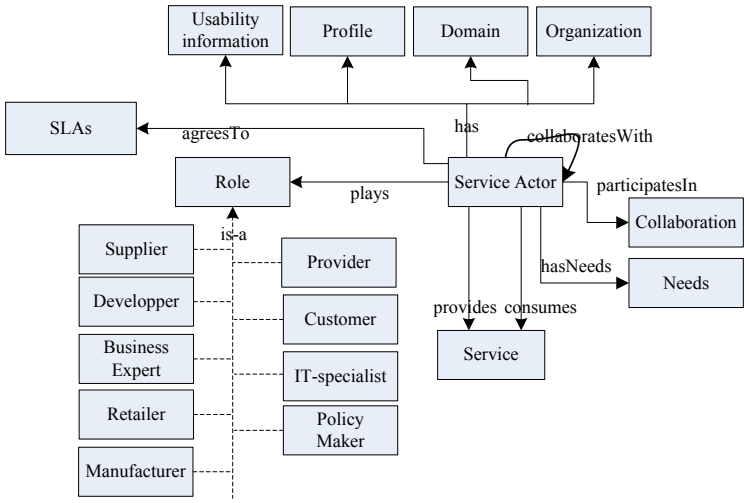


Figure 3.5 Service actor ontology

As shown in Figure 3.5, a service actor has domain that describes a field where the service actor has common interest with other service actors. Service actors play different roles such as providers, customers, dealers, suppliers, clients, business experts, IT specialists, and developers just to mention a few. A service actor may provide and/or consume services. In addition, a service actor also has the following information:

- **Profile**: involves the most basic information of a service actor, such as name, type, role, and behaviours.
- **Organization**: describes the organization or system service actor belongs to.

- **Role**: A role is the abstraction of behaviours in a certain context. Rather than completely, we list the following roles in IT-enabled service systems as examples:
- **Manufacturer**: denotes the role of a participant that produces tangible products. Note that the boundary of manufacturers and service providers becomes more and more blurring since a manufacturer may also provide services.
- **Service providers**: interact with customers and deliver services to them.
- **Customer**: denotes participant that consume IT-enabled services. Customers provide their inputs into the service system and collaborate with providers.
- **Retailer**: denotes participants that sell tangible goods for individual customers.
- **Supplier/Vendor**: denotes participants that supply tangible goods or intangible software services for organizational customers or retailers.
- **Policy maker**: denotes participants that formulate laws or regulations to guide or regulate service actors' business activities. For example, the government formulates long-term strategies or legislations to encourage the shift from good-centric to service-centric economy, to ensure sustainable development or to guide the service provider's behaviours under laws and regulations.
- **Business expert**: refers to participants that have professional skills, knowledge, or techniques to work in the business level, for example, requirement engineers and service designers.
- **IT specialist**: refers to experts work in the IT domain, for example, IT consultant and system architect.
- **Developer**: refers to experts that take part in software design, development and testing.

As shown in Figure 3.6, manufacturers provide tangible goods to suppliers and retailers. Retailer or suppliers provide tangible goods to customers or providers. Suppliers also provide software services to customers or providers. Customers collaborate with employees of providers (including business experts, IT specials, developers, etc) to co-create IT-enabled services. Policy makers ensure the legality of other service actors' business behaviours.

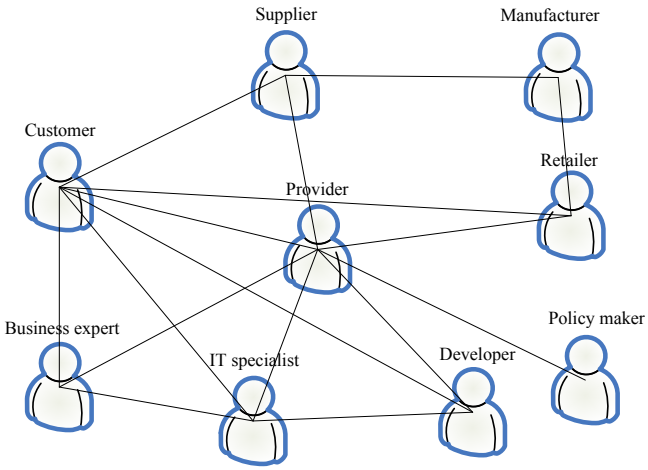


Figure 3.6 Collaboration based on service actors' roles

Since there are gaps between service perception and service expectation and among service actors that may use different models or languages, we propose a requirement model to conceptually deal with these drawbacks. The requirement ontology is illustrated in Figure 3.7. We distinguish four levels of requirements: respectively customer needs, customer wants, customer demands, and customer requirements.

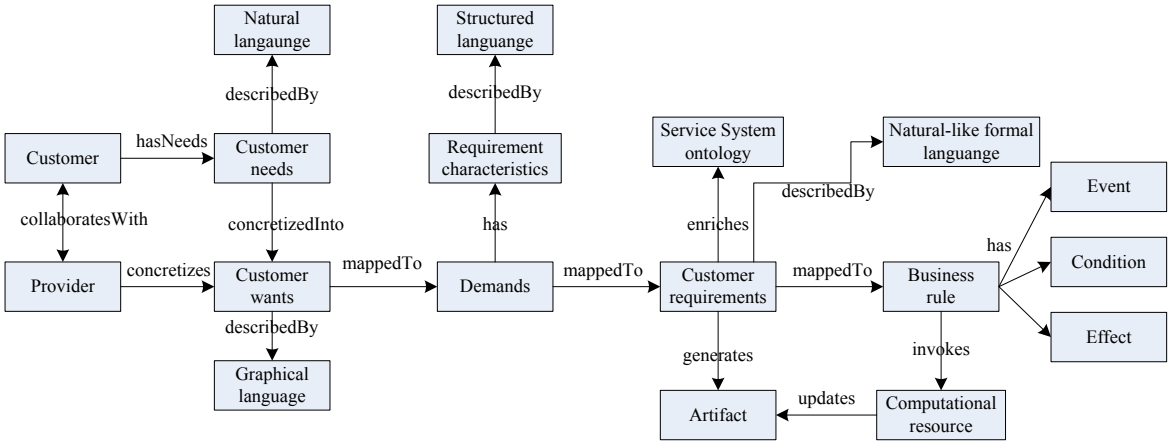


Figure 3.7 Requirement ontology

Customer needs: denote unclear and original expressions to describe what customers want from expected services. Due to the lack of domain knowledge, customers tend to express needs in an ambiguous way using informal languages (i.e., natural language). Needs are prone to issues of ambiguity, incompleteness, and inconsistency. For example, a student wants to improve his English language. This need shows imprecision and unstructured requirements expressed in natural language. At this stage, customers look for providers to develop their expected services.

Customer wants: provides more details in terms of customer needs through negotiations with service providers (Kotler and Bloom 1984). On the basis of customer needs, service providers assist customers to express their needs, to clarify their expectations and to improve the specification of their needs in terms of concise and clear statements. At this stage, providers and customers decide whether a formal collaboration can be established to co-create the expected services. In our context, a graphical model (i.e., UML) can be adopted by providers to collaborate with customers to concretize customer needs since its graphical interface is understandable by both of them. In addition, in this phase, customer needs will be specified by a set of service characteristics related to internal characteristics. In Chapter 4, we will elaborate the appropriate graphical model used to express customer wants.

Customer demands: based on a legal agreement, service providers negotiate and agree with the customers on formal descriptions of their wants to deal with conceptual and semantic ambiguities. From the service provider's perspective, demands are described with structured languages (i.e., SRML) to specify the utilities of the expected services. In Chapter 4, we will elaborate the SRML language used to express customer demands.

Expected characteristics or requirement characteristics: customers set requirements to structured service characteristics related to internal characteristics thereby obtaining expected characteristics or requirement characteristics through negotiation and trade-offs with providers.

Example: a student wants to improve his English language through his participation in a group of students. The customer needs can be represented as a set of structured service characteristics, such as 'group-oriented e-learning', 'freely access to large quantities of digital study resources', and 'flexible teaching plan', where each of these service characteristics are related to a collection of internal characteristics. As mentioned before, for example, the service characteristics 'group-based e-learning' can be specified by two internal characteristics: 'Video Conference Technology' and 'Teacher Resource'. After specifying all the service characteristics with corresponding internal characteristics, customers can set requirements on these internal characteristics thereby obtaining expected service characteristics.

Customer requirements: identify necessary capabilities, characteristics or qualities that the expected service system must be delivered from a system. Requirements are documented in structured formal language to describe what the service system must do (i.e., Functional Requirements), how well it performs during the service consumption (non-functional requirements), what must be done by identifying the necessary

resources (architectural requirements), and what are the imposed limits upon the designed services (constraint requirements). Customer requirements should be easily understood by all service actors and converted to specify technical and business requirements. SBVR can be chosen to describe customer requirements since its natural-like interface can be understood by non-professional service actors (i.e., customers) and also its formal specification is understandable by software or systems.

Business artifact: the exchanged explicit business knowledge is represented as a set of business artifacts, which are self-contained business records composed of a set of attributes, states, and lifecycle model to illustrate how to make a transition from one state to another state. As shown in Figure 3.7, business artifacts can be generated from customer requirements. We will elaborate how to generate business artifact in Chapter 4.

Business rules: are statements to express business logic or control business behaviours, which are composed of events, conditions and effects. The logic of a business rule relies on that if the advent of some event under some conditions, then it will lead to some effects. For example if the advent of the event "begin course" and the student does not have the assigned study material, then the student should buy the study material. In Chapter 4, we will discuss which rule language is appropriate to describe business rules in our context.

Computational resources: denotes web services or SaaS invoked by business rules to update artifacts and execute business logics.

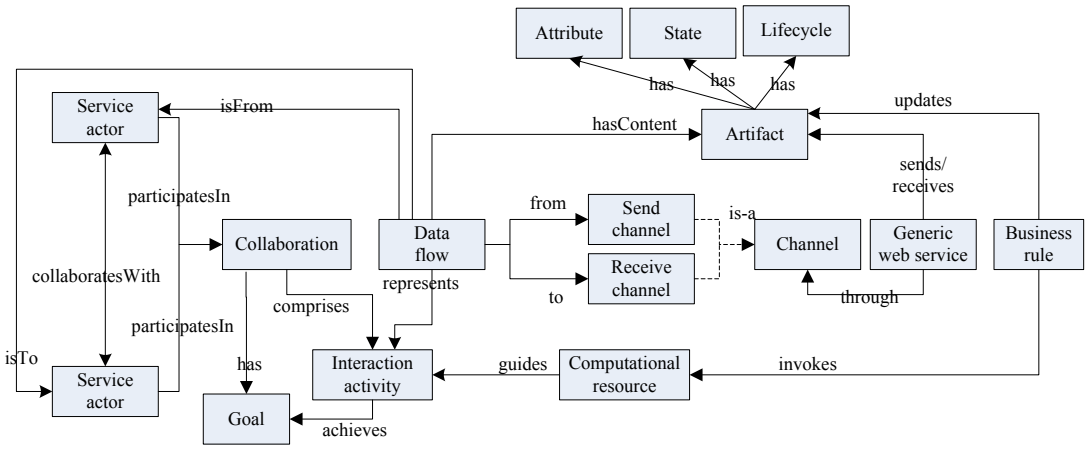


Figure 3.8 Collaboration ontology

As mentioned before, the collaboration of service actors are quite dynamic and the predefined and sequential business process can not model or encompass its dynamics. Also the strategy to achieve customer satisfaction changes in terms of service actors' decision makings. In this sense, we propose an artifact-driven approach to model ad hoc collaboration process

to be presented in Chapter 5. The collaboration ontology provides a big picture of the artifact-driven approach to deal with these issues, as shown in Figure 3.8.

Based on the collaboration ontology, we outline the following features for collaborations in KIBS firms: 1) each **service actor** participates in one or many **collaboration**. 2) Each collaboration activity has a **goal**. 3) Each collaboration goal is achieved by a set of **interaction activities**. 4) Each interaction activity is represented by a directed **data flow** that is from one service actor to another service actor. The data flow has **delivery channel** and it is from **send channel** and to **receive channel**. 5) Each flow is defined by a '**generic' web services** supported simultaneously by the service provider and the service customer. Unlike traditional approaches that attempt to ensure collaboration interoperability by integrating business processes through composing various web services, we propose to use what we refer to as a 'generic' Web service whose role is limited to receiving and sending artifacts. Each artifact is allocated to a 'generic' web service. 6) Exchanged business knowledge in an interaction activity is represented as business artifacts (i.e., order, invoice, etc). 7) Business rules are developed to invoke **computational resources** that are used to manipulate artifact attributes, update artifact states and lifecycle. 8) After receiving an artifact, since artifacts are a self-contained business records, a reference engine will reason on business rules and decides, based on the current state of the artifact and shared business rules, which computational resources should be invoked and what its new state is before returning the artifact to the sender. 9) There are a variety of options to reach a collaboration goal and the decision made will gradually determine the appropriate strategy to guide interaction activities and the invoking of computational resources to update artifacts. We will articulate the data-driven collaboration model in Chapter 5.

The main value of our proposed ontological view provides a common understanding of the structure of information in service system to deal with conceptual ambiguities, to formalize knowledge on service systems, and to facilitate collaboration. However, the ontological view does not show how to bundle various service systems into a system-of-systems value network for value addition and how to formulate innovation in service systems. To this end, we present a characteristics view of IT-enabled service systems.

3.4.3 Characteristics View

In IT-enable service systems, there are various service system components (physical or digital). At design time, when creating a service system, designers have to take into account all levels of service system components, customer inputs, and interrelationships among various internal service systems, which will make their design activities quite complicate and low-efficient. In the run time, an IT-enabled service system is dynamic and emergent, in which customers take part in all aspects of decision makings to collaborate with their service providers to co-create an innovative and adaptive service with high service quality. It is thus a tough task to bundle them for value addition, to formulate innovation, and to propagate changes to customer requirements for adaptability and evolution.

To this end, the characteristics view expects to separate complex internal service system components from exposed IT services for an IT-enabled service system and to describe both of them with a set of quantitative and qualitative features and constraints. When a service system is integrated with another service system to create a new service system, they do not have to consider mutual internal components since their relationships are modelled as a set of interactions via delivery channels by exchanging explicit business knowledge encapsulated as business artifacts. In the ontological view, we have presented some basic concepts of characteristics to be used in the characteristics view, including service characteristics, internal characteristics, and requirement characteristics or expected service characteristics. We will articulate the characteristics view in the following paragraphs.

As mentioned in the generic designing process for a service system in Chapter 3, based on service characteristics exposed by the service system from a service provider, customers collaborate with the provider to negotiate and set requirements to service characteristics for expected service characteristics and then the provider describes it with a formal structured language. Then the provider will mobilize internal service system components to satisfy the obtained expected service characteristics. If there is any change to customer requirements, it is easy to be propagated and to be satisfied by remobilizing internal service system components.

However, besides qualitative and quantitative features of non-functional requirements of services, they also have functional requirements. In this sense, more exactly, service characteristics should be called non-functional service characteristics that denote the quality of exposed services of an IT-enabled service system. In the following of this thesis, if without indication, the service characteristics denote non-functional service

characteristics. In this thesis, the functional information of a service is called functional utilities.

Functional utilities: describe behaviours and functions of services and the way to invoke services. Herein, we present an example to illustrate activities that need to be performed by a student in the run time through a use-case in the e-learning domain, as shown in Figure 3.9. The functional service characteristics may include functions or operations listed in Figure 3.9 (i.e., download online resource). In traditional process-centric approach, at design time, each operation is represented as a set of features, such as input, output, interface, and the well-agreed format for exchanged messages. In the run time, the outputs of an operation have to match the inputs of its sequential neighbour. For example, the output of the 'Log in' operation has to match to the input of 'Upload online resource'. This traditional approach is quite effective to stable businesses but lacks of flexibility for dynamic and ad hoc businesses. In our approach, at design time, each operation is implemented as web services or SaaS and in the run time, its execution is controlled by business rules.

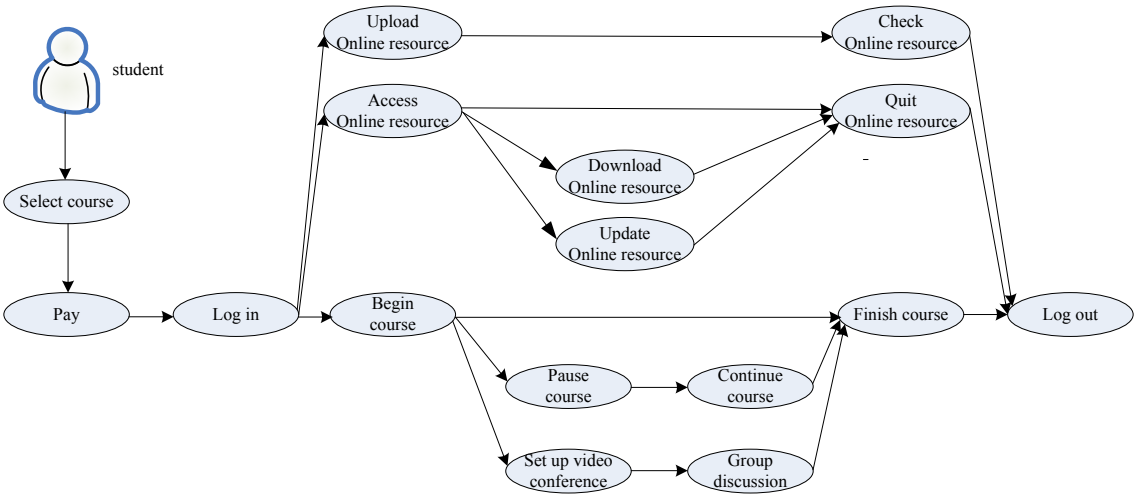


Figure 3.9 A use-case to illustrate functional utilities of an e-learning service

Example: for example, the operation 'download online resources' can be controlled by the rule: 'if a student logs in, he/she has the authority to download, and the file size is less than the student's allowed max size, then he/she can download the file via the specified online download interface'.

In an IT-enabled service system, a service system could be a customer and a provider at the same time, it has to consume services from some service systems and provide services to other service systems as well. When a service provider collaborates with a service customer, customer inputs are necessary. In this sense, a service system exposes its services to its

customers and expects customer inputs from its customers. We denote expected customer inputs as expected customer input characteristics that have the same structure with customer input characteristics.

Further, as mentioned before, clear and precise customer requirements are represented as requirement characteristics or expected service characteristics. Therefore, the characteristics view of a service system may have internal service components characteristics including resource characteristics, technology characteristics, competency characteristics and channel characteristics, customer input characteristics, expected service characteristics, functional utilities, exposed service characteristics, expected customer input characteristics. In addition, a service system also has delivery channel for exchanging business artifacts. The characteristics representation of a service system is shown in Figure 3.10.

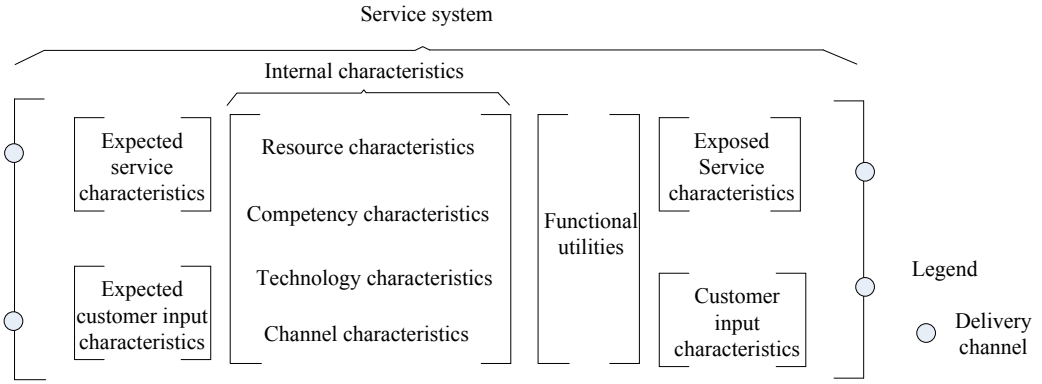


Figure 3.10 Characteristics representation of a service system

The main value of the characteristics view of a service system is that it separates exposed services from internal service system components of an IT-enabled service system and makes service systems focus on their own business and the delivery channels with other service systems. It also helps to formulate innovation. In the following paragraphs, we at first, through an example, present how various service systems are bundled together without taking into account mutual businesses and internal components. Then we discuss how it helps to produce innovation.

As shown in Figure 3.11, the service system E (bundled by A, B, and C) and D are integrated together. We view that if two service systems (i.e., A and B) are possible to be bundled, the exposed service characteristics of A has to match the expected service characteristics of B. Conversely, if the service system B expects to collaborate with the service system A, B's customer inputs characteristics have to satisfy the expected customer input characteristics of A. As mentioned before, service providers and customers collaborate with each other to specify customer needs by set-

ting requirements to service characteristics and customer requirements are described by expected service characteristics.

In this thesis, we provide a general bundling rule by assuming that customer requirements for an expected service described by a set of expected service characteristics can be satisfied by an appropriate mobilization of internal service system components characteristics (resource characteristics, competency characteristics, technology characteristics, and channel characteristics). Although in this thesis, we are not expected to propose a bundle model to discuss how to bundle two service systems by matching characteristics, the characteristics view presents strategic guidelines to integrate two service systems together by comparing service characteristics and expected service characteristics. In the next chapter, we will present a formal structured language to represent service characteristics.

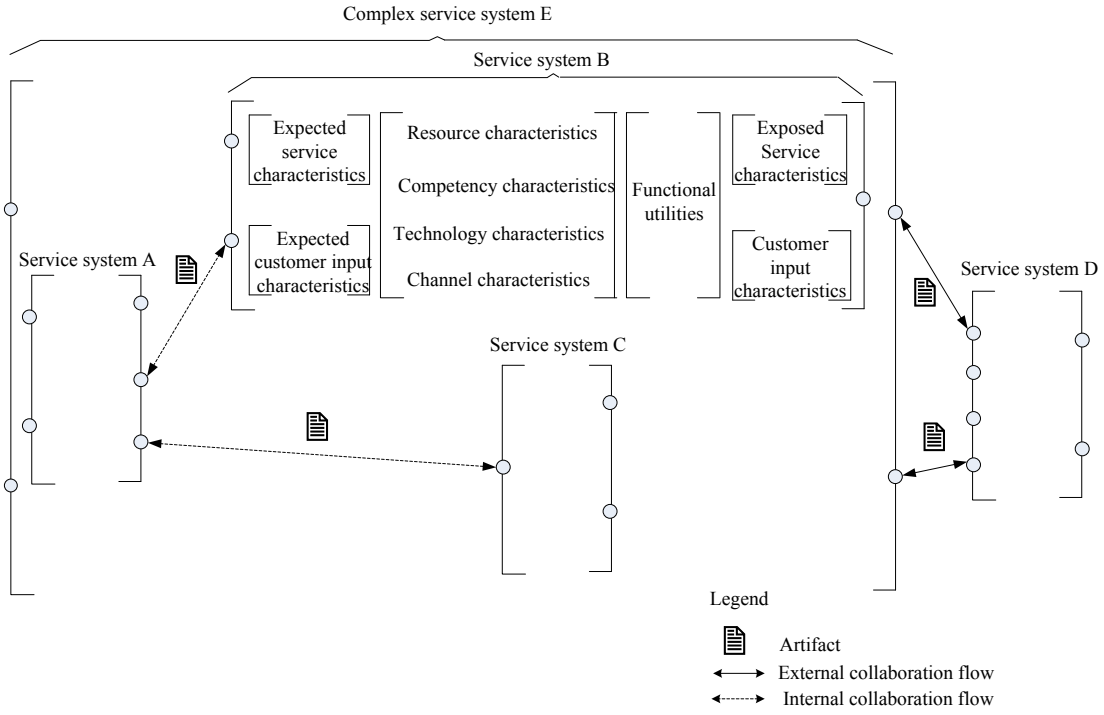


Figure 3.11 Characteristics representing service system value chain

In addition, a service system exposes its service characteristics to its partners for collaboration. In collaboration, various service systems do not necessarily consider respective internal service system components, but they just have to know how to establish collaboration with one another, which delivery channel they should adopt in collaboration to exchange business artifacts, and what kinds of explicit business knowledge (business artifacts) they will exchange in collaboration. For example, as shown in Figure 3.11, the service system E comprises the service system A, B, and

C. In this example, it is not necessary for the service system D to be clear of the internal structure of the service system E when it sets up collaboration with the service system E. The service system D just has to be clear of their delivery channels, exchanged business artifacts and when to invoke computational resources to update artifacts.

In addition, another major value of the characteristics view is to formulate innovation. Christensen and Tan (2000) view that an innovation of products should contain a difference that induces appropriate, valuable, and desirable effects on various participants, such as consumer, manufacturer, service provider, society, and environment. Similarly, we view that different compositions of service system components will result into different service characteristics and different service characteristics may lead to innovation (Peng et al. 2010a). In other words, service innovation will be generated by modifying service characteristics resulting from the different compositions of internal service system components.

For example, in mail delivery services, letters are mailed based on priority and can be classified into three categories: respectively ordinary mail, registered mail, and EMS (Express Main Ports). The three different priorities share the same competency characteristics, technology characteristics, and resource characteristics, but hold differences in the delivery channel characteristics. EMS obviously costs more but has the benefit of being faster than the other two solutions.

The changes to those aforementioned characteristics (i.e., resource characteristics, customer input characteristics, service characteristics, etc) can generally be hold in the following three ways: 1) via adding new elements, 2) via deleting existing elements, and 3) via updating or replacing current elements. In addition, based on these innovation styles, we identify the following innovation patterns that may lead to service innovation: respectively radical innovation, improvement innovation, combinative innovation, customer-driven innovation, policy-driven innovation, technology-driven innovation, and competition-driven innovation.

1- Radical innovation: this kind of innovation embodies huge changes in service characteristics which results from large quantities of changes to aforementioned characteristics.

Example: for example, compared to traditional learning mode, the e-learning could be considered as a radical innovation. Taken another example, a software firm traditionally earns its money by selling software to its clients but now it rents its software to clients or offers its software to clients for free or in relatively lower price but earning its money by selling services, such as maintenance, training, and consultancy.

2- Improvement innovation: this kind of innovation does not hold huge changes to service characteristics but the improvement of current business process (adds some new elements or replaces some elements in service characteristics), which may reflect slight changes to those component characteristics. This kind of innovation is always driven by the improvement of efficiency and the reducing of cost and it principally holds changes to the provider's competency characteristics.

Example: for example, the car manufacturer optimizes its production process to improve efficiency and reduce cost by introducing online expert system for online diagnosis.

3- Combinative innovation: this kind of innovation refers to the combination of several simple services to a complex service.

Example: for example, a telecommunication company (e.g., T-mobile) binds several services together (such as mobile phone, fixed telephone, Internet service, and cable television) for customers.

4- Customer-driven innovation: this kind of innovation is driven by customer requirements thereby providers have to provide customized services to satisfy the customer's dynamic requirements. This innovation holds huge changes in requirement characteristics.

Example: Ikea provides customized services to allow customers to online set parameters of their desirable products (e.g., size, colour, style, and the way of delivery).

5- Policy-driven innovation: this kind of innovation is enabled by policy makers' policies, which then leads to changes to other component characteristics.

Example: for example, if the government forces a manufacturer to reduce its carbon emission with laws, which may increase its cost in producing the same product, then the manufacturer has to optimize its production process or innovate its business process to create a new service.

6- Technology-driven innovation: this kind of innovation is driven by the emergence of new technologies. New technologies can enable the changes to competencies, delivery channels, customer inputs, which will always lead to innovation.

Example: traditional shopping and shopping online.

7- Competition-driven innovation: service providers imitate their competitors to reconfigure current business process in order to produce similar services to that of their competitors or they make some changes to their competitors' business process for surviving in fierce competition.

Example: competitors (e.g., Subway) imitate to the linkage model of Mcdonald's but always holding some differences that may lead to innovation.

Finally, we generalize the following characteristics or advantages of the proposed reference model:

1- Indicates static service system components and their relationship and dynamic behaviours.

2- Provides common understanding of shared information in service systems and formalizes various kinds of knowledge on service systems via ontology.

3- Supportive to traditional top-down approaches for EA, provides a middle-out approach integrate various puzzles to build a complex puzzle to comparing service characteristics for complex problems through collaboration and to model complex relationships between service systems as a set of collaborations that exchange explicit business knowledge encapsulated as business artifacts hiding their mutual complex internal structure.

4- Makes possible to propagate requirement changes and design adaptive services.

5- Helps to explore innovative IT-enabled services.

3.5 Comparison with Some EA

In this Section, in order to illustrate the value of our middle-out methodology, we compare it to three famous EA: including ISA, TOGAF, and FEA.

Our approach is partially influenced by the ISA proposed by Sowa and Zachman (1992) but present more details to design an IT-enabled service system. Sowa and Zachman in this work present six architectural dimensions to describe and design an enterprise information system: respectively data-based (what), process-based (how), location-based (where), people- or organization-based (who), time-based (when), and motivation- or strategy-based (why). They also present six player perspectives (planner, owner, designer, builder, subcontractor, and enterprise). In our work, we mainly take into account the collaboration of the following players: customers, requirement engineer, IT specialists, developers and internal or external service enterprises. So far, in our reference model, we take into account three different views to integrally describe a service system and each view includes some architectural dimensions (i.e., data, process, people, and motivation) from the proposed six architectural dimensions in the work (Sowa and Zachman 1992). The data dimension in service system denotes resources, competencies, knowledge, and ontology. The process dimension in our domain denotes collaboration process. The people dimension denotes service actors (human, organization/enterprises, software, and hardware) in

service systems. The motivation dimension denotes business rules in our context. However, their proposed ISA does not refer to any detail approaches or methodologies to create a service system. In addition, their work simply takes into account the interactions between two levels of players that locate next to one another in their top-down hierarchy. In our work, we consider the collaboration of various service actors to co-create a service system and present a middle-out methodology supportive to traditional top-down or bottom-up approaches.

TOGAF mainly includes business architecture, application architecture, data architecture, and technical architecture (TOGAF 2009). Business architecture mainly describes the processes used by business uses to achieve its goals. Application architecture describes how to design specific applications and how to integrate them together. Data architecture describes how the enterprise data repository or warehouse is organized and accessed. Technical architecture describes the hardware and software infrastructure that supports applications and their integrations. TOGAF also involves Architecture Develop Method (ADM) that is mainly composed of eight phases, which provides strategic guidelines to create an enterprise architecture, as shown in Figure 3.12. However, the detailed approaches adopted in each step are quite flexible. Col and Beveridge (2003) view that "*TOGAF is not wholly specific with respect to generated documents; in fact, it provides very little in the way of prescriptive document templates—merely guidelines for inputs and outputs*". Roger Sessions (2007) view that "*TOGAF merely describes how to generate an enterprise architecture, not necessarily how to generate a good enterprise architecture*".

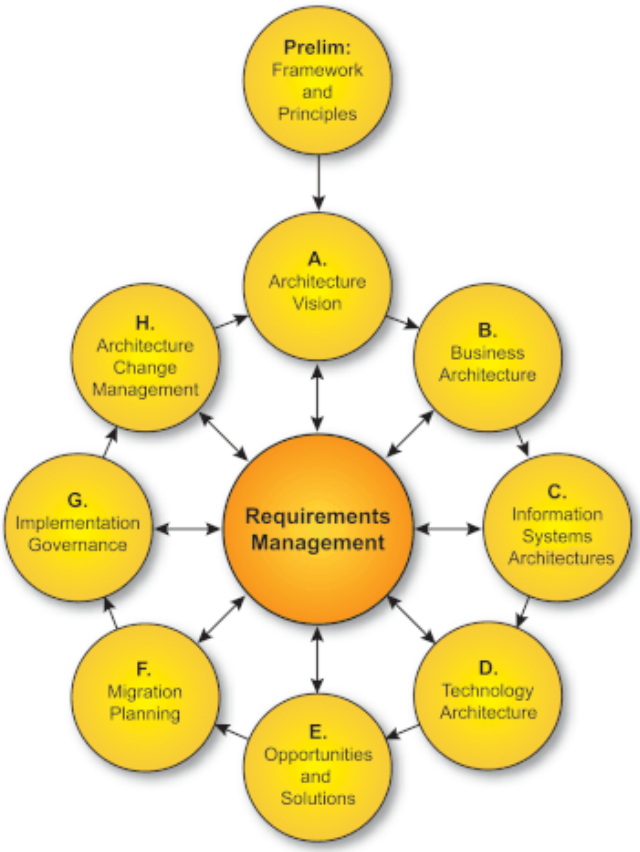


Figure 3.12 TOGAF Architecture Development Method (ADM) (source: TOGAF 2009)

In this study, our service system architecture includes business architecture (see Chapter 3), process architecture (see Chapter 5), and technical architecture (see Chapter 7). Supportive to TOGAF, our service system architecture presents approaches to formalize knowledge from business domain and technical domain to automate collaboration. It also involves a service system reference architecture to separate internal complex business components from exposed services through its characteristics view and helps to "*architect the line, not the boxes*" (Gartner Group 2009). In addition, it refers to a top-down requirement engineering approach to concretize unclear and imprecise customer requirements into clear and precise customer requirements and a side-away approach to support the propagation of requirement changes for adaptability. Rather than process-centric, it includes a data-centric approach to model dynamic and ad hoc collaboration, which makes business people focus on their businesses rather than integrating various business process. Supportive to the TOGAF, these approaches to be presented are potential to ensure an adaptative and flexible IT-enabled service system.

FEA (FEAF 1999) adopts the taxonomy of Zachman's framework (Zachman 1987) and also involves methodologies to create enterprise architecture. The core idea of FEA is that an enterprise is built of segments of different views on federal enterprises. For example, Figure 3.13 lists the different views of a federal government from the FEA Practice Guide (FEA Practice Guide 2006). FEA also provides five reference models, respectively focusing on business level, component level, technical level, data level, and performance level. The five reference models from FEA can be used to develop a common taxonomy and ontology for describing IT resources and to facilitate collaboration across the federal government in order to ensure interoperability by providing standard terms and definitions for the domains of enterprise architecture.

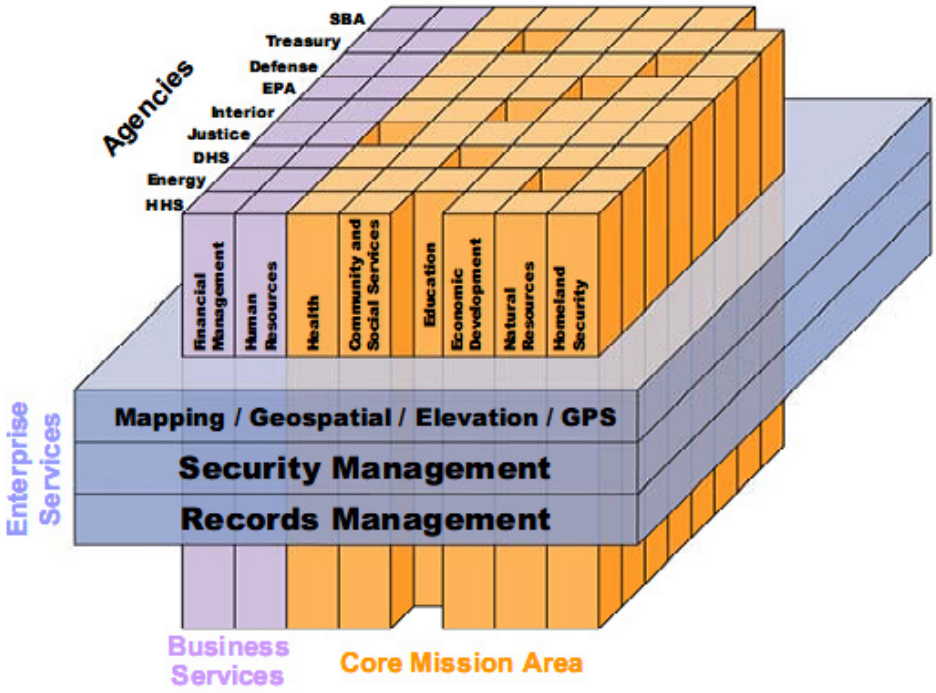


Figure 3.13 Segment and services in federal government (source: FEA Practice Guide 2006)

Leist and Zellner (2006) view that one of disadvantages of FEA is that "*it does not contain a detailed description sufficient to generate specification documents for each cell/segment in FEA and each organization has to define the meta-model for each cell, such as data, process, network, people, time, and motivation*". Compared to TOGAF, it lacks of detailed approaches to tell how to create an enterprise architecture step by step. In addition, Since FEA includes five reference models, FEA process to tell how to segment enterprises, and FEA success measurement to measure organizational success, which make it difficult for architects to be commanded in

practice. As a result, it may lead to the whole project quite long. In addition, FEA targets at "*transforming the U.S. Federal government into a citizen-centered, results-oriented, and market-based organization as set forth in the President's Management Agenda*" (FEA Document 2006).

FEA seeks to provide guidelines for all kinds of complex distributed and decentralized enterprises to design enterprise architecture. Compared to FEA, our target is specifically to create a collaborative, innovative and adoptive IT-enabled service system for KIBS firms. In this sense, our research domain is a sub-set of that of FEA. In our research domain, service actors mainly collaborate with one another by exchanging explicit knowledge for value co-creation. In addition, our reference model mainly provides a middle-out approach, supportive to traditional top-down approaches, to facilitate the collaboration of various service actors. Our requirement engineering approach could be supportive to FEA to deal with the requirement gap between service perception and service expectation and also to ensure the propagation of requirement changes for adaptability. In addition, our collaboration model can also be used in FEA to deal with collaboration between various distributed service actors if they mainly exchange explicit knowledge, such as be federal officials and citizens. In addition, as mentioned before, our blended (top-down and middle-out) approaches for EA to be presented in this study main follow the strategy suggested by Gartner Group (2009, 2010a, 2010b), which could make service participants focus on collaboration rather than mutual businesses for value addition. In addition, compared to FEA, we presented detailed process to create an IT-enable service system (See Figure 3.2).

3.6 Conclusion

In today's economy, firms increasingly focus on their core business processes and propose their expertise as services to partners. In addition, knowledge is considered to be one of the main sources of sustained growth for current service economy, especially for KIBS firms. Also, ICT greatly influence the businesses for KIBS firms and make possible for service actors to expose their computational features as e-services.

However, as an economic-social-technical system, service system is quite complex, dynamic and emergent and it always refers to large quantities of collaboration between service actors (human, organization, software, hardware, etc) who have complicate behaviours, such as knowledge sharing, dynamic collaboration and evolvement to co-create value.

In this chapter, we present a new definition of service system covering the economical-technical-social-systematic feature. Further, in order to design a service system for value-added services, we present a service

design framework mainly composed of three models: service system reference model, requirement, and collaboration model. The proposed reference model conforms to Gartner Group's core idea for EA in 2009, namely "*architect the line not box*". As per the framework, we broadly discuss the middle-out methodology to design an IT-enabled service system driven by requirements and collaboration.

Our reference model includes three kinds of views to describe an IT-enabled service system: respectively systematic view, ontological view, and characteristics view. The systematic view illustrates fundamental service system components and the structure of a service system. However, it lacks of dynamic representation of service system behaviours and relationships. The ontological view describes dynamic behaviours with a formal model, which addresses the conceptual ambiguities, provides a common understanding of service systems, and facilitates the collaboration of service actors. One challenge may be that it is a tough task to develop an ontology for an IT-enabled service system due to its complexity and diversity.

The characteristics view separates complex internal service system components from exposed IT services for a service system. When one service system is integrated with another service system to create a new service system, they do not have to consider respective internal components since their relationships or collaboration are modelled as a set of interactions via interaction channels. The characteristics view also helps to design innovative services at design time and measure and adapts to dynamic customer requirements in the run time for customer satisfaction. It also presents guidelines to formulate innovation. However, it may raise the following two challenges: 1) difficult to evaluate the weight for various characteristics; 2) we do not provide models to match service characteristics to expected service characteristics for establish potential collaboration relationship at design time.

Chapter 4

Requirement Model for Service Systems

Abstract: In IT-enabled services, customers and providers differ in terms of their expectations on what the customers really want, what the providers will offer and how they design, develop and provide these services. In other words, service actors, namely service consumers, business specialists, but also IT-specialists and developers most likely use various models and/or languages to describe service needs from different perspectives and thereby encounter difficulties in collaborating and co-creating added-value services. In this chapter, we present a top-down and side-way requirement model for IT-enabled services to respectively address the gaps between service actors. The originality of this requirement model stems from the top-down requirements on service characteristics based on our proposed IT-

enabled service system architecture, and on the application of constraints on the service system components at customer, business and technical levels. From a technical perspective, the proposed requirement model relies on a structured English-like language, the Semantics of Business Vocabulary and Rules (SBVR) language, as a common business requirements language convenient for business professionals and customers. IT experts deduce from the SBVR-based requirements business objects (i.e., business artifacts) and business-technical rules, expressed in the Semantic Web Rule Language (SWRL) to invoke software. The requirement engineering model substantively stimulates interactions among all involved service providers and customers throughout the whole service lifecycle.

4.1 Introduction

In the previous chapter, we presented a service system reference model. In this chapter, we will discuss how to refer to the proposed reference model to concretize unclear, unstructured, imprecise customer needs into clear, structured and precise customer requirements to deal with the gap between service perception and service expectation. We will also present how to facilitate interactions among service actors of the same discipline that use different models or languages to specify their requirements with a set of requirement characteristics or expected service characteristics related to internal characteristics.

The initial and key step in designing service systems for IT-enabled services consist of identifying customer requirements and the service utilities that should be clearly described by customers and well understood by service providers. Well-identified and well-specified customer requirements not only improve the innovation process in designing the expected service by various service actors, but they also increase customer satisfaction and improve loyalty towards service providers (Baida 2006). Building IT-enabled service systems based on customer requirements requires a rigorous engineering approach to identify and specify customer, business, and technical requirements through collaboration. Unfortunately, traditional requirement models are developed for software or/and system developments and therefore encounter the following challenges if they are applied to IT-enabled service systems:

1- Customers are typically inexperienced in service engineering and lack domain knowledge in IT. This potentially disrupts effective co-design of the service with the provider. When a new service is needed, customers thus encounter problems to exactly express or identify appropriate requirements to take full advantage of IT services in increasing the value of the expected service.

2- Vasarhelyi and Greenstein (2003) argue that customers and providers frequently have different views on service perception and service expectation. Therefore, all service actors should collaborate with each other to reach an agreement on the needs, to co-design the service system and to co-produce value through the whole service lifecycle. As a result, any requirement model or language should be expressive enough to be understood by customers and rigorous enough to be processed by software.

3- As opposed to traditional services (i.e., transportation services), Information Technology (IT) specialists play important roles in IT-enabled services. They should not only collaborate with business analysts and customers to specify customer requirements and design business services, but also interact with developers to specify and design software and communication infrastructures that support customer business services. In addition to being brokers in the service design, they play important roles in service innovation since IT services have become inevitable enablers for innovation. As a result, any requirement model should rely on IT resources and service system components.

4- In IT-enabled service systems, various service actors including customers, business analysts, service engineers, IT specialists, and front and back-office staff should work together to innovate and create added-value services. Unfortunately, they often rely on different models and/or languages related to their inter-related disciplines, which raises a challenge about how to find a common understanding that enables them to jointly design innovative services by varying the combination of their resources and competencies, and ensuring the propagation of changes at any level of the service system.

In order to help customers to express their needs with their natural-like languages and agree on a well-structured requirement language with the service providers and impose the follow-up of these requirements downward by administrative officers, business analysts, IT-specialists and developers, the abovementioned challenges should be addressed to particularly handle the requirement gaps between customer expectations and perceptions, and provider intentions of designing and delivering IT-enabled services, and the gap to facilitate collaboration among service actors. We tackle these challenges with two complementary approaches:

1- A top-down requirement approach to guarantee that the service is driven by customer needs, their requirements and that all underlying resources of service providers are mobilized towards this end.

2- A side-way requirement approach to facilitate interactions among service actors of the same discipline (i.e., domain of expertise) in order to specify their requirements in an expressive language and make

possible propagation of further changes through the top-down requirement approach.

The remainder of this chapter is organized as follows: in Section 4.2, we briefly survey current top-down requirement engineering approaches in the literature and discuss how their work influences our approach. In Section 4.3, we present the high-level requirements framework to discuss the general process to identify customer requirements and the approaches or models to be used. In Section 4.4, we articulate our requirements engineering approach by referring to our proposed reference model. We theoretically discuss the mapping from SBVR to OWL 2 DL and SWRL in Section 4.5. And we will conclude our work in Section 4.6.

4.2 Related Work

The Requirements Engineering (RE) discipline describes the general process of identifying requirements, including at a minimum: elicitation to gather the requirements from stakeholders, analysis for checking for consistency and completeness, documentation or specification, verification to ensure its correctness, and management and maintenance (Shroff 2001).

So far, Requirements Engineering has received a lot of attention in traditional system design, such as product system design, software system design, and information system design. Traditional requirement engineering approaches are applied during the design time and before implementation (i.e., waterfall model (Royce 1970) in software system domain). Since an IT-enabled service system is viewed as a dynamic collaboration and value co-creation network, traditional requirement approaches are not appropriate to jointly specify requirements with customers and service actors having different backgrounds. Throughout this section, we briefly survey some representative requirement engineering approaches.

The Novel Approaches to Theories Underlying Requirements Engineering (NATURE) (Rolland 1993, Schmitt 1993, Pohl 1993) applies artificial intelligence (AI) techniques to engineer requirements taking into account multi-stakeholders' points of view. Pohl (1993) specifies customer requirements from three dimensions: requirements specification, requirements representation (move from informal to formal), and agreement on specification. The NATURE framework is supported by five approaches mainly based on AI research: 1) a decision-oriented process model denoted by 5-tuple <situation, decisions, arguments, context actions> to represent the fact that developers react contextually according to the situation they are faced with; 2) a knowledge representation model and reasoning techniques on knowledge reuse to enable requirements traceability along the aforementioned three dimensions; 3) a reverse engineering approach to

build the system from specifications at design level and implement it by reusing current software system components such as code, database, algorithm, sub-system, and requirements specification; 4) a similarity-based reasoning model for reusing software specifications; and 5) a set of tools for domain matcher and problem classifier in order to support domain abstractions and assist requirement specifications. This framework has several advantages to deal with those aforementioned challenges:

1- It presents a complete formal specification approach through an iterative process to define and validate requirements (e.g., analysis, trade-off-studies, and prototyping).

2- It deals with the consistency of formal or informal representation languages and discusses the necessity of transforming one representation language to another.

3- In addition, it supports trade-offs and negotiations among various stakeholders with different roles to obtain a common understanding of requirements.

This framework is particularly appropriate for software systems but it fails to discuss socio-technical characteristics of services and lacks approaches to describe how IT drives collaboration of business experts, customers, IT specialists, and developers.

The Win-Win spiral model (Lee 1996) is a requirement engineering approach that takes into account individual stakeholders' requirements (i.e., win conditions) and helps producing reconciled stakeholders' requirements by capturing conflicting requirements and using options to solve them. The Win-Win spiral model permits accommodating requirement changes and takes into consideration various service actors and their points of view. With the Win-Win model (Lee 1996) as their basis, Toffolon and Dakhli (1999) present a requirement model that supports cooperation between customers, architects, developers, and end-users to design software systems. Their core idea is based on the fact that software engineering is governed by a set of contracts among service actors concerned with the software system to be developed and maintained. In this sense, they define software requirements as information sets needed by a stakeholder (e.g., customer, architect, developer, and end-user) to carry out contracts linking all stakeholders, which have interests in software system development. We argue that this model is advantageous to deal with services by exploring cooperation activities to enable the trade-offs and negotiations between various levels of stakeholders and integrating stakeholders at the same level (i.e., architect). However, the Win-Win spiral model lacks formalisms and coordination among stakeholders and does not deal with persistent conflicts between them.

Based on the graphical *i** framework/model (Lee 1996), Donzelli (2004) presents a requirements engineering framework (REF) for the design of software systems, which mainly includes two phases that evolve in a cyclical way: the goal-modelling phase and the organization phase. In the goal-modelling phase, designers refine soft goals into implemented hard goals. A soft goal is related to quality features, context and functions to be provided by the required software system (e.g., I want to improve my English). A soft goal can be decomposed into a set of achievable hard goals, tasks and constraints. During the organization phase, designers use information obtained from the goal-modelling phase to enrich and extend the previous obtained organization model. An organization model is generally composed of agents (i.e., human, software component, organization, etc), tasks (i.e., well-specified activities), resources (i.e., entities used by agents to perform tasks), organizational contexts (i.e., interaction network between agents characterized by collaborative or conflicting), soft goals, hard goals and dependency links describing the relationships among them. Despite the fact that Donzelli's model (2004) provides an effective graphical tool based on the *i**framework to transform high-level goals to low-level goals that are achievable and understandable by both providers and customers and to introduce quantitative and qualitative constraints, it does not guarantee the propagation of changes in requirements in different levels during the service delivery.

In the field of goal-oriented requirements engineering, Baida (2006) proposes an ontology-based requirements engineering method to deal with the gap of what service a customer expects and what service a provider produces. He provides a top-down approach firstly to concretize abstract customer needs expressed in natural language into customer wants that are further specified into clear and implementable customer demands expressed in structured languages. Finally customer demands are translated into customer requirements based on trade-offs and negotiations. In this context, external customer demands, services, and internal resources are described by a structured language and managed by production rules to regulate whether, for example, customer demand X for an expected service described by utilities L, M, ... , N, can be satisfied by appropriate resources described by properties A, B, ... , C. In addition, the trade-offs and negotiation processes in the goal-oriented requirements engineering model address the gap between service customers and providers' different views.

While observing the complexity of approaches dealing with requirement engineering approaches to capture customer needs and specify implementation details, we observe that these approaches, originally developed in the realms of software engineering and system theory, are not ap-

propriate for services defined as networks of social-technical and organizational systems. Even IT-enabled services are considered as a subset of services, they still require requirement-engineering models that reconcile customer relationships, resources, information technologies and business requirements through collaboration and take into account the following service characteristics:

1- IT-driven requirements: in an IT-enabled service system, IT specialists play active roles to stimulate collaboration with business experts, customers, and developers to produce innovative IT-enabled services. IT becomes an important ingredient to add value to services and foster innovation.

2- Customer involvement: customers take part in the whole service lifecycle through service production processes, service delivery processes and service consumption processes. Their inputs are appreciated at different levels and their feedback is handled to measure and improve customer satisfaction. Requirements are not only specified for implementation but they are also used in measuring customer satisfaction during the delivery time.

3- Continuous changes and top-down requirement propagation: services are designed for people and driven by their needs and satisfactions. Requirement engineering cannot be one-time process achieved before implementation or production. It should be revised and updated to take into account customer satisfaction and the service context. All changes introduced at the customer requirements level should be propagated to reach low-level requirements. This implies that low-level requirements should not be included in software code and should be easily managed to update them.

4- Side-way requirements in collaborative environments: establishing requirements in services involves all service actors that collaborate to increase values of co-created services. Requirements should be easy to be decomposed into self-contained modules describing single service features or single service characteristics to be realized by a subgroup of actors belonging to the same discipline. Separation of concerns is an important issue in such complex and collaborative environments.

In the following sections, we will present our requirement framework to identify customer needs for IT-enabled service systems.

4.3 Requirement Engineering Framework

We attempt to build a framework for IT-enabled services to identify, specify and apply requirements engineering through service design and delivery. Establishing requirements should rely on negotiations and trade-offs between customer needs and service provider capabilities. Requirements should also undergo several phases including initial contact, negotiation,

agreement and supervision of expected services. They should involve informal and formal languages as well as natural and technical languages. Finally, they should be subject to changes and their effects should be propagated through the whole service system.

Our methodology to build such a framework consists of making an abstraction of the service system and its resources. This is possible by presenting the service in terms of its service characteristics to refer to the utilities that the consumers received from consuming the service. As discussed in Chapter 3, service characteristics can be expressed as qualitative and quantitative features and constraints easy to be understood by the customer. In addition, defining service system components and their resources makes it possible to refer to them in requirements and add constraints on them. The abstraction of service system components and resources and the service characteristics will be of great value to hide service system complexity and to specify expressive requirements. Since requirements result from negotiations and trade-offs and require formal and informal languages, we adopt terms such as customer needs, customer wants, customer demands, and customer requirements used in the work (Baida 2006) to describe the negotiation process and the transformation of customers needs expressed in unstructured and informal languages (i.e., natural languages) to well structured languages (i.e., SBVR).

Our proposed requirement engineering model can be briefly described as following: if a customer needs an IT-enabled service with a set of expected characteristics L, M, ..., N, then the service provider and the customer establish requirements and co-design a service system described with a set of service characteristics SL, SM, ..., SN supported by an appropriate combination of IT resources or service system components (i.e., resources, technologies, competencies and delivery channels).

The general requirements framework is illustrated in Figure 4.1. As depicted in Figure 4.1, we decompose the requirement engineering model into various levels ranging from high level to low level and including customer needs, customer wants, customer demands, and customer requirements. We also adopt a top-down approach to specify ambiguous, incomplete, and inconsistent customer needs into clear, precise and structured customer requirements.

1- Customers first express their unclear and imprecise needs in natural languages, and then service providers help customers to concretize customer needs into clear customer wants. We adopt the graphical presentation of the *i** model/framework (Rolland 1993, Donzelli 2004) to transform high-level organization requirements into low-level system requirements.

Due to its graphical interface, the *i** model can be easily understood by both providers and customers, which facilitates negotiation and trade-offs.

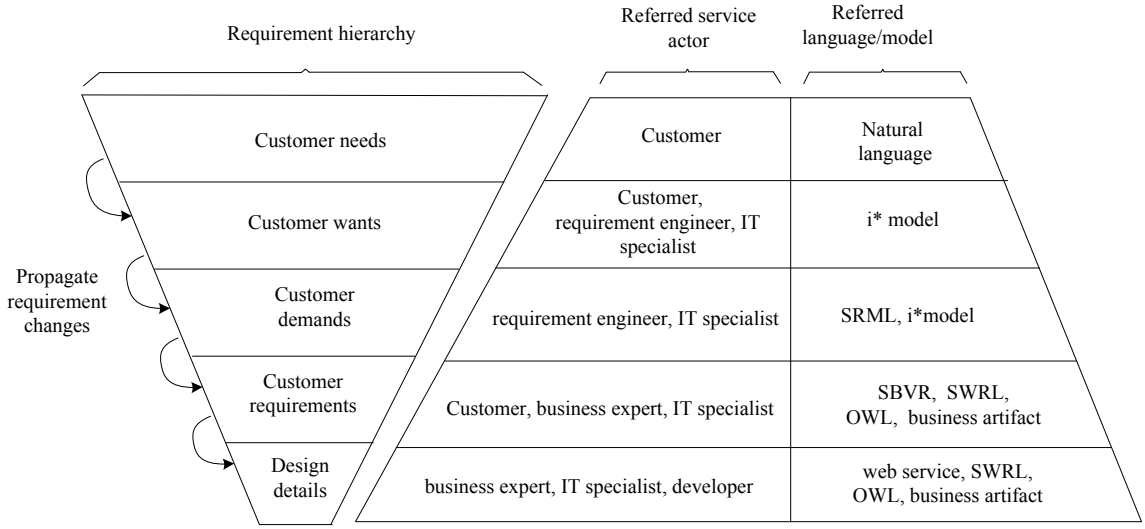


Figure 4.1 Requirements framework to concretize customer requirements

2- Due to the lack of semantics of the *i** graphical model, we introduce a formal model, named SRML based on the Quality of service Modelling Language (QML) to represent customer demands from customer wants by dealing with semantic ambiguities. We will demonstrate in Section 4.4 that the SRML can be used not only at design time to concretize customer demands but also to measure customer satisfaction in the run time due to its power to effectively describe service quality and characteristics.

3- Although SRML is a formal language and is appropriate to describe service characteristics, it is not expressive enough to express business rules. It is also quite technical and can just be understood by technical experts, such as IT-specialists and developers. Since business analysts play active roles in collaboration with customers and various service actors to co-produce IT-enabled services, they need a common business language that can be used by business experts, IT specialists, developers, customers, and other service actors. To this end, we agree on SBVR as the common language to facilitate the transition from SRML to business requirements. As aforementioned, business workers can use SBVR to describe their day-to-day businesses (business vocabularies and business strategy and business rules). Relying on SBVR expressions, IT specialists can interpret SBVR to implement language understandable by computer systems for automation. In this study, apart from describing business vocabularies and rules, SBVR is also proposed to describe structured customer requirements given its internal formal core and external natural-like interface. SBVR-based customer requirements, together with business vocabularies, business strate-

gies, and business rules in SBVR for an enterprise, can be interpreted as implementation languages to be executed by computer systems.

4- Finally, IT specialists have to collaborate with developers to implement the IT-enabled service system. Since IT-enabled service systems involve dynamic and ad hoc collaboration, activity-driven business modelling methods are not appropriate to model such collaboration processes. We argue that IT specialists can deduce from SBVR expressions business artifacts and build business rules expressed by OWL 2 DL axioms (OWL2 2009) and SWRL rules to automate the invocation of appropriate software components. In Section 4.5, we will introduce the mapping from business rules in SBVR into OWL 2 DL axioms and SWRL rules.

The top-down approach is to guarantee that the service is driven by customer needs, their requirements and all underlying service provider's resources are mobilized towards this end. The side-way approach is to ensure collaboration among service actors in the same discipline to specify their requirements in a language understood by them and the propagation of requirement changes.

In the following sections, we articulate the top-down and side-way approach and the mapping from SBVR to SWRL rules.

4.4 Requirement Engineering Model

4.4.1 Setting Requirements on Service Characteristics

As aforementioned, we provide a generic production rule to specify customer requirements by assuming that customer requirements for an expected service described by a set of expected service characteristics can be met by an appropriate composition of internal characteristics satisfying the customer's requirements.

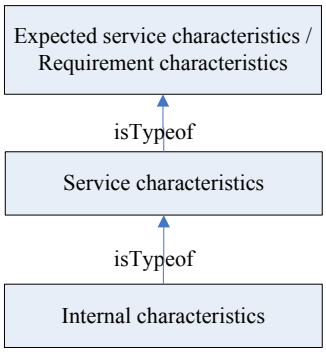


Figure 4.2 Relationships among various characteristics

Figure 4.2 illustrates the relationship between internal characteristics, service characteristics and expected service characteristics. Internal

characteristics refer to the characteristics of IT resources and service system components. Service characteristics denote the expressive and structured customer requirements related to internal characteristics. Customers collaborate with providers to set requirements on service characteristics thereby leading to expected service characteristics.

In order to illustrate the relationship between these three characteristics, we use the example presented in Chapter 3. For instance, there are two internal characteristics: including TeacherCharacteristic and VideoConferenceCharacteristic, as shown below:

TeacherCharacteristic = {Age, Name, Nationality, Title, Major, Availability, workingYears};

VideoConferenceCharacteristic = {response time, delay, connectionFailureRate, connectionBlockRate, ConnectionNumber, internetBandwidth};

Assume that a student needs 'efficiently learning mathematics via online group', which can be described by the service characteristic 'group-oriented e-learning' can be satisfied by these two internal characteristics. As for the performance of this service characteristic, providers and customers can negotiate with one another to set requirements on these two internal characteristics thereby obtaining the expected service characteristics shown below:

TeacherCharacteristics = {Age(>=20&&<=50), Name(any), Nationality(France), Title(at least associate professor), Major(Computer Science), Availability (24h), workingYears(>=3)};

VideoConferenceCharacteristics = {connection response time(<=20s), delay(<=3s), connectionFailureRate(<=5%), connectionBlockRate(<=3%), ConnectionNumber(<=10), internetBandwidth(>=20Mbits/s};

As a result, the initial needs undergo successive refinements to identify precise and implemetable requirements and to propose an e-learning service with the service characteristic 'VideoConference-based e-learning' that guarantees satisfaction. Setting requirements on service characteristics results from negotiation and trade-offs between providers and customers to reach an agreement on the expected service characteristics. The expected characteristics can also be viewed as goals to achieve by mobilizing service system components.

However, since customers are always non-professional and providers and their customers may use different models and languages, an issue is raised about how to enable and facilitate their initial negotiation. We argue that in this initial interaction step, a graphical model is potentially helpful to deal with this issue.

In this sense, in the next sub-section, we articulate how to concretize customer needs with a graphical model understood by both providers and customers based on our proposed generic production rule.

4.4.2 Expressing Customer Wants with Graphical Models

Based on customer needs, the first challenge is to express customer wants with user friendly models to illustrate how customer needs described in natural languages can be refined to identify consistency, complete and unambiguous wants. Since customer wants result from collaboration among service actors and customers, we found that the goal-oriented model presented in the *i** framework (Lee 1996, Donzelli 2004) is quite appropriate to concretize needs into wants and to describe them as expected service characteristics taking into account various service actors' views. It provides a graphical presentation that could easily be understood by both professional business experts and non-professional customers.

As mentioned in the related work of this chapter (see Section 4.2), goals (Donzelli 2004) are composed of hard goals when the achievement criterion is sharply defined (e.g., I need an e-learning service, I need an application to rate teacher, I need a computer, etc.) and soft goals when they are subjective and always related to quality features, contexts and functions to be provided by required software systems (e.g., I need an effective and flexible e-learning service, I need a good computer, etc.). Soft goals can be composed of more precise subordinate soft goals to reduce complexity, until, eventually, reaching a set of well-defined hard goals associated with constraints and quantitative and qualitative requirements.

By considering customer wants as soft-goals representing expected service characteristics, service customers and providers collaborate with each other to transform the soft-goals into a set of implementable hard-goals (internal characteristics) and constraints on the service system components.

The adopted goal-oriented model applied in our context is illustrated in Figure 4.3. The core idea to specify customer wants lies in the fact that a service system is defined as a list of service characteristics and each of which is specified by characteristics of a set of service system components (namely internal characteristics) in terms of our proposed production rule.

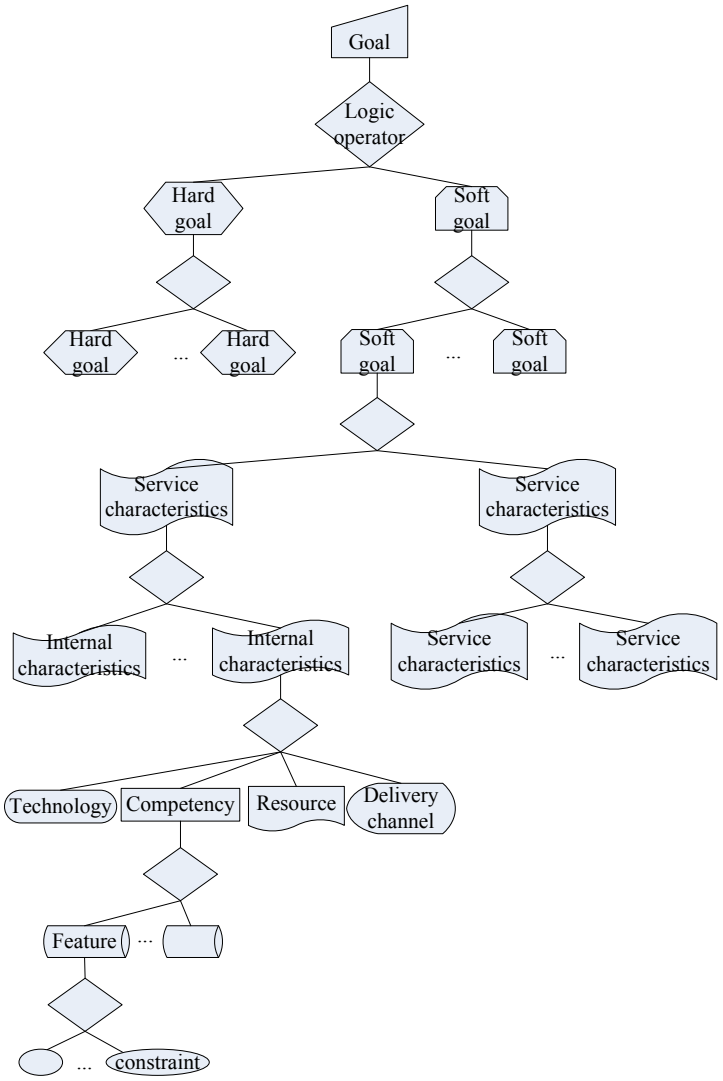


Figure 4.3 Graphical meta-model to specify customer needs

The general process to concretize customer needs into customer wants is as follows:

1- With the negotiations and trade-offs between providers and customers, they at first specify unclear and imprecise customer needs into a set of hard goals and soft goals (service characteristics). Hard goals are quite clear and are thereby easily achieved.

2- Then they specify soft goals into a set of hard goals with internal characteristics and set requirements on them and they finally reach an agreement on them thereby leading to expected service characteristics (namely clear and precise customer wants).

Figure 4.4 shows an e-learning example to illustrate its goal-oriented meta-model and how to specify wants from customer needs.

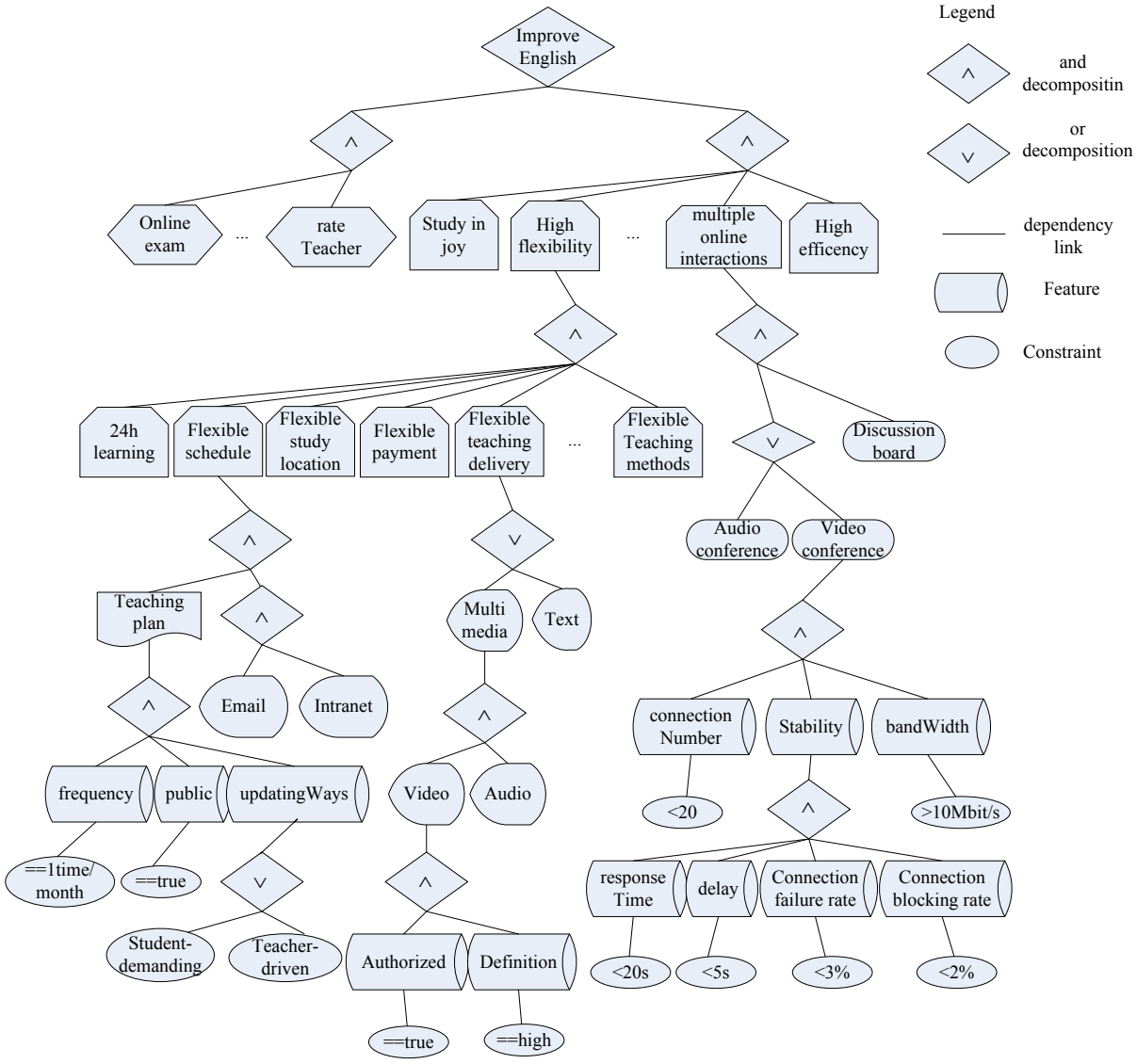


Figure 4.4 An example to illustrate how to specify customer needs with the graphical model

In this example, the provider will identify two possible hard goals to satisfy the required service namely 'online exam' and 'rateTeacher', and some soft goals including 'Study in joy', 'high flexibility', 'flexible online interaction', and 'high efficiency'. The characteristic of high flexibility is further decomposed into various sub-soft goals (sub-characteristics): such as 'flexible schedule', 'flexible studying location', '24h learning' (flexible study time), 'flexible teaching delivery', 'flexible payment', and 'flexible teaching methods'. Each of these soft goals may also be decomposable. Providers and their customers have to negotiate and reach an agreement on the granularity of soft goals or hard goals.

Assuming that they reach an agreement on the granularity of the 'flexible teaching delivery', they can achieve the goal by specifying the characteristics of internal service system components (internal characteristics) in terms of our proposed production rule. As shown in Figure 4.4, the service characteristic 'flexible teaching delivery' can be satisfied by a set of delivery channels (multimedia or text). The multimedia is further specified into audio and video and further described by a set of features and constraints. For example, the audio has two features: *authorised ==true, definition==high*. In other words, if a customer expects the service characteristic 'flexible teaching delivery', the provider can offer the customer a set of internal characteristics to satisfy the service characteristics. They can negotiate with one another to set requirements for the internal characteristics thereby leading to expected service characteristics.

Taking another example, the service characteristic 'multiple online interactions' can be satisfied by a set of technologies: video conference, audio conference, and discussion board. The characteristics of 'the video conference' are: *max connection=20, bandwidth speed>=10Mbits/s, response time <=20s, delay <=5s, connection failure rate <3%, connection blocking rate <=5%.*

In addition, the service characteristic 'flexible schedule' can also be satisfied by internal resource 'teaching plan' and internal delivery channel ('email' or 'intranet'). The resource 'teaching plan' can be specified by the characteristics: *frequency==1times/month, public ==true, updatingWays==students-demanding or teacher-driven*.

In this phase, the *i** graphical model is quite effective to facilitate the trade-offs and negotiations between providers and customers to identify clear and precise customer wants. However, this graphical representation may lead to unexpected designs and implementations for low-level workers (i.e., IT specialists and developers) due to its lack of formal interpretation. As a result, if a customer update his/her needs, the *i**model can not propagate changes to low-levels (e.g., business level, IT-level and implementation level). In order to avoid the semantic ambiguity of requirements at low level service components, Pohl (1993) argues that the final specified customer requirements should be represented in a common agreed formal language to specify customer wants and must be understood by all service actors (i.e., IT specialists, developers, etc). In this sense, we will present a formal language to specify customer wants into customer demands.

4.4.3 Expressing Customer Demands with SRML

Frolund and Koistien (1998) present the Quality Modelling Language (QML) to define QoS properties in software engineering and allow software systems to replace old components with new components as long as the new components satisfy the QoS specifications. In a similar way, we view that well-defined QoS-based customer requirements would help service system designers to update service components, or replace old components with new components as long as the new components satisfy the QoS customer requirements specifications. In addition, QML is also used by developers to design interfaces of software systems.

In this study, we extend the QML by developing the Service Requirement Modelling Language (SRML) to specify customer demands in IT-enabled service systems based on customer wants presented by the *i** graphical model. We highlight the major differences between QML and SRML as follows:

1- QML is mainly used by developers to model the quality of software services in distributed systems. A software service profile is defined as a list of requirements each of which specifies one or more interface entities describing operations parameters, operation results and attributes. Conversely, the SRML is used by business experts, IT specialists and developers to describe requirements on service characteristics and constraints on service system components. Requirements expressions can refer to service system components to set constraints.

2- SRML focuses on service characteristics each of which is described by a set of features. Service characteristics are satisfied if all requirements and constraints on features are satisfied (default is the 'AND' operators among all features). SRML also introduces the disjunction operator ('OR') to specify alternatives among features to ensure that service characteristics are satisfied if one of the alternative features is satisfied.

3- The SRML allows specifying customer demands for IT-enabled service systems using a structured language that specifies quantitative and qualitative constraints. Further, it describes customer demands at design time, can be used in measuring customer satisfaction at delivery time and determines how far the perceived service is different from the expected service. In this sense, it helps to concretize customer demands from customer wants and to ensure the propagation of changes at the characteristics level towards service system components level.

4.4.3.1 Syntax of SRML

The syntactic grammar of the SRML is illustrated in Table 4.1, 4.2, 4.3 and 4.4. A characteristic is composed of various features each of which has a type, a scale (i.e., ordinal, interval, nominal and ratio) unit to measure value, value order, and value sequence (decreasing or increasing). In addition to basic types, the SRML supports the following value types:

1) enum{ n_1 **,…,** n_k } specifies an enumeration domain where "$n_1, \ldots, n_k$" denotes the names of value domain. For example, 'a teacher's title could be assistant professor, associate professor, or full professor', can defined as *enum{assistant professor, associate professor, full professor}*.

2) set{ $n_1, \ldots, n_k$ }: denotes the value set for a feature. For example, 'a student may select more than one lesson at the same time, such as English, French, and Mathematics', denoted as *set{English, French, Mathematics}*.

3) numeric: contains the real numbers that are restricted by a constraint set. For example, 'the working years for a teacher should be more than 3 years', denoted as *workingYears>=3 years*.

4) string: denotes the regular strings, e.g., a student has name 'Jim'.

The attribute 'order' denotes the self-defined value order of value domain. For example, *enum{ excellent, good, moderate, bad} with order {bad< moderate, moderate < good, good < excellent}*.

The attribute 'valueUnit' denotes the unit to measure a value. It may have the following structures: 1) single unit: for example, *3 years, 2 seconds, 5 hours, etc*.; 2) two units: for example, *20 Mbits/second*; 3) probability (i.e., *80*%); and 4) null (i.e., *80*).

Table 4.1 Syntax for characteristic type

characteristicsType::=*InternalCharacteristics*
{ $featureName_1$: $featureType_1$;…; $featureName_k$: $featureType_k$;}
InternalCharacteristics::=*SupportintInputCharacteristics*|
|*CustomerInputCharacteristics*
featureName::=**string** *featureScale*
featureScale::= **ordinal | interval | nominal | ratio**
featureType::= *featureSort* | *featureSort unit*
SupportintInputCharacteristics::=**ResourceCharacteristics|CompetencyCharacteristics |ChannelCharacteristics|TechnologyCharacteristics**
CustomerInputCharacteristics::=**CustomerResourceCharacteristics| CustomerCompetencyCharacteristics |CustomerChannelCharacteristics|CustomerTechnologyCharacteristics**
featureSort ::= **enum{** n_1 **,…,** n_k **}** | *relSem* **enum{** n_1 **,…,** n_k **} with** *order*

| **set{** n_1 **,…,** n_k **}**
| *relSem* **set{** n_1 **,…,** n_k **}**
| *relSem* **set{** n_1 **,…,** n_k **} with** *order*
| *relSem* **numeric**
| **boolean**
| **string**

order **::= order{** n_i<n_j **,…,** n_k<n_m **}**

valueUnit **::=** *valueUnit/valueUnit* | **%** |*valueUnit*| Mbits| **s**| **year**|**hour**|...

relSem **::= decreasing** | **increasing**

Although the 'boolean' type can be indirectly described through 'enum{true, false}', we also indicate the 'boolearn' type in this syntax due to its widely use.

Once the characteristic is defined with the syntax presented in Table 4.1, we can add constraints on its features with the syntax illustrated in Table 4.2. The composition of various characteristics by logical operators ('AND' or 'OR') constructs the service characteristics with the syntax shown in Table 4.3. Similarly, the setting requirements on service characteristics by logical operators ('AND' or 'OR') composes the expected service characteristics with the syntax shown in Table 4.4.

In a constraint set, each constraint has the following attributes: 1) name; 2) constraintValue; and 3) quantitative operator (>, >=, ==, <=, <, and !=). As opposed to QML, our SRML introduces a '*!=*' quantitative operator to denote 'not'. For example '*role != student*' means that the role of feature is not a student.

The core value of QML lies on that it introduces the attribute aspect to denote a statistical characterization of a feature, including means, percentile, variance, and occurrence frequency range of this feature. In our SRML, we also keep this key feature. For example, '*the constraint frequency [5, 8) >20% for a specific feature denotes that we expect 20% of the actual occurrences of this feature to be at least 5 and less than 8*'.

Table 4.2 Syntax for expected characteristics

expectedCharacteristics::=**expectedCharacteristics**
{ *constraint*$_1$ **;***constraintRelationOp constraint*$_2$ **;** *constraintRelationOp* **…;** *constraintRelationOp constraint*$_k$ **;}**
constrain **::=** *featureName constraintOp constraintValue*
| *featureName*{ *aspect*$_1$ **; &&** *aspect*$_2$ **;&&…;&&** *aspect*$_k$ **;}**
constraintRelationOp **::= &&** | **||**
constraintOp::= **==** | **>=** | **<=** | **<** | **>** | **! =**
constraintValue::= *literal valueUnit* | *literal*
literal::= *string* | **{** n_1 **,…,** n_k **}** | *number* | *bool*
valueUnit::= *valueUnit/valueUnit* | **%** |*valueUnit*| Mbits| **s**| **year**|**hour**|...
bool **::= true** | **false**
aspect::= **percentile** *percentNum constraintOp constraintValue*
| **mean** *constraintOp constraintValue*
| **variance** *constraintOp constraintValue*
| **frequency** *freqRange constraintOp number***%**

freqRange **::=** *constraintValue*
| *lRangeLimit constraintValue*, *constraintValue rRangeLimit*
lRangeLimit **::= (** | **[**
rRangeLimit **::=)** | **]**
percentNum **::= 0** | **1** | … | **99** | **100**

Further, the expected value of a constraint for a feature may also be a value range. For example, the available time for a course can be described as a set of constraints '*availableTime>=8:00 && availableTime<=11:30 || availableTime>=13:00 && 'availableTime<=18:00*'. In thise sense, different from QML, we add a value range in our SRML by introducing the constraint relationship operator, where '&&' denotes 'AND' relationship of two constraints and '||' denotes 'OR' relationship between two constraints.

Another key element in QML is 'refined by', which is used to impose a more stringent restriction on the constraints of a feature for a characteristic in order to reuse the characteristic. For example, the teacher refined by a special teacher implies that the special teacher has more stringent restriction on the value of constraints, as shown below:

```
teacher = humanResource expectedCharacteristics{
Age >=20 years && Age <=60 years;
Title >=Associate professor;
Major = ={TOFEL, GRE};
Availability > 0.8;
workingYears >=3;
};
specialTeacher = teacher  refined by
{
Age >=30 years && Age <=50 years;
Title == professor;
Major = ={TOFEL, GRE};
Availability > 0.9;
workingYears >=5;
publishedJournal >=5
};
```

In QML, various characteristics simply have dependencies of 'AND'. However, in real cases, a service characteristic may be specified by the collection of internal characteristics with both 'AND' and 'OR' relationships. For example, the service characteristic 'Group-oriented e-learning' may be satisfied by: '*TeacherCharacteristic && VideoConferenceCharacteristic || AudioConferenceCharacteristics*'. In this sense, as shown in Table 4.3, in SRML, a service characteristic is composed of a set of internal characteristics with dependencies of 'AND' or 'OR', which enriches QML.

Table 4.3 Syntax for service characteristics

serviceCharacteristics **::= serviceCharacteristic** *characterlistisList*
characterlistisList ::= **{** $characteristicsType_1$ **;** *charaOp* $characteristicsType_2$ **; …;** *charaOp* $characteristicsType_k$ **;}**
characteristicsType **::=** *defined in Table 4.1*
charaOp ::= **&&** | **||**

In a service system, there may be large quantities of internal characteristics. A certain service characteristic may be satisfied by some of them, which means that a service characteristic does not refer to all internal characteristics. As aforementioned, setting requirements on various internal characteristics that are referred to in service characteristics will lead to expected service characteristics. The syntax of expected characteristics is shown in Table 4.4. In Table 4.4, x_c denotes a variable that holds characteristic value and '*y'* denotes a variable that holds characteristic type.

Table 4.4 Syntax for expected service characteristics

expectedServiceCharacterisitcs **::= expectedServiceCharacterisitcs** *expectedCharacteristicsList*
expectedCharacteristicsList **::=** { $charaExp_1$; *charaOp* $charaExp_2$; **…;** $charaExp_{k-1}$; *charaOp* $charaExp_k$;}
charaExp **::=** *y expectedCharacteristics*
| x_c **refined by** { $constraint_1$;*constraintRelationOp* $constraint_2$; *constraintRelationOp* **…;** *constraintRelationOp* $constraint_k$;}
charaOp **::= &&** | ||
expectedCharacteristics **::=** *defined in Table 4.2*
constraintRelationOp **::= &&** | ||

4.4.3.2 Example to describe Customer Demands with SRML

The syntax of SRML is quite abstract and may be difficult to understand. In this sense, we present how to use SRML to represent customer wants denoted by the *i** model via the example shown in Figure 4.4. Before interpreting the example, we first discuss the mapping from the *i** model to SRML, as shown in Table 4.5.

Table 4.5 The mapping from *i**model to SRML

i* model	SRML
hard goal: e.g., *issue a student card*	expectedCharacteristics: e.g., *Type studentCardType= ResourceCharacteristics{* *isValid: boolean ;* *limitsOfAuthority: increasing enum{school-level, city-level, national};* *};* *studentCard = studentCardType expectedCharacteristics{* *isValid== true;* *limitsOfAuthority >= city-level;* *};*
soft goal: e.g., *group-oriented e-learning*	serviceCharacteristics: e.g., *group-oriented e-learning=serviceCharacteristics{* *Type VideoConferenceCharacteristics=Technology Characteristics{* *};&&* *Type TeacherCharacteristic= ResourceCharacteristics{* *};* *};*

goal: e.g., *efficiently study mathematics through online group*	expectedServiceCharacteristics: e.g., *group-oriented e-learning=serviceCharacteristics{ VideoConferenceCharacteristics{…};&& TeacherCharacteristic{…} };*
InternalCharacteristics (i.e., resource, competency, technology, and delivery channel): e.g., *a teacher, computer,Video conference technology, etc.*	characteristicsType: *e.g.,* *teacher = humanResource expectedCharacteristics {* *Age >=20 years && Age <=60 years;* *Title >=Associate professor;* *Major = ={TOFEL, GRE};* *Availability > 0.8;* *workingYears >=3years;* *};*
Service characteristics: e.g., *group-oriented e-learning*	serviceCharacteristics:
feature: e.g., *workingYears*	featureName: e.g., *workingYears*
constraint: e.g., *workingYears >=3 years*	constraint: e.g., *workingYears>=3 years*
logic operator ('AND' or 'OR'):	constraintRelationOp or charaOp:

Now we return to the example shown in Figure 4.4. Figure 4.4 lists the specification of three service characteristics: including flexible teaching delivery, flexible schedule, and multiple online interactions. We add our aforementioned service characteristic 'group-oriented e-learning' into the service characteristic list. We respectively illustrate the SRML expressions for these four service characteristics as examples. Table 4.6 lists the excerpt of internal characteristics to achieve these four service characteristics and the complete list of them can refer to Appendix A.

Table 4.6 Excerpt of characteristic type

```
// characteristic type or internal characteristics
Type highLevelTeacherType= ResourceCharacteristics{
Age : numeric years;
Name: string;
Title: increasing enum{teaching assistant, associate professor, professor, director} with order { teaching assistant< associate professor, associate professor< professor, professor< director};
Major: string;
Availability: numeric;
workingYears : numeric years;
};
Type VideoConferenceType = TechnologyCharacteristics{
response time: numeric s;
delay: numeric s;
connectionFailureRate: numeric %;
connectionBlockRate: numeric %;
ConnectionNumber: numeric ;
internetBandwidth: numeric Mbits/s;
};
Type AudioType= ChannelCharacteristics{
Authorized: boolean;
```

```
Reliability: numeric;
soundQuanlity: increasing enum{low, average, high} with order { low<average<high};
};
Type ScolarCourseType = ResourceCharacteristics{
{
Reliability: numeric;
Format: set{video, audio, text};
Price: numeric;
accessAuthority: boolearn;
updateAuthority: boolearn;
downloadAuthority: boolearn;
downloadCreditUnit: numeric /100Mbits;
};
Type SelfTestVideoGameType = ResourceCharacteristics{
Price: numeric euros;
GameQuality: increasing enum{High, average, low} with order {high>average>low};
Enjoyment: increasing enum{High, average, low} with order {high>average>low};
Diversity: increasing enum{High, average, low} with order {high>average>low};
};
Type OnlineRobotForChattingType = TechnologyCharacteristics{
intelligence: increasing enum{High, average, low} with order {high>average>low};
chatWays: set{text, sound};
};
…
};
```

The SRML expressions for these four service characteristics are illustrated in Table 4.7. Table 4.7 also lists three additional service characteristics. For example, the Table illustrates 'Large Quantities of Reliable Teaching Resource', 'Study in Entertainment', and 'Support Social learning', respectively.

Table 4.7 Example of service characteristics

```
// Service characteristics
Group-oriented e-learning = serviceCharacteristics
{
TeacherType{};&&
VideoConferenceType;{}||
AudioConferenceType{};
};
Flexible Teaching Delivery = serviceCharacteristics
{
VideoType();||
AudioType():&&
TextType();
};
Flexible Schedule=serviceCharacteristics
{
TeachingPlanType{};&&
EmailType{};||
IntranetType{};
};
Multiple Online Interactions = serviceCharacteristics
{
VideoConferenceType{};||
Audio ConferenceType{};&&
DiscussionBoardType{};
};
Large Quantities of Reliable Teaching Resource= serviceCharacteristics
```

```
{
OpenCourseType{};&&
SchlarCourseType{};
};
Study in entertainment= serviceCharacteristics{
StudyViaVideoGameType {};&&
SelfTestVideoGameType{};&&
OnlineRobotForChattingType {};
};
Support social learning{
SocialLearningToolType{};
};
```

Based on these internal service characteristics in Table 4.7, customers and providers can then set requirements on them to obtain expected service characteristics in SRML as shown in Table 4.8. Indeed, in Table 4.8, we present some expected characteristics and expected service characteristics as examples.

In this example (see Tables 4.6. 4.7, and 4.8), we successively represent the graph representation in the *i**model in Figure 4.4 into SRML expression. This example also proves the value of our proposed characteristics view helping to transform unclear, unstructured, and imprecise customer needs into clear, structured, formal SRML expressions to deal with service perception and service expectation via negotiations and trade-offs between customers and providers.

Table 4.8 Example of expected characteristics and expected service characteristics

```
// examples of expected characteristics
SpecialTeacher = TeacherType  refined by
{
Age >=30 years && Age <=50 years;
Title == professor;
Major = ={TOFEL, GRE};
Availability > 0.9;
workingYears >=5;
publishedJournal >=5
};
VideoConference = VideoConferenceType  expectedCharacteristics{
response time<= 20m;
delay<=5m;
connectionFailureRate <=3 %;
connectionBlockRate<=3 %;
Connection<=20;
internetBandwidth>=20 Mbits/s;
};
AudioConference = AudioConferenceType  expectedCharacteristics{
response time<= 10m;
delay<=3m;
connectionFailureRate <=3 %;
connectionBlockRate<=3 %;
Connection<=40;
internetBandwidth>=10 Mbits/s;
};
TeachingPlan= TeachingPlanType  expectedCharacteristics {
Plan period>=1times/month;
```

```
Public==true;
UpdatingWays==set{student-demanding, teaching-driven};
};
Video= VideoType expectedCharacteristics{
Definition>=average;
SoundQuality:>=average;
Authorized ==true;
Reliability>=0.9;
};
 Audio=Audio Type expectedCharacteristics{
Authorized==true;
Reliability>=0.9;
soundQuanlity>=average;
};
Type TextType= ChannelCharacteristics{
Authorized==true
Reliability>=0.9;
};
// expected  service characteristics
Group-oriented e-learning = expectedServiceCharacteristics
{
SpecialTeacher{};&&
VideoConference;{}||
AudioConference{};
};
Flexible Teaching Delivery = expectedServiceCharacteristics
{
Video{…};||
Audio{…};&&
Text{};
};
…
```

Compared to the *i**model, SRML is a formal model that adds semantic interpretations and illustrates both quantitative and qualitative constraints to describe customer demands at design time. Developpers can develop service system interfaces in terms of SRML expressions.

Based on customer demands, the next step is to design business services to satisfy customer demands. However, an IT-enabled service system refers to large quantities of collaboration where customers take part in the whole process and input their knowledge to influence decisions to determine the appropriate strategy to specify requirements and produce IT-enabled services. Therefore, a predefined business process cannot accommodate its dynamics. We argue that the rule-based approach is appropriate to deal with this issue. The general idea behind a rule-based approach lies in that decisions are made based on business rules. Current business rule languages such as natural language, Event-context-Action (ECA) (Tang et al. 2011), DL, Frame logic (FL) (Durand 2010), and SWRL, are either too abstract or too technical to be used by various service actors (Business experts, IT specialists, customers, etc) in service systems to design or read business rules. In this sense, it requires a common formal rule-based language that is close to natural language and also close to a rule-based im-

plementation language (e.g., FL, and SWRL) to connect customer, business experts, IT specialist, and developer for collaboration. SRML is thus not appropriate. In the next part, we present the desired common formal language and discuss how it is used to integrate customers, business experts, IT specialists, and developers in collaboration for automation.

4.4.3.3 Expressing Customer Requirements with SBVR

SBVR proposed by the OMG is recommended as the very common formal rule-based language to connect customers, business experts, IT specialists, and developers for collaboration. Compared to other kinds of rule languages (either formal or informal), SBVR has the following advantages:

1- It is a formal business rule language with a natural language interface, which implies that either users or business analysts can express their business logics with a natural-like language.

2- Compared to other semantic ontology language or business rules languages (i.e., DL, FL, and SWRL), SBVR provides natural-like interfaces to define rich business vocabularies and rules and also allows their integration.

3- SBVR separates symbols from their meanings, which indicates that it supports multilingual development.

4- It is possible to transform SBVR to other implementation rule languages. For example, researchers have achieved the mapping from SBVR to OWL DL (Ceravolo et al. 2007), and SBVR to FL (Durand 2010).

5- Besides OMG, some organizations or enterprises have adopted SBVR as a standard to describe their business rules, e.g., the Digital Business Ecosystem (DBE) (an integrated project of the European Commission Framework Programme). In this sense, SBVR provides a formal and common interface.

SBVR distinguishes the meaning of rules and vocabularies from their expression in textual, diagrammatic, or other forms. The meaning of rules and vocabulary is captured as instances of the SBVR meta-model. Tool developers can express rules and vocabulary in any way they want, but the standard itself is defined in “Structured English (SE)”, which is a very limited subset of the English language. The technique of converting logical statements to a restricted human language is similar to Controlled English (CE) explored in Fuchs and Schwitter in 1996 and Bernstein and Kaufmann in 2006. The description of the SBVR Structured English (SBVR SE) is divided into several subclasses and more details can refer to SBVR 1.0 documentation (SBVR 2008).

1- Expressions in SBVR structure English: refer to sentences that conform to SBVR grammar and use defined SBVR vocabulary to describe rules.

2- Describing a vocabulary: a vocabulary is described in a document having glossary-like entries for concepts having representations in the vocabulary.

3- Vocabulary entries: each vocabulary is composed of concept entries. A concept entry starts with a primary representation which is either a designation (name or term) or a fact type form for the concept.

4- Specifying a rule set: The introduction to a rule set includes the rule set's name and other additional information, such as the scope and purpose of the rules, and the used vocabulary in the rules, the notation of the rule set.

5- Guidance entries: denotes each entry in a rule set if an element of guidance, which may refer to any one of the following statements, including an operative business rule statement, a structural business rule statement, a statement of advices of permission, and a statement of advices of possibility.

Generally, there are two ways to express definitions and business rules: one is through statements (e.g., SBVR and Attempto Controlled English (ACE) (Fuchs and Schwitter 1996), and the other is diagrammatic illustration (e.g., ORM Notation). The major difference between SBVR and ACE lies in the fact that the former is a formal structured natural-like language while the later is not. A diagrammatic illustration is useful to see how concepts are related, but it does not prove very intuitive for people to understand. The more practical and intuitive way is to use natural language. There are many ways to describe vocabulary and rules with SBVR, for example, combine with common English words and structures to express them or use a small number of English structures and common words. The former method is helpful to express semantics, but can just partially map logical formulation and as a result, is difficult to implement (Durand 2010). The latter method sacrifices a little bit of expressiveness for logical formulation and computing. As per these arguments, in this study, we adopt the latter method, namely SBVR SE, to describe business vocabulary and rules. In the remainder of this thesis, unless it is indicated otherwise, we use SBVR to replace SBVR SE for the sake of simplicity.

Figure 4.5 illustrates how SBVR is used to facilitate collaboration among customers, business experts, IT specialists, and developers. The general idea is that on the business level, specified customer requirements are described in SBVR that is used to validate customer requirements between customers and providers, while on the technical level, SBVR vo-

cabularies are transformed into OWL to enrich the service system ontology and SBVR rules are mapped into SWRL rules to enable the reasoning on business facts and to execute business logics. Further, IT specialists generate business artifacts from SBVR rules. As aforementioned, business artifacts are self-contained business records that include attributes, states and life cycles that reflect the changes to these states. The artifact concept not only describes a business entity, but also encompasses knowledge about what to process without explaining how to do it. On the implementation level, web services or SaaS developed by software developers are invoked by SWRL rules to update business artifacts.

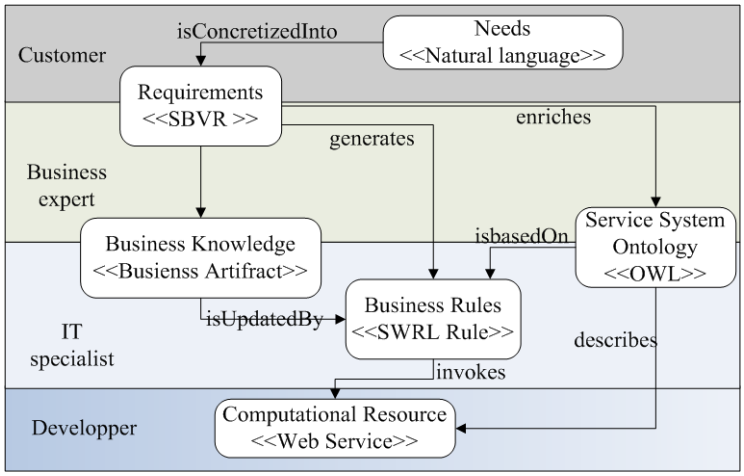


Figure 4.5 A common model to connect customers, business experts, IT specialists and developers

In the following paragraphs, we articulate how SBVR is used as a common formal model to connect various service actors from different levels.

Customer requirements: business experts have to restructure identified customer demands in SRML into customer requirements in SBVR. Since both SBVR and SRML are formal languages, it is possible to implement their mapping.

Herein, we discuss the mapping from SRML into SBVR, as shown in Table 4.9.

Table 4.9 The mapping from SRML to SBVR

SRML	SBVR
characteristicsType *e.g., humanResource*	Object *e.g., humanResource*
expectedCharacteristics *e.g., student, Jim*	Object or individual *e.g., Teacher, Jim*
Feature *e.g., age, title*	Fact type *e.g., has age, has title*
valueUnit *e.g., year*	Object *e.g, year*

ConstraintOp *e.g., > 10*	Quantification *e.g., more than 10*
constraintRelatioshipOp *e.g., age >18 && age <60*	Logical formulation *e.g., the age is more that 18 and less that 60*
Feature type *e.g., titile: enum{teaching assistant, associate professor, professor, director}*	Logical formulation *e.g., The tile is teaching assistant or associate professor, or professor or director.*
set{ n_1 , ..., n_k } e.g., *set{video, audio, text}*	Logical formulation 'and' *e.g., video and audio and text*
charaOp *e.g., teacher && videoConfrence*	Logical formulation *e.g., teacher and videoconference*
freqRange *e.g., [3,5)*	Quantification *e.g., At least 4 and less than 5*
enum{ n_1 , ..., n_k } e.g., *enum{assistant professor,associate professor, professor, director}*	Logical formulation 'or' e.g., assistant professor, associate professor, professor or director
Service characteristics and expected service characteristics	SBVR expressions

As shown in Table 4.10, we present an example to map SRML to SBVR.

Table 4.10 The example of the mapping from SRML to SBVR

SRML customer demands:

Type TeacherType = ResourceCharacteristics{
Age : numeric years;
Name: string;
Title: increasing enum{teaching assistant, associate professor, professor, director} with order { teaching assistant< associate professor, associate professor< professor, professor< director};
Major: string;
Availability: numeric;
workingYears : numeric;
};
EnglishTeacher =TeacherType expectedcharacteristics{
Age >=20 years && Age <=60 years;
Title >=Associate professor;
Major = ={TOEFL, GRE};
Availability > 0.8;
workingYears >=3;
};

SBVR customer requirements from SRML:

The teacher is a resource characteristics type and it has age, name, major, availability and title that can be teaching assistant, associate professor, professor, or director. The title order from low level to high level is respectively teaching assistant, associate professor, professor, or director. It also has working years. The English teacher is an instance of Teacher. The age of this teacher is at least 20 and at most 60. His tile is at least an associate professor; this teacher has major of TOEFL or GRE. The availability of this teach is bigger than 0.8 and the teacher has worked for more than 3 yeas.

OWL 2 ontology: although SBVR provides a formal way to capture specifications and knowledge in natural-like language, SBVR is not an implementation language and cannot be understood by software systems to enable reasoning based on an established knowledge base (i.e., ontology repository). In this sense, it is necessary to map SBVR to implementation

rules (i.e., OWL and SWRL). Additionally, the use of ontology in service systems has the following two benefits:

1-Ontology (Mora et al. 2011, Izumi et al. 2009) provides common vocabulary and terminologies facilitating interactions and collaboration among various stakeholders.

2- Additionally, an ontology is well suited to be exploited in service systems engineering (Kattenstroth 2007, Lienhard and Künzi 2005) to automatically or semi-automatically execute business logics through rule-based approaches.

In this chapter, we adopt OWL 2 DL to describe the service system ontology since it is a formal language proposed by the World Wide Consortium (W3C) to represent knowledge in the Semantic Web, in which data and knowledge are allowed to be shared and reused across applications, enterprises, and community boundaries. Compared to other ontology language, such as OWL DL, RDF, and XML, OWL 2 is more powerful to express complex business knowledge. For example, OWL DL cannot express the business knowledge '*The teacher of a student's classmates is the student's teacher*'. However, this rule is very easy to represent by OWL 2 DL with its new feature property chain, namely *SubPropertyOf(ObjectPropertyChain(:hasClassmates :hasTeacher) :hasTeacher))*

OWL is generally composed of classes, instances, object properties, and data properties. Both OWL DL and OWL 2 DL are based on DL that represents the knowledge of an application domain in a structured and formal way. DL principally models concepts, roles, individuals, and their relationships, which makes OWL DL and OWL 2 DL appropriate to represent structured knowledge in terms of classes and relationships. Compared to OWL DL, OWL 2 DL provides some new functionalities to improve its expressiveness, such as keys, property chains, richer data types, data ranges, and qualified cardinality restrictions. More details can be found in OWL2 2009.

Broadly speaking, in SBVR-based customer requirements, concepts can be described by OWL 2 DL class, instances can be denoted by OWL 2 DL individuals, relationships between concepts can be expressed by OWL Object Property, relationships between concepts and instances can be described by OWL 2 DL Data Property. In addition, some complex relationships between concepts can also be denoted by OWL 2 DL axioms. For example, the OWL 2 DL axiom can express the union of several concepts. Taking another example, the OWL axiom can describe the disjointness of two concepts, e.g., Male and Female. However, OWL 2 DL axioms are not expressive enough to express complex facts and rules (always represented

as complex cyclic rules) in the business domain. For example, the OWL 2 DL axiom cannot express the following rule:

> It is necessary that a BookBorrow is open and the BookBorrow is not an international borrow then the borrowed book of this borrow is owned by the local area of the pick-up branch of the BookBorrow.

However, this rule is easily described by business rule languages (e.g., SWRL and FL). In the later of this section, we introduce SWRL rules to achieve the mapping.

SWRL rules: compared to OWL 2 axioms, logic rule languages (e.g., FL, DL, and SWRL) are more powerful to improve complex business knowledge. We herein adopt SWRL to describe SBVR rules based on OWL 2 DL. Another alternative is FL. FL provides powerful rules and some salient features, for example non-monotonic inheritance. Researchers (Kattenstroth 2007, Rempel et al. 2010) have proposed to use OWL to describe structured knowledge and FL rules to express constraints and rules.

However, compared to FL, the horn-like SWRL has the following two strengths. Firstly, it is easier to understand, read, and write, and secondly SWRL rules are closely integrated with OWL ontology to extend its expressiveness to express complex business rules.

Generally, a SWRL rule has the following human-readable syntax:

$$\text{Precondition} \rightarrow \text{Effect}$$

where both *Precondition* and *Effect* are conjunctions of atoms written $A_1 \wedge \ldots \wedge A_n$. Atoms A_i can be of the form C(x), P(x,y), sameAs(x,y) differentFrom(x,y), or BuiltIn(x,y,...) (i.e., swrlb:equal(?data, ?Saturday)) where C is an OWL class, P is an OWL property (either object property or data property), BuiltIn is a built-in function, x and y are either variables, OWL individuals or OWL data values (i.e., 10), as appropriate. If x or y is a variable, then it is indicated using the standard convention of prefixing them with a question mark (i.e., ?x). For example,

$$\text{Student}(?\text{student}) \wedge \text{hasClassmates}(?\text{student},?\text{b}) \wedge \text{hasTeacher}(?\text{b},?\text{c}) \rightarrow \text{hasTeacher}(?\text{student},?\text{c})$$

In order to design the desired service system, we introduce ontology and business rules into service systems design and view that clearly identified customer requirements in SBVR can enrich a service system's ontology and business rules base. Since both OWL 2 DL expressions and SWRL rules are machine-readable and can be shared across applications, enterprises, and community boundaries, the mapping from SBVR to the integration of OWL and SWRL cannot only express business vocabularies and business rules, but can also help to enable the reasoning based on the captured specification and knowledge for automated service systems.

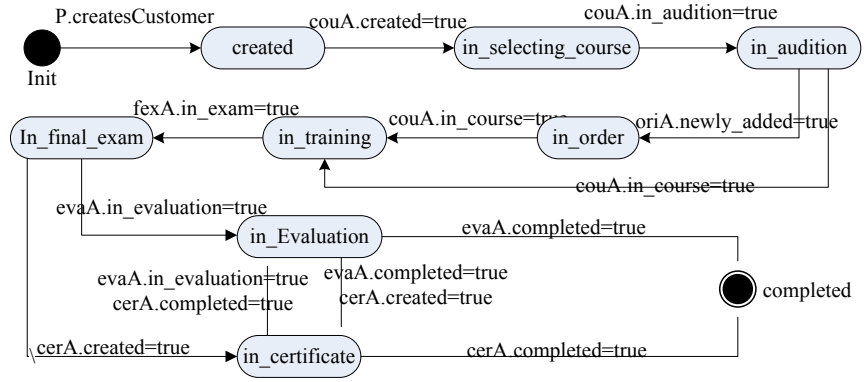
Business artifact: as aforementioned, the exchanged explicit business knowledge in a service system is represented as business artifacts. The general solution relies on that all the objects from SBVR rules can be

viewed as business artifacts and its features or constraints can be viewed as attributes. For example, '*it is necessary that a student has an account if he downloads online educational resource*'. In this SBVR rule, we can identify the following business artifacts: student, account and online educational resource. However, in a real case, the granularity of an artifact is quite important. If it is too large, it may lose some levels or aspects of information to describe business logics. On the contrary, it will refer to too many details thereby leading to an increase of complexity and a reduction of efficiency. Further, due to the limited expressive power of the collection of OWL 2 and SWRL, some SBVR expressions cannot be denoted by them (for example, the negation of properties). The manual work of business people is needed to identify the granularity of business artifacts and related business rules.

Herein, as shown in Table 4.11, we present a customer artifact as an example. In Chapter 5, we will articulate how to use artifact to represent explicit knowledge and how to model collaboration processes with a data-driven method.

Table 4.11 Example of a customer artifact

Customer artifact:
Attributes: CustID, CustCredit, CustType
States: init, created, in_selecting_course, in_audition, in_order, in_training, in_final_exam, in_evaluation, in_certificate, completed
Lifecycle:



Legend:

P: provider	oriA: order item artifact
couA: course artifact	evaA: evaluation artifact
fexA: final exam artifact	cerA: certificate artifact

Web service: service systems ontology is used to represent computational resources (e.g., web services) and SWRL rules are adopted in terms of service systems ontology to invoke these computational resources to update business artifacts that enable collaboration between various service actors. For example, as shown in Table 4.12, the 'service' is used to estimate the customer's discount.

Table 4.12 Business rule and web service examples

Service : EstimateDiscount (discount)
Rule 1:
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order), hasState(?Var_Order, "ready_to_pay"), hasCustCredit(?Var_StudentArt, ?Var_CustCredit), greaterThanOrEqual(?Var_CustCredit, 2500)
->
hasDiscount(?Var_Order, 0.10)
Rule 2:
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order), hasState(?Var_Order, "ready_to_pay"),hasCustCredit(?Var_StudentArt, ?Var_CustCredit), greaterThanOrEqual(?Var_CustCredit, 1500), lessT-han(?Var_CustCredit,2500)
->
hasDiscount(?Var_Order, 0.06)
...

In this section, we proposed a collaboration-based requirements framework (See Figure 4.1) mainly composed of four spiral phases: 1) specify customer needs into customer wants with a graphical *i**model; 2) concretize customer wants into customer demands in formal language SRML; 3) restructure customer demands into customer requirements in SBVR and 4) generate service system ontology (OWL 2 DL), business artifacts from SBVR rules and map SBVR rules into SWRL rules. We also briefly discuss the mapping from the *i**model to SRML, and SRML to SBVR. In the next section, we discuss the mapping from SBVR to OWL 2 DL and SWRL. We argue that our proposed requirements model has the following characteristics:

1- Supports both the top-down (concretizing customer needs into customer requirements) and side-way approach (propose to use SBVR as a common structured language to make various service actors collaborate with one another);

2- Helps to propagate requirements changes from high level to low level.

3- The SRML model is quite appropriate to specify customer demands, but also to enable QoS-based design, trade-offs and negotiation between service providers and customers and to measure customer satisfaction.

4- Facilitates the collaboration of various service actors and takes into account the software service level as well.

5- Appropriate to IT-enabled service systems where IT-specialists play an active role in connecting customers, business experts, and developers to design innovative and customized IT-enabled services.

6- Deals with the gap between service perception and service expectation for customer satisfaction that may lead to more value to both providers (achieving customers' loyalty) and customers (achieving satisfaction).

In the next section, we articulate the mapping from SBVR to OWL 2 DL ontology and SWRL rules.

4.5 Mapping from SBVR to OWL 2 DL and SWRL

As mentioned before, although SBVR provides an effective way to capture specifications and knowledge in natural language and represent them in formal logic, SBVR is not an implementation language and cannot enable reasoning based on the captured specification and the knowledge base. In this sense, it requires to map SBVR expressions into implementation rules that can be understood by machines to realize the automatic reasoning for decision making.

Some researchers have described the feasibility of the transformation from SBVR to formal logical formulations, such as FL, REWERSE Rule Markup Language (R2ML) (Demuth and Liebau 2007, R2ML 2008), and SWRL. Durand (2010) theoretically explores the mapping between SBVR and FL and argues that FL can be partially mapped to FL. Demuth and Liebau (2007) discuss the mapping from SBVR to R2ML and Ceravolo et al. (2007) preliminarily explore the mapping from SBVR to OWL DL and SWRL rules. However, so far no work explores in detail the mapping from SBVR and OWL 2 DL and SWRL.

As mentioned before, in this thesis, we adopt OWL 2 to develop service system ontology and SWRL to describe implementation business rules. More exactly, we adopt DL-Safe SWRL rule to develop business rule based on OWL 2 ontology for the sake of decidability (the problem of OWL reasoners are guaranteed to terminate when classifying an OWL ontology, but the inference with SWRL rules is not). Parsia et al. (2005) argue that the expressiveness of horn-like SWRL rules (built on OWL DL and shares the formal semantics of OWL DL) come at the expense of decidability and DL-safe SWRL rule are a decidable part of SWRL rules. In this sense, we adopt DL-safe SWRL rules.

DL-Safe SWRL rule is a decidable part of SWRL rules and means that only explicitly named individuals are bound to the variables in the rule (SWRL 2011). In other words, in DL-Safe SWRL rules, every variable appears in a non-DL atom in the rule condition. For example,

$$O(\text{student}) \wedge O(b) \wedge O(c) \wedge \text{hasClassmates}(\text{student},b) \wedge \text{hasTeacher}(b,c) \rightarrow \text{hasTeacher}(\text{student},c)$$

where "O" is not a concept from the DL knowledge base but a named individual. In order to ensure the decidability for SWRL rules, a constraint (also called DL-safety constraint) (Haase et al. 2006) has to be achieved as "*variables in SWRL rules bind only to explicitly named individuals in your ontology*".

In the remainder of this thesis, unless otherwise indicated, we use the terms SWRL rules to replace DL-safe SWRL rules for simplicity's sake.

Further, based on the work by Motik et al. in 2009 (Motik et al. 2009), the combination of OWL 2, DL-safe SWRL rules and a Description Graph can ulteriorly improve the expressiveness of OWL 2 and DL-safe SWRL thereby being potential to express complex knowledge (e.g., circular knowledge structure rather than tree knowledge structure). In this Section, we focus on the mapping from SBVR to OWL 2 DL and DL-safe SWRL rules. In our future work, we will explore the map from SBVR to the combination of OWL 2.0, Description Graph, and DL-safe SWRL rules for improving expressiveness.

Finally, when we develop SWRL rules in Protégé 4 (Protégé 4 2011), the symbol '∧' is replaced by ',', and the symbol '→' is replaced by '->'. In the remainder of this thesis, we follow this syntax. For example,

Student(?a), hasAge(?a,?b),greaterThan(?b, 17)-> Adult(?a)

In the remainder of this section, we explore how to map SBVR concepts to OWL 2 ontology and SBVR rules to SWRL rules.

4.5.1 Theoretical Aspects

This part focuses on theoretical aspects in mapping from SBVR expressions to OWL 2 DL ontology and SWRL rules. In order to better demonstrate their mapping, we present the following two procedures:

1- We list each of the main elements for SBVR and at the same time if possible, we identify its equivalent from OWL 2 (including OWL 2 ontology and OWL 2 DL axioms) and then list them in the same table in order to compare them.

2- If OWL 2 DL-axioms cannot achieve the mapping, we will turn to SWRL for help and look for appropriate mapping.

In addition, we apply functional-style syntax to describe OWL 2 axioms and another readable syntax, Manchester Syntax. More details about these two syntaxes can be found in OWL2 2009.

SBVR have wild-used marks to differentiate a concept from an instance. In SBVR, terms are defined using lower case letters and are usually represented underlined, for example, Service System, Customer, and Resource. Instances are always represented with double underlines, for example, *Jim Green*, and *25*. In the remainder of this thesis, we use these marks when writing an SBVR expression.

4.5.1.1 Object

In SBVR, a term is used to represent a noun concept. In OWL 2, the equivalent is class.

Noun	Class
Teacher	Teacher

4.5.1.2 Individual

Individuals are instances of classes. In SBVR, 'name' is used to denote an individual, while in OWL 2, the equivalent concept is 'individual'. For example, assume the concept is a person.

name	ClassAssertion(:name: noun)
e.g., Jim Green	e.g., ClassAssertion(:Jim Green :Person)

4.5.1.3 Predicate

Predicate denotes the relationship between two concepts. In SBVR, the predicate is denoted as a verb, while in OWL 2, the equivalents are either object property or data property.

verb	ObjectSomeValuesFrom, ObjectAllValueFrom(), DataSomeValuesFrom() or DataAllValuesFrom()

4.5.1.4 Facttype

In SBVR, three kinds of fact types are described: including 1) unary fact type; 2) binary fact type; and 3) n-ary fact type.

The unary fact type also has two categories: 1) express property (e.g., student is industrious), and 2) denote activity (i.e., student graduates).

Binary fact type describes the relationship between two classes. If the verb is 'is' or 'are', then 'noun1 is noun2' is translated into "SubClassOf(:noun1 :noun2)".

Assume the n-ary fact type has the form of 'noun 1 verb1 noun2 verb2 …verbk-1 noun k' in SBVR. Generally speaking, a n-ary fact type is composed of several unary fact types and binary fact types. For example, a student works in school during schooltime. This fact may be composed of the following two binary fact types: 1) Student **WorksIn** School, and 2) student **worksDuring** SchoolTime.

Taking another example, 'Student works together with teacher in school located in city'. This example may be composed of the following three binary fact types: 1) Student **worksTogetherWith** Teacher, 2) Student **worksIn** School, and 3) School **locatesIn** City.

In conclusion, since the structure of the n-ary fact type is of varable, it is impossible to work out a general rule to represent the mapping. However, it is sure that a n-ary fact type can be divided into a set of unary fact types and binary fact types.

4.5.1.5 Quantification

The keyword 'some' is not easily translatable. Its translation depends on its meaning. Broadly speaking 'some' has two kinds of interpretations: one is interpreted as existential quantification and the other has the same meanings of 'at least one'.

Quantifiers in SBVR	Mapping to OWL 2
Each	SubClassOf
At least one	objectMinCardinality or dataMinCardinality
At least n	objectMinCardinality or dataMinCardinality
At most one	ObjectMaxCardinality or dataMaxCardinality
At most n	ObjectMaxCardinality or dataMaxCardinality
Exactly one	FunctionalDataProperty or FunctionalObjectProperty
Exactly n	ObjectExactCardinality or dataExactCardinality
At least n and at most m	objectMinCardinality and ObjectMaxCardinality or dataMaxCardinality and dataMinCardinality
More than one	objectMinCardinality or dataMinCardinality

4.5.1.6 Logical Formulations

Logical formulations in SBVR refer to semantic formulations that formulate a proposition. We list logical formulations in SBVR below:

Logical operators in SBVR	Mapped to OWL 2
It is not the case that p	ObjectCompelementOf (:p)
p and q	ObjectIntersectionOf(:p :q)
p or q	ObjectUnionOf(:p :q)
m is p or q but not both	ObjectIntersectionOf(ObjectUnionOf(:p :q) ObjectComplementOf(ObjectIntersectionOf(:p :q))) Or DisjointUnion(:m :p :q)
If p then q	SubClass(:p:q)
p if q	SubClassOf(:q :p)
p if and only if q	EquivalentClass(:p :q)
Not both p and q	ObjectUnionOf(ObjectIntersectionOf(:p ObjectComplementOf(:q)) ObjectIntersectionOf(:q ObjectComplementOf(:p)))
Neither p nor q	ObjectUnionOf(ObjectComplementOf(:p) ObjectComplementOf(:q))
p whether or not q	p

4.5.1.7 Modal Operation

Modal operators are used in SBVR to express structured or operative/behavioural rules. Structured rules express criteria for business facts and business rules that cannot be violated by people, while operative rules refer to conducts to manage business activities and may be affected by the will of people thereby being possible to be violated by people. The modal operation keywords are listed below:

Modal operation in SBVR	Rule type
It is obligatory that p	Operative rule
It is prohibited that p	Operative rule
It is necessary that p	Structural rule
It is impossible that p	Structural rule
It is possible that p	Structural rule
It is permitted that p	Operative rule
Must	Operative rule
Must not	Operative rule
Always	Structural rule
Never	Structural rule
May	Operative rule

Some examples of structural rules include:
It is obligatory that a student passes the final exam.
It is possible that a teacher works in more than one school.
The examples for operation rules are:
A student always has a lot of homework.
A teacher must not be late.
A teacher is never late for classes.

Demuth and Liebau (2007) argue that structured ruled are possible to be mapped to R2ML rules whereas operative rules are impossible to achieve the mapping. R2ML is more expressive than SWRL rules since its integration with Object Constraint Language (OCL) (OCL 2000), Rule Markup Language (RuleML) (RuleML 2011) and SWRL (R2ML 2008). In this sense, we argue that operative SBVR rules can not be mapped to SWRL while structured SBVR is possible to be mapped to SWRL rules.

In addition, the keyword phrase 'only if' is often used in combination with some of the key words and phrases, for example, 'it is possible that p only if q', which is equivalent to 'it is necessary that not q if not p'.

Further, the key word 'only' can also be used before a preposition in combination with 'may'. For example, *A book may be lent only to a student with student card* is equal to *A book must not be lent to a student that does not have a student card.*

4.5.1.8 Other Keyword

Besides the aforementioned keywords, we list other keywords in SBVR and analyze their possible mapping to OWL 2, as listed below:

Other keyword	Meaning and translation
The: e.g., 1) A teacher is absent if and	Class: Teacher

only if the teacher is ill	
a or an: Universal or existential quantification e.g., 1) each student has at least a name 2) a student has exactly one SSN (each person) 3) a student has dog as pet.	1) SubClassOf(:student DataMinCardinality(1 :hasName xsd:string)) 2) SubClassOf(:student DataExactCardinality(1 :hasSSN xsd:string)) 3) ObjectSomeValuesFrom(:hasPet :Dog)
Another: e.g., A student x that helps another student y is happy.	differentIndividuals(:x :y)
A Given: e.g., On a given date, a student has discussion with his teacher in a given classroom.	DataPropertyAssertion(:hasDiscussionIn: classroom) DataPropertyAssertion(:hasDiscussionOn :date)
That: 1) equals to "the" to refer to make a pronominal reference to a previous use of the same designation; 2) Used between a noun and a fact type. e.g., a person that teaches in schools is teacher. 3) It is necessary that	1) refer to "the" 2) SubClassOf((ObjectIntersectionOf(:Person ObjectSomeValuesFrom(:teachesIn :School)) :Teacher) 3) structured rule
Who: e.g., a person who studies in schools is student.	SubClassOf((ObjectIntersectionOf(:Person ObjectSomeValuesFrom(:studiesIn :School)) :Student)
is of: the reverse property of "has". e.g., "If a person has a name then the name is of the person.	InverseObjectProperty(:has :isOf)
What: e.g., What lesson is assigned to the teacher John?	Herein, DL query can be used to query the result of what. For example, what lesson is assigned to the teacher John? We can use DL query to get the instances of lesson that are assigned to a given teacher John. ObjectIntersectionOf(:Lesson ObjectSomeValuesFrom(:isAssignedTo :Teacher))

4.5.1.9 Object Identifier

In SBVR, an object can not only have several identifiers (property to uniquely identify an object or a person), but also one identifier can be used by several objects in different contexts. For example, a person can be represented by his name, social security number, or identification number. Peter can be used to describe a person or an animal.

In OWL 2, each class is uniquely identified by a set of properties (object property or data property). For example,

HasKey(:LearningService() (:hasName :hasTeacher :hasStudent :hasCharacteristics))

This example means that a service is uniquely identified by a set of properties, such as hasName, hasProvider, hasCustomer and hasCharacteristics.

Therefore, if an object in SBVR is uniquely identified by a set of properties, we can find its counterpart in OWL 2. For example, a person is uniquely identified by the SSN. Its counterpart in OWL 2 is: HasKey(:Student() :hasSSN).

In addition, compared to OWL DL, OWL 2 DL partially allows the punning, for example, a name can be both a Class and Individual, or both a Class and Object Property, or both an Individual and a property. However, OWL 2 also imposes additional restrictions and stipulates that two different classes should not have the same name, two different properties should not have the same name, and two individuals should not have the same name. For example,

> Declaration(Class(:LearningService)) Declaration(Class(:Provider)) Declaration(ObjectProperty(:provides)) ClassAssertion(:Provider :p1) ObjectPropertyAssertion(:provides :p1 :LearningService:)

In this example, the concept 'LearningService' is both a class and an individual. However it is not allowed to define two classes or two individual with the same name 'LearningService'.

4.5.2 Example

As aforementioned, SWRL can improve the expressiveness of OWL in the aspect of describing business facts and rules. We regulate that when mapping from SBVR to OWL 2 DL and SWRL, if OWL 2 DL axioms cannot express SBVR rules, then we take SWRL into consideration. Generally, SWRL does not support complex disjunction (e.g., disjunction of property), negation of property, and quantification, but it supports the disjunction of classes and the negation of classes (SWRL 2011).

The general process to achieve the mapping between SBVR and OWL axioms and SWRL rules are set out in three steps:

1- Use OWL ontology to describe SBVR vocabularies;

2- Evaluate if OWL axiom can capture SBVR rule; and

3- If OWL axioms are not competent, then introduce SWRL to achieve the mapping. If both of them fail to achieve the mapping, it implies that the SBVR expression cannot be mapped.

We present an example to demonstrate how to manually accomplish the mapping based on our aforementioned theoretical discussion to the mapping.

The SBVR rule is shown below:

> It is necessary that a BookBorrow is open and the BookBorrow is not an international borrow then the borrowed book of this borrow is owned by the

local area of the pick-up branch of the BookBorrow.

Step 1: List all fact types and then decompose them into unary or binary fact types:

A BookBorrow is open,
the BookBorrow is not an international borrow,
The borrowed book of the BookBorrow,
BookBorrow is owned by the local area,
The local area of the pick-up branch,
The pick-up branch of the BookBorrow

Step 2: Present the Knowledge Structure: the knowledge structure is illustrated in Figure 4.6. The dash line denotes the effect to be validated by this rule.

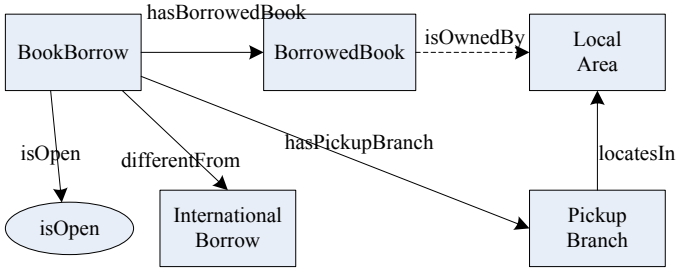


Figure 4.6 Knowledge structure

Step 3: DL-safe SWRL rule: We can directly transform SBVR rules into SWRL rules, shown in Table 4.13.

Table 4.13 Example of the mapping from SBVR to SWRL

SWRL rules:
BookBorrow(?a), InternationalBorrow (?b), LocalArea(?c), PickupBranch(?d), BorrowedBook(?e), isOpen(?a, true), DifferentFrom(?a,?b), hasBorrowedBook(?a,?e), hasPickupBranch(?a,?d),locatesIn(?d, ?c) -> isOwnedBy(?e, ?c)

Another way to map SBVR rules to OWL2 DL and SWRL is by introducing a class named SpecialBorrow to depict the expression "a BookBorrow is open and the BookBorrow is not an international borrow". This expression can be captured by OWL axioms, namely:

Equavalent(:SpecialBorrow ObjectIntersectionOf(:BookBorrow DataSomeValuesFrom(:isOpen DatatypeRestrictin(xsd:Boolean "true"^^xsd:Boolean))ObjectComplementOf(:InternationalBorrow)))

The corresponding DL-safe rule is shown as follows:

BookBorrow(?a), SpecialBorrow (?b), LocalArea(?c), PickupBranch(?d), BorrowedBook(?e), SameAs(?a,?b), hasBorrowedBook(?a,?e), hasPickupBranch(?a,?d),locatesIn(?d, ?c) -> isOwnedBy(?e, ?c)

Step 4: Then we justify the transformation with the tool Protégé 4.1 that is an open source ontology editor and knowledge-based framework (Protégé 4 2011) and we query the result with its plug-in DL-query. The result is shown in Figure 4.7. As shown in Figure 4.7, the resulting instance that satisfies the query '*isOwnedBy some LocalArea*' is '*aBorrowedBook*', which meets our expectations.

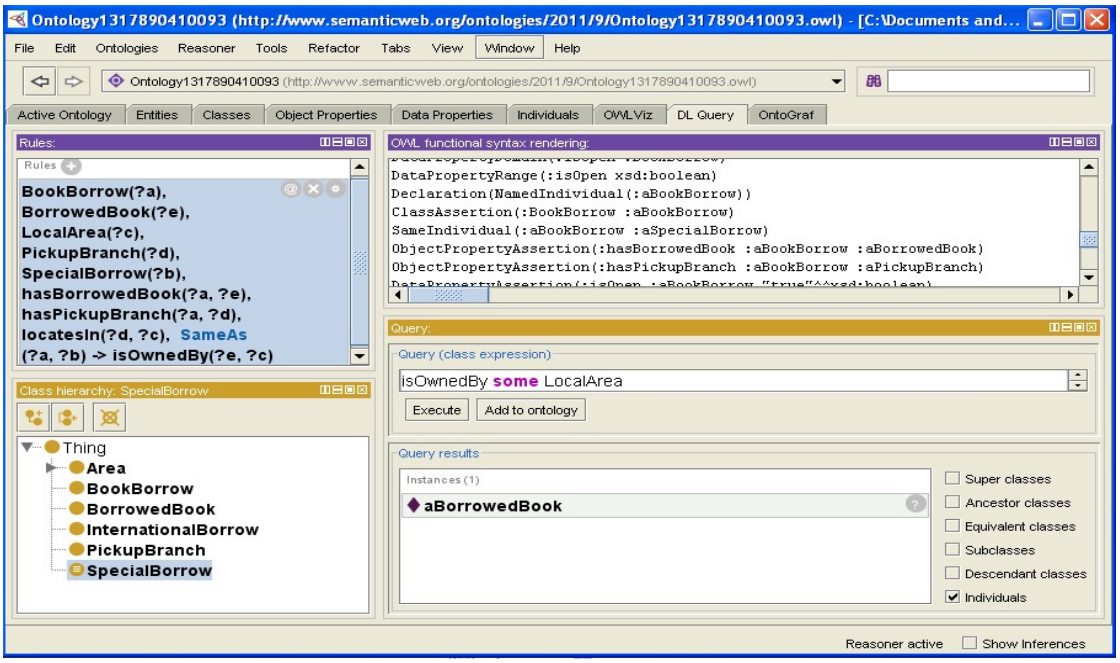


Figure 4.7 Inferred fact validated in Protégé 4.1

4.6 Conclusion

Service systems are highly complex and difficult to study, design and implement due to their socio-technical-organizational nature. In service systems, we identified two kinds of gaps, gaps between service perception and service expectation and gaps concerning how to connect various service actors (customers, business workers, IT specialists, etc) that use different models and languages to work together to co-design an innovative IT-enabled service.

In this chapter, we presented a requirements model to deal with these two gaps. The requirements model supports the transformation of unclear, unstructured and imprecise customer needs in natural language into customer wants in an *i**model and from customer wants to customer demands in SRML, from SRML to SBVR-based customer requirements, and from SBVR to implementation rules based on OWL 2 DL and SWRL rules. We argue that the proposed requirements engineering model is advantageous in the following aspects:

1- a common model to connect customer, business workers, and IT specialists) that use different models and/or languages and to take into account the software service level as well.

2- Takes into account top-down and side-way collaboration to concretize abstract and low level customer needs to customer wants, the to customer demands, and at last to precise customer requirements.

3- Propagates requirements changes from high level to low level.

4- SRML model is quite appropriate to specify customer demands, and also enables the enable QoS-based design, trade-offs and negotiation between service providers and customers and measure customer satisfaction.

6- Appropriate to IT-enabled service systems where IT-specialists play an active role in connecting customers, business experts, and developers to design innovative and customized IT-enabled services.

7- Supports a dynamic decision-making process to determine the appropriate strategy to specify requirements.

However, compared to OWL 2 DL and SWRL, SBVR has more powerful expressiveness, which leads to the fact that some SBVR expressions cannot be mapped to SWRL rules. R2ML or the combination of OWL 2, DL-safe SWRL rules and Description Graph seems to be effective in improving the expressiveness of OWL 2 and SWRL. One part of our future work will consist in taking into account these languages.

Chapter 5

Data-driven Collaboration Model

Abstract: In IT-enabled KIBS, IT specialists play an active role in integrating various service actors to collaborate for value co-creation. Due to the exchange of explicit business knowledge and the participation of human actors, their collaboration is quite ad hoc and dynamic. In addition, there are various ways to achieve customer satisfaction but which choice will be made is unclear in the beginning of collaboration. In this sense, various service actors (customers, business experts, IT specialists, developers, etc.) have to collaborate with each other to jointly take part in all levels and aspects of decision making to dynamically determine the strategy to deliver services. In order to enable and guide their collaboration, in this chapter, we present a hybrid method integrating an activity-driven and artifact-driven approach to model the ad hoc and dynamic collaboration. We propose to represent exchanged explicit business knowledge as business artifacts and then we present a set of collaboration patterns to model the cyclic and spiral service lifecycle. Each collaboration pattern is modelled as a set of interactions and each interaction activity is viewed as a directed data flow from one service actor to other service actors to exchange business artifacts. We propose to introduce business rules to invoke appropriate web services for updating artifacts (including their attributes and states transition). In addition, we propose a structure for an artifact-based collaboration pattern, which could be used to reuse collaboration patterns and improve design efficiency.

5.1 Introduction

In the service sector, KIBS firms have emerged as business operations heavily reliant on professional knowledge to support customers' business processes (Hamdouch and Moulaert 2006). KIBS firms are currently experiencing dynamic growth enabled by the emergence of ICT and are viewed as an instrument of technological and organizational innovation characterizing new trends in the service economy (Aslesen 2007, Mullera and Doloreuxb 2009). A typical knowledge-intensive service system mainly consists of entities such as service providers, service consumers, exchanged explicit business knowledge and delivery channels (Peng et al. 2009). In KIBS firms, service providers and customers collaborate with each other to co-create value. In their collaboration, they mainly exchange explicit business knowledge, which could be in the format of formal (i.e., ontology), semi-formal (e.g., XML file), or informal data (i.e., e-mail). Also different customers have different requirements, inhabiting in different environments, consequently, they require personalized and customized handling of their cases. In this sense, their collaboration activities are fairly ad hoc and dynamic. In addition, as mentioned before, in service systems, customers take part in all levels and aspects of decision making to gradually co-formulate strategy to achieve customer satisfaction with service providers.

In such a context, in the previous chapter, we have presented a requirement engineering approach to enable the collaboration of service actors to concretize unclear, unstructured, imprecise customer needs into clear, structured, and precise customer requirements at design time.

In addition, we also broadly discuss that, based on SBVR requirements, IT specialists can collaborate with business experts to enrich service system ontology, obtain business artifacts and generate SWRL rules that can be used to invoke computational resources (i.e., Web services) developed by software developers to update business artifacts. However, we did not discuss in detail how they collaborate with each other by using business artifacts and how business artifacts can help to achieve collaboration interoperability, flexibility and security. In this chapter, we will further elaborate how to implement the collaboration and value co-creation among various service stakeholders to co-design IT-enabled services to satisfy identified structured customer requirements.

In this study, we propose to represent explicit business knowledge in collaboration as a business artifact. As mentioned before, artifacts are self-contained business records that include attributes, states and life cycles that reflect the changes in these states. The artifact concept not only describes a business entity, but also encompasses knowledge about what to process without explaining how to do it (Hull 2008). We also present a

data-driven collaboration model to deal with these issues and our collaboration model is expected to achieve the following engineering objectives:

1- To provide flexibility to human workers, especially knowledge workers to assess customer needs after which further actions have to be performed and scheduled accordingly.

2- To facilitate the decision making of various service actors (providers and customers) to gradually determine the appropriate strategy to deliver the service to satisfy customers.

3- Due to the lack of clarity associated with exchanging data or explicit knowledge, the collaboration should be flexible enough to allow service providers to dynamically select appropriate resources (i.e. activities, applications, or knowledgeable worker,) to process data and to send it back to the service consumer and possibly automate the selection of appropriate resources.

4- To make business experts focus on business itself rather than integrating business process, which could improve flexibility to adapt their businesses to strategic changes.

In addition, we propose a structure of an artifact-based collaboration pattern, which could be used to improve design efficiency.

This chapter is organized as follows: in Section 5.2, we articulate the term 'collaboration' and its implications as well as current work to use both a process-centric and data-centric approach to model business processes. In Section 5.3, we present an artifact model to represent exchanged explicit business knowledge in collaboration. In Section 5.4, we present our data-driven collaboration model. We will further discuss some interaction patterns and how artifact is used to achieve high-level semantics and security in Section 5.5. In Section 5.6, we discuss collaboration patterns throughout the service lifecycle and propose a structure for artifact-based collaboration patterns. Finally, we conclude our work in this Chapter in Section 5.7.

5.2 Related Work

In today's world, we are witnessing the trend that various enterprises are interconnected in a collaborative chain for value co-creation. However, the concept 'collaboration' has different levels of meanings and it has ambiguities with two other concepts: 'coordination' and 'cooperation'. In this sense, it is necessary to elaborate on the implications of the term 'collaboration' and the differences among the three concepts: 'coordination', 'cooperation', and 'collaboration'.

5.2.1 Collaboration Interpretation

The term 'collaboration' is widely used in numerous situations: "g*roup based on the complementary nature of partners (covering increasing), groups concerning enterprises focused on the same service (power increasing), groupings of providers from a same decision-maker (optimization and safety), common buying platforms (improvement of negotiation capacity) and other original networks, permanent or limited (due to specific constraints of the milieu)*" (Bénaben et al. 2006).

Kosanke (2005) proposes four levels of enterprise collaboration in terms of the level of compatibility, as shown below:

1- Level 1 (Coexistent): may exist independently in a single network,

2- Level 2 (Inter-connectable): may share or exchange information,

3- Level 3 (Inter-workable): may share functionalities or services,

4- Level 4 (Interoperable): may work according to a predefined collective behaviour.

Lauras et al. (2003) presents four levels of collaboration among enterprises in terms of their understanding levels:

1- Level 1 (communication): sporadic data exchange,

2- Level 2 (coordination): structured and instituted data sharing,

3- Level 3 (collaboration): sporadic data and applications exchange,

4- Level 4 (cooperation): structured and instituted data and applications sharing.

Camarinha-Matos and Afsarmanesh (2006) distinguish the concept of 'coordination', 'cooperation', and 'collaboration'. They view that 'coordination' is the lowest level of collaboration and it mainly involves the exchanging of information and aligning/altering activities so that more efficient results are achieved, whereas 'cooperation' involves "*not only information exchange and adjustments of activities, but also sharing resources for achieving compatible goals*". They view that 'coordination' refers to some low-level of co-working activities for a common plan among various participants who perform their respective work in an independent way and it does not refer to some high-level co-working activities, such as co-planning, co-evaluating, and co-decision-making. They define 'collaboration' as "*a process in which entities share information, resources and responsibilities to jointly plan, implement, and evaluate a program of activities to achieve a common goal*", which implies the sharing of risks,

resources, responsibilities, and mutual trust and engagement of participants to co-create value.

In addition, the collaboration among various enterprises in a collaborative network concerns the field of their information systems and their behaviours, reactivity and all the dynamic aspects highly depend on the information system and the processes, services, and data they manage (Bénaben et al. 2006). Touzi et al. (2007) view a collaborative information system as the combination of the information systems in different partners into a unique system for interoperability.

In this thesis, we adopt the definition for the term 'collaboration' from the work of Camarinha-Matos and Afsarmanesh (2006) and focus on the collaboration between various IT-enabled service systems in KIBS firms for value co-creation. Collaboration in the context of KIBS firms consists of a set of interactions between service providers and customers which define collaborative processes of assessment, planning, facilitation and advocacy for services to satisfy customer needs through communication and mobilization of internal or external resources. Such interaction is characterized by a human-driven process, also known as case management (de Man 2009) and as a practice, supports services in many areas. Among them are administrative environments, licensing in Government, market segments, call handling in call centres, advocacy, claim processing in insurance, medical diagnosis in healthcare or mortgage processing in finance, and so on. Additional industrial examples include sales and operations planning, invoice handling, business requirements in Research and Development, analysis and study in consulting and engineering firms. Case management often remains a paper-based process. All these examples are managed by knowledgeable workers in the context of service provision based on collaboration with customers.

5.2.2 Collaborative Business Process Modelling and Business Artifact

Rooze et al. (2007) classify case management for business processes into three categories including mass cases, regular cases, and special cases. They further argue that mass cases can almost be automated and completed by using workflow models, regular cases are often managed through a series of milestones whereas special cases are not predefined in advance and thereby allow the given service actor, worker or service consumer leeway in determining how to handle them. Rooze et al. (2007) claim that there is no clear direction in how to manage special cases often characterized by dynamic and ad hoc collaborative processes requiring human participation.

Conversely, activity-driven business process modelling techniques (i.e., Casati et al. 2000, Lammermann 2002, Issa et al. 2006, Ko et al. 2009, Fujii and Suda 2009, Peng et al. 2010b, etc), mainly deal with relatively static and stable business processes. However, as to human-driven and dynamic collaboration, they raise the following challenges:

1- Activities in business processes are always predefined with respect to the underlying business logic. During the execution of business processes, it is quite difficult to dynamically modify current activities and their execution order.

2- Each activity is considered as a black box and does not show what attributes or fields are updated in their input file or cases, and their current status (i.e. created, pending, closed).

3- Activity-driven modelling does not match the way people do business (Cohn and Hull 2009). Bhattacharya et al. (2007) argue that an activity-based paradigm is quite appropriate to model mass cases.

4- An activity-driven modelling approach is process-centric and focuses on technical interoperability dealing with distributed and heterogeneous applications to exchange information messages. Indeed it is difficult for enterprises to achieve dynamism and flexibility.

Because of the inadequacy of activity-driven techniques to model regular and special cases, distinct principle types of process-control paradigms seek to support them. The artifact-based paradigm (Wang and Kumar 2005, Bhattacharya et al. 2007) defines business entities as business records which are used to store relevant information pertinent to a given business context. Each artifact is characterized by its lifecycle describing its evolution from creation to achieving its goal. It is an indivisible and unique entity which has a unique identifier allowing its identification throughout the enterprise. A business artifact is an abstraction to focus businesses on core information entities that are most meaningful to business users. It supports business user activities by measuring whether or not the business is on track to achieve its business goals or milestones.

Artifact-based modelling (Cohn and Hull 2009) is relevant to modelling regular case management and case handling whereby activity control is based on state transitions and business rules. As for the special cases, the communication- (or conversation-) based paradigms (Martyn 2005, Kethers and Schoop 2000) model the process goal as a series of interactions, communications, or conversations between process participants. Individual participants perform activities which are not explicitly defined in advance to reach the process goal. The Role Activity Diagram (RAD) (Martyn 2005) and the Language-Action Perspective (LAP) (Kethers and Schoop 2000) are examples of techniques to model how people reach their

goal through communications. de Man (2009) argues that communication-based models seem to be abstract approaches and lack sufficient details to implement them and solve real world business problems as well.

Nigam and Caswell (2003) present a comprehensive discussion of using business artifacts and their life-cycles in business modelling. Kumaran et al. (2008) stress the importance of developing guidelines to identify artifacts from business processes. Muller et al. (2008) present data-driven process structures in which Object Life Cycles (OLC) of different objects are linked through OLC dependencies. Based on objects and relations between them, process structures are specified at the model level and then automatically instantiated as per a given data structure. This approach enables adaptations of data-driven process structures at design time and in the run time.

Narendra et al. (2009) introduce a unified approach for business process modelling using context-based artifacts and Web services. In this work, a business process is modelled as a set of interacting artifacts and artifacts progress through their respective lifecycles updated by Web services that are invoked by context-based business rules. However, they also recommend that the unified approach should use models that are understandable by non-IT specialists. In addition, it also has conceptual ambiguities for various interactive business artifacts.

Badr et al. (2011) identify five levels of interoperability, as shown in Table 5.1. Interoperability is broadly defined as "*the ability of a system or a product to with other systems or products without special efforts*" (Touzi et al. 2007). In this study, we respect the classification of interoperability as presented by Badr et al. (2011). The authors further argue that current initiatives to use artifacts are process-centric and mainly focus on technical interoperability but rarely deal with business level interoperability in service systems where diverse enterprises collaborate together to co-create value.

Table 5.1 Interoperability level (source: Badr et al. 2011)

Interoperability levels	
Business	Ability of diverse enterprises to collaborate together to co-produce added-value despite their different business strategies, organizational constraints, and IT infrastructures
Organizational	Ability to seamlessly interoperate pre-existing organizational structures and their business processes in response to different types of business collaborations
Semantic	Ability to automatically interpret the data exchanged meaningfully and accurately in order to produce useful results
Syntactic	Ability to integrate and exchange data by using specified data formats
Technical	Ability of systems to technically provide data transfer protocols to communicate with other systems and operate effectively together

In terms of these initiatives, it is possible for artifacts to make interoperability focus on the needs of the business itself rather than on the needs of integrating business processes. However, since data passing from one partner to another is a core activity in any level of collaboration and interoperability, service actors always have difficulties in identifying the data they require from their partners due to a lack of common understanding and restricted access to data resources.

In our approach, we present an ontology to provide a common understanding for interactive artifacts. In addition, business interoperability can be built upon artifacts that can cross organizational boundaries sharing common understanding and whose life cycle can be split into parts, each part associated with a partner in compliance with some access rights. Each artifact is associated with a lifecycle that shows all the changes in states that the artifact goes through.

In addition, although collaboration activities in service systems are quite ad hoc, they also have to respect to the service lifecycle. In other words, in service system, some collaboration activities in terms of service lifecycle may occur repeatedly, for example, requirement specification and service evaluation. In order to improve the design efficiency by reusing recurring collaboration activities from various service systems, we propose to introduce collaboration patterns to formulate recurred collaboration in terms of the service lifecycle for value addition. Papageorgiou et al. (2009) regard collaboration patterns as a means to capture best practices about recurring collaborative problems and solutions. In this study, we explore collaboration patterns throughout the service lifecycle and present a structure of artifact-based collaboration patterns, which could be used to improve design efficiency.

5.3 Business Artifact to Represent Explicit Knowledge

The exchanged business knowledge in collaboration could be any format of data including structured, semi-structured, and unstructured data. Structured data are described by data models to facilitate their storage in data management systems for example, relational databases. Unstructured data refers to information that does not have a data model. Examples of unstructured data include word processing documents, email messages, pictures, digital audio and video. Semi-structured data is a form of structured data that does not respect data models associated with data management systems. Semi-structured data, for example, XML documents, contain tags to separate semantic elements and hierarchies of fields within the document content. In this study, we take into account semi-structured data and structured data for automating the collaboration process for the sake of automa-

tion since the processing of unstructured data requires manual work. In the remainder of this thesis, unless otherwise indicated, the exchanged business knowledge denotes semi-structured or structured data.

Business Artifacts are self-described business records and are mechanisms used to record chunks of information that can be used by a business expert to manage a business. Changes in an artifact are usually reflected in an Artifact LifeCycle (ALC). These changes are the result of executing one or more tasks in business processes. A business process complies with a business model that prescribes the activities to be performed as part of a business operation, the sequencing of these activities, and the inputs and outputs of these data. However, in the context of IT-enabled service systems, collaboration is ad hoc and dynamic and a predefined activity-based business process does not support dynamic decision making.

Artifacts can be modelled with any suitable information modelling approach such as Entity-Relationship diagrams (Song et al. 1995) to facilitate the understanding of them by business professionals. At the technical level, artifacts can be modelled as XML schema which specifies exchanged input and output messages.

In this thesis, we use a 3-tuple to represent an artifact model <attribute, state, lifecycle> and each item in the 3-tuple is a vector which includes various items. For example, as to an artifact '*A'*, it can be described with the following expressions:

$$Artifact_A = \{Attribute_A, State_A, ALS_A\}$$

$$Attribute_A = \{Attri_1, Attri_2, ..., Attri_k\}$$

$$State_A = \{Sta_1, Sta_2, ..., Sta_m\}$$

$$ALS_A = \{State_A, Sta_1, Sta_m, Rule, Trans\}$$

where $Artifact_A$ denotes the artifact A; $Attribute_A$ denotes its attribute set; $State_A$ denotes its state set; $Attri_i$ denotes the ith attribute in $Attribute_A$ $(i \in [0,k])$; Sta_j denotes the jth state in $State_A$ $(j \in [0,m])$; k denotes the count of attributes for the $Artifact_A$; m denotes the count of states of $Artifact_A$; ALS_A denotes the lifecycle of $Artifact_A$; *Rule* denotes a set of transition rules; and *Trans* is a set of transitions. A rule *r* in *Rule* consists of three components (event E, condition C, and effect F) and is written as E[C] →F. A transition is defined from a state to another as $t = (sta_i, r, sta_j)$ where sta_i and sta_j are in the same ALC. In general, the transition from one state to another state, the updating of attributes and the execution of business operations or tasks are associated with Web services.

5.4 Data-driven Collaboration Model

In Chapter 3, when discussing the ontological view of our proposed reference model, we presented a high-level collaboration ontology. Since in this section, we have to use that ontological model to elaborate our collaboration model, we re-list it, as shown in Figure 5.1.

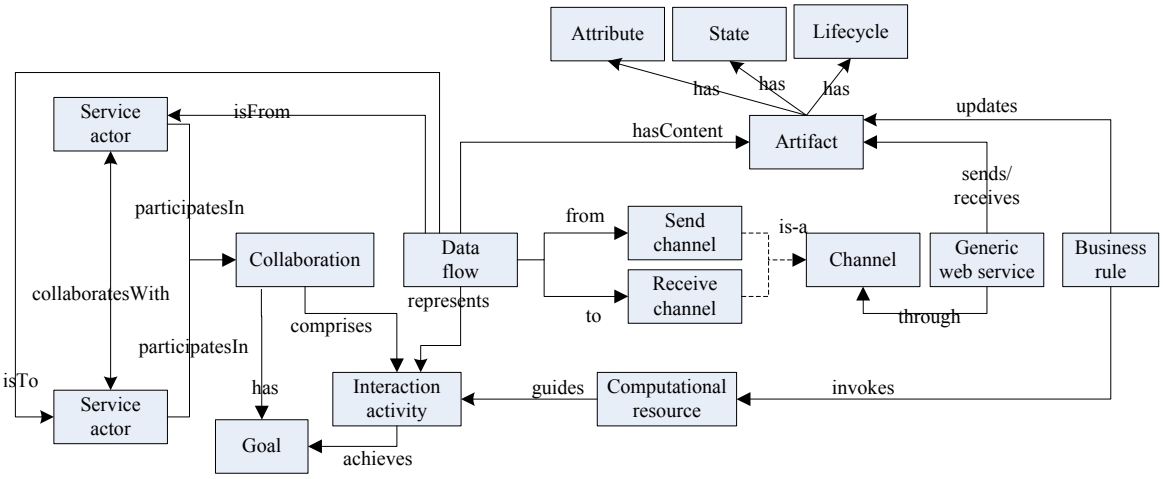


Figure 5.1 Collaboration meta-model

Generic web services: unlike traditional approaches that attempt to ensure collaboration interoperability by integrating business processes through composing various Web services, we use a 'generic' Web service whose role is limited to setting up a connection of two service actors for collaboration and taking charging of sending or receiving artifacts. They facilitate the exchanges of business artifacts expressed in terms of input and output messages.

Since artifacts are self-contained, each partner can select an appropriate computational resource (i.e., web services and SaaS) to update the artifact's attributes and make this artifact transit from its current state to a new state regarding this artifact's lifecycle. In such a way, partners do not have to reveal their internal applications or Web services but just guaranteeing that they will process the artifacts based on the current attribute values and current states of these artifacts. This could help to reduce the complexity for business workers to manage their day-to-day businesses.

Computational resources: in the previous chapter, we have presented this concept. In this study, the computational resource used to denote business operations or tasks could be web services or SaaS. They are invoked by business rules to update business artifacts. Generally, a web service is expressed in terms of inputs, outputs, pre-conditions, and post-conditions using OWL-S and semantic Web service specifications (Martin et al. 2003, Fritz et al. 2009, Deutsch et al. 2007).

SWRL rules: specify the invoking of appropriate Web services to update artifacts, such as which service can be performed by whom, under what condition, and how to update artifacts. In general, a business rule for an artifact is composed of events, preconditions and effects. Conditional effects typically create new attribute values or change current state to another state in terms of the artifact lifecycle. As mentioned before, in this thesis, we use SWRL to represent implementation rules since it is a formal rule language that can be directly understood by computers and integrated with OWL ontology to express semantic business rules. It is also easy to read, write, and understand even for non-professionals. In addition, an inference engine is required to read and reasons over SWRL rules to choose the appropriate Web service to update artifacts in the run time in order to provide guidelines for collaboration and decision making. Details about the inference engine will be presented in the implementation part (See Chapter 7). In this study, we classify two kinds of business rules in service systems:

1- **Generic rules**: these kinds of rules refer to generic business facts in service systems based on high-level service system ontology. We present two examples of generic rules:

Rule 1: this rule means that collaboration takes place between different service actors.

```
ServiceActor (?sa1), ServiceActor(?sa2),
,Collaboration(?c), participates(?sa1,?c), participates(?sa2,?c),
collaborates(?sa1,?sa2) ->  differentFrom(?sa1,?sa2)
```

Rule 2: if two service actors have dataflow, then the two service actors collaborate with each other.

```
ServiceActor (?sa1) ,ServiceActor(?sa2),
Dataflow(?df),isFrom(?df,?sa1),isTo(?df,?sa2)
-> collaborates(?sa1, ?sa2)
```

2- **Domain rules**: refer to rules in certain domains. For example, in the e-learning domain, we can formulate the following domain rule: 'if a student does not have microphone and handset, the rule will invoke a purchase service to help the student to buy these facilities'. In addition, domain rules also involve business rules to manage artifacts, such as which service can be performed by whom, under what condition, and how to update artifacts. We list three domain rules as examples:

Rule 1: if a student has a microphone and handset, then the student is eligible to take the course.

```
Student(?stu), resources(microphone), resources(handset)
->isEligible(?stu, true)
```

Rule 2: if a student's customer type is 'golden', then the student can get 10% discount.

```
Artifact(stuA) ,Artifact(ordA)  ,Attributes(CustType) , Attributes(discount)
,has Attributes(stuA, CustType)  ,hasAttributes(ordA, discount)
,hasValue(discount, ?dis) ,hasValue(CustType, "golden")
->
```

```
hasValue(?dis, 0.1)
```

Rule 3: if the order artifact has current state 'shipped' and the returnedItem artifact has current state 'created' and the attribute 'pendingDays' for the order artifact is less than 10, then the order's state is changed to 'in_arguing'.

```
Artifact(ordA) , Artifact(reiA) , States(shipped) , hasStates(ordA, shipped)
,Attributes(pendingDays) ,hasAttributes(ordA, pendingDays)
,hasValue(pendingDays, ?x) ,lessThan (?x,10) ,States(in_arguing) ^
hasStates(rei, created)
->
hasStates(ordA, in_arguing)
```

In KIBS firms, the collaboration between providers and customers is ad hoc and dynamic and the pivotal substance/content of their collaboration is explicit business knowledge. Generally a collaboration activity has a goal which may be achieved by a set of interaction activities. Herein we define each interaction activity as one data-sending activity to a service actor or data-receiving activity from another service actor. We introduce a data flow to represent each interaction activity. Each data flow requires specifying interaction channels and exchanged business artifacts.

Based on the proposed collaboration model, we re-outline the following features for collaboration in KIBS firms that are also presented in Chapter 3: 1) each service actor participates in one or many collaborative efforts; 2) each collaboration activity has a goal; 3) each collaboration goal is achieved by a set of interaction activities; 4) each interaction activity is represented by a directed data flow that is from one service actor to another service actor; 5) each flow is defined by two generic web services supported simultaneously by the service provider and the service customer; 6) exchanged business knowledge in an interaction activity is represented as business artifacts (i.e., order, invoice, etc); 7) business rules are developed to invoke common web services that are used to manipulate artifact attributes, update artifact states and lifecycle; 8) after receiving an artifact, since artifacts are self-contained business records, a reference engine will reason on business rules and decides, based on the current state of the artifact and shared business rules, which operation/web service should be invoked and what its new state is before returning the artifact to the sender; and 9) there are a variety of options to reach a collaboration goal and the decision made will gradually determine the appropriate strategy to guide interaction activities and the invoking of common web services to update artifacts.

In order to explain how to use our data-driven approach, we introduce an example here.

Example: we present an example describing a student's purchase of study materials and equipments from e-learning service systems to dem-

onstrate how artifacts maintain business interoperability in service systems. The general picture of the presented example is shown in Figure 5.2.

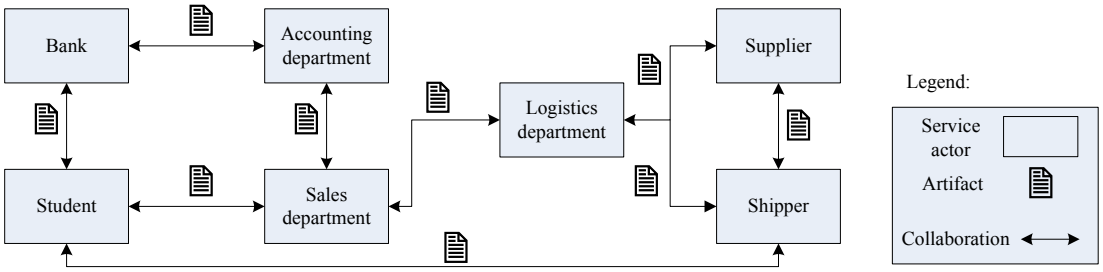


Figure 5.2 Collaboration framework based on the presented case

At first we briefly discuss the business process of this scenario. Initially a student places an order to the Sales actor. The Sales actor checks the student's previous purchase history with the Accounting actor. The Accounting actor calculates the student's bill based on his or her history details (e.g., whether eligible for reward points, discounts, etc.) and sends it to the Sales actor. The Sales actor prepares for the detailed purchase order based on the bill and sends it out to the Stock actor. As per those products that are in stock, the Stock actor sends a shipment request out to the Shipper actor, who is now in charge of delivering the products to the student. For those products not in stock, the Stock actor sends a supply message out to the requisite Supplier actor who will supply products to the Shipper actor for subsequent shipments to the student. In addition both the student and the Accounting actor need to collaborate with the Bank to deal with the payment for products in order.

In this example, the Accounting actor needs to collaborate with both the Sales actor and Bank actor. One collaboration goal between the Sales actor and the Accounting actor is to evaluate the student's bill based on his or her historical purchase information to determine whether the bill is eligible for reward points or discount. To achieve this goal, it requires multiple-interactions between these two service actors. For example, the Sale actor first sends the student's personal portal information and bill information to the Accounting actor and asks it to check the student's historical purchase records. The Accounting actor investigates the student's history purchase information and current bill information and determines if the student satisfies the discount requirements and then sends the evaluation result to the Sales actor. At last the Sale actor sends back an acknowledge message to inform the Accounting actor. The referred artifacts in these interaction activities may refer to 'Student' 'Bill', 'Order', 'OrderItem', etc. The 'Student' artifact is used to record the profile information of the student and its purchase history. The 'Bill' artifact records the student's pay-

ment information. The 'Order' artifact is used to record the student's shopping list of products and the 'OrderItem' artifact is used to record the student's shopping.

In the running example, the 'Student' artifact could be jointly updated by the Sales and Bank enterprises to ensure that the student's purchase and credit history are regularly maintained.

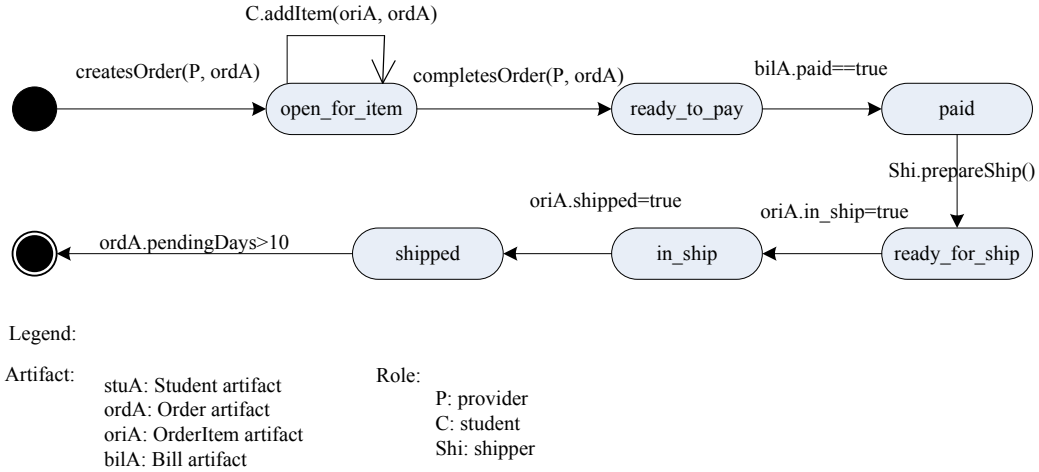


Figure 5.3 Lifecycle model for 'Order' artifact

In order to demonstrate how to design artifacts and how to develop web services to update artifacts and develop business rules to invoke web services, we present an 'Order' artifact as example. The lifecycle model of the 'Order' artifact is shown in Figure 5.3. As shown in this Figure, the 'Order' artifact is shared by customers, providers and shippers.

Table 5.2 Examples of business rules for the 'Order' artifact

Artifact: Order	Transition rules
Attributes: OrderID: string CustomerID: string Bid: float ScheduleDate: date Discount: float GrandTotal: float finishOrder: bool pendingDays: int States: open_for_item, paid ready_for_ship, in_ship, shipped, closed	**Service 1: setAttributes(custCredit)** Rule1: Artifact(stuA) , Attributes(Bid) , Attributes(CustCredit) ,has Attributes(ordA, Bid) ,hasAttributes(stuA, CusCredit) ,hasValue(CusCredit, ?credit) ,hasValue(Bid, ?bid) greaterThanOrEqual (?bid,2000) ,add(?new, ?credit, 80) ->hasValue(CusCredit, ?new) Rule 2: Artifact(stuA) , Attributes(Bid) , Attributes(CustCredit) ,has Attributes(ordA, Bid) ,hasAttributes(stuA, CusCredit) ,hasValue(CusCredit, ?credit) ,hasValue(Bid, ?bid) ,greaterThanOrEqual (?bid,1500) ,lessThan (?bid,2000) ,add(?new, ?credit, 60) ->hasValue(CusCredit, ?new) Rule 3: Artifact(stuA) , Attributes(Bid) , Attributes(CustCredit) ,has Attributes(ordA, Bid) ,hasAttributes(stuA, CusCredit) ,hasValue(CusCredit, ?credit) ,hasValue(Bid, ?bid) ,greaterThanOrEqual (?bid,1000) ,lessThan (?bid,1500) ,add(?new, ?credit, 40)

-> hasValue(CusCredit, ?new) …

In Table 5.2, we present some business rules and a web service as examples for the artifact. For example, the web service 'setAttributes(custCredit)' is used to update the custCredit attribute of the 'Student' artifact, which regulates that if the bid is no less than 2000, then the customer's credit will be added by 80, otherwise if the bid is no less than 1500 and less than 2000, then it will be added by 60, and finally if the bid is no less than 1000 but less than 1500, it will be added by 40 …

In this example, various service actors share the 'Order' artifact and jointly manage its business rules to invoke appropriate web services to update the artifact. In addition, the 'Order' artifact also interacts with the 'Student' artifact to update the attributes of the 'Student' artifact (i.e., CustType, CustCredit). However, in service systems, it is not obligatory for service actors to share business artifacts and jointly manage business rules if they do not have enough trust in one another. In addition, we do not take into account security issues in collaboration. In this sense, in the next section, we will discuss how to achieve interoperability and security in cases where service actors do not have enough mutual trust.

5.5 Artifact for Interoperability and Security

In the previous chapters, we have presented an example to show how to achieve business interoperability by sharing business artifacts and jointly managing business rules among service actors. However, in service systems, service actors do not always share business artifacts and jointly managing business rules. Further, two partners can exchange several artifacts and sometimes expect to partially or totally hide their artifacts' contents for the sake of privacy or security. This will typically lead to the proposal of a dedicated flow in which a specific artifact can be exchanged back and forth between two partners. In this context, we differentiate between public/common and private artifacts to ensure the privacy and security for service actors.

Public/Common artifact: is exchanged between service actors. This case holds the prerequisite that partners who share common artifacts must have enough trust in each other. If this precondition is satisfied, this method can greatly improve the interoperability and efficiency of collaboration. This is a convincing case in KIBS firms, in which various service actors trust each other and collaborate with each other to achieve a common goal.

Private artifact: refers to an artifact that is private and not shared or partially shared by different service actors.

In practice, each enterprise has relevant and complete definitions of artifact attributes and lifecycles that it does not wish to reveal to its partners. Consequently, at design time, partners should agree in advance on public artifact structures and their lifecycles and regulate approaches and mechanisms to map between public artifacts and private artifacts. In the run time, service actors will follow their regulated mechanism to achieve the mapping between public artifacts and private artifacts for security.

For example, when paying for online courses via a credit card, a student generally has to provide the following information about his/her credit card: 'holder name', 'account number', 'expiration date', and 'security number'. However, for the sake of security, the student always holds the security number of the credit card as private and does not expect to share this information with other service actors.

In fact, public artifacts can be considered as reduced views of private artifacts. The analogy to the relationship between a super class and a subclass may help to understand this relationship. A public artifact is treated like a super class while a private artifact can be viewed as its derived class.

In order to obtain interoperability and security in service systems, we classify the following interaction patterns:

1- Partners have artifacts and manage them separately. Such an approach can be used in situations that demand the highest level of autonomy among interacting partners. Neither side is aware of the other side's artifacts – agreement is only reached about the messages to be exchanged and the conditions in which messages are to be exchanged. In this type of interaction, business partners communicate with each other by sending or responding to messages. As a result, we view this pattern as a request-response pattern. For example, given service actor 'A' and 'B', A holds the artifact 'a', and B holds the artifact 'b'. 'A' collaborates with 'B' by exchanging messages. Each message can be considered as an event to trigger actions to update artifacts.

2- Partners share a common public artifact and jointly manage it. In this case, two partners agree to share a common artifact and jointly contribute towards its lifecycle. They also share business rules to update the common artifact. This is to say that each partner is aware of the other partner's rule to manage the invoking of web services for updating artifacts. In this kind of interaction, the trust level between partners is very high and they share both artifact and business rules. As demonstrated in the aforementioned example, the 'Student' artifact could be jointly updated by the Sales and Bank enterprises to ensure that the customer's purchase and credit history are regularly maintained.

3- Partners share a common public artifact but manage it separately. The rules to manage the public artifact of one service actor are not revealed to its partner. In this case, the trust level between partners is not enough to make them share rules to manage their shared artifact. As demonstrated in the aforementioned example, the Sales department and the Bank could also independently update the 'Student' artifact, while ensuring privacy for their respective rules.

4- Partners share a common public artifact provided by a contractor and other partners play the role of sub-contractor and mange the artifact separately. In this case, there is only one artifact and the prime contractor and their sub-contractors separately manage the lifecycle of this artifact. This case is different from the case in which partners share a common artifact and rules to manage the artifact. The former case denotes the relationship between a prime contractor and its sub-contractors, whereas the latter describes that partners have a very high level of trust, which makes them share both artifact and rules. For example, outsourcing is a typical application of this case.

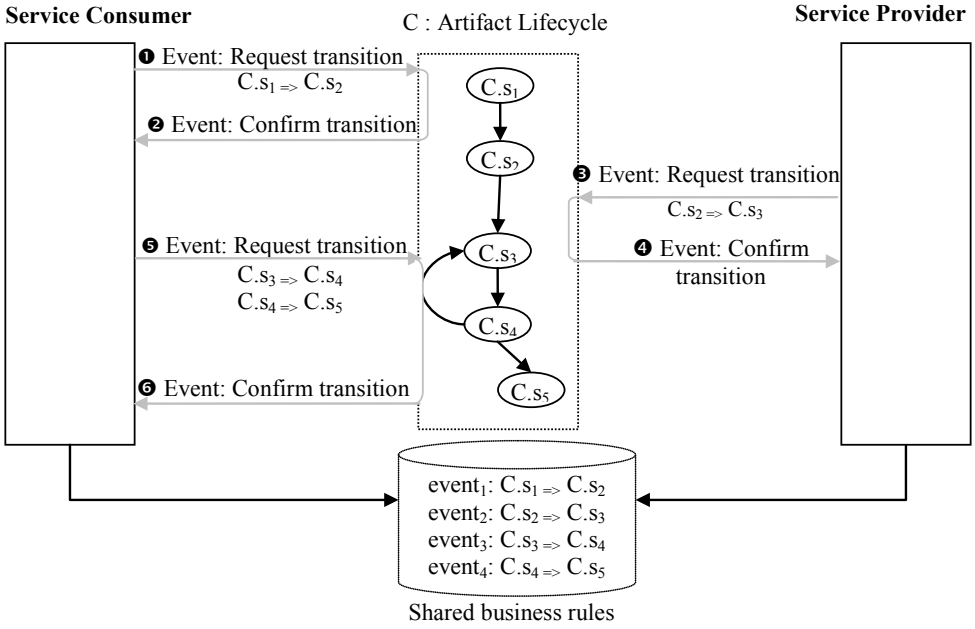


Figure 5.4 Demo for one common artifact and shared business rules

When designing a service system, designers have to analyze relationships among service actors in order to choose the appropriate interaction pattern. However, in this thesis, we only take into account one interaction pattern, namely service actors' shared common artifacts and business rules, as shown in Figure 5.4. In our future work, we will take into account other kinds of interaction patterns and design mapping mechanisms to en-

able the mapping between public artifacts and their types, namely private artifacts.

As illustrated in Figure 5.4, although it simply shows the interaction between two service actors, we argue that this pattern can be extended to multiple service actors. In the meanwhile, it is worth noting that this pattern holds the prerequisite that two service actors that have a common artifact and shared business rules should highly trust each other. Also, this prerequisite completely conforms to the core logic that various service actors trust each other and collaborate with each other to co-create innovative and customized services in KIBS firms.

Our presented collaboration model is data-centric rather than process-centric, which makes business people focus on their businesses rather than integrating various business processes. As a result, business people gain increasing flexibility to manage their day-to-day businesses and adapt them to changes. We also add that in spite of the diversity of service systems, throughout the service lifecycle, some collaboration activities recur in different service systems. The efficient way to design a service system is to reuse these common collaboration activities. In this sense, in the next section, we discuss some reused collaboration patterns throughout the service lifecycle.

5.6 Collaboration Patterns throughout the Service Lifecycle

A service lifecycle refers to the succession of strategies used by the service provider to manage the service from creation to completion. Service lifecycle goes through multiple phases over time as it moves through the collaboration between service providers and consumers. Effectively managing the service lifecycle (Wall 2006, Räisänen et al. 2005) is fundamental to achieve success within service systems.

Service lifecycle can be abstracted as a succession of operations which can be divided into design-time aspects and run-time aspects. Design-time aspects include service modelling, development and packaging while run-time aspects include the deployment, discovery and execution of these services (Wall 2006).

Rittgen (2006) divides service lifecycle into three phases, the contract phase, the operation phase, and the result phase, respectively. In the contract phase, the service provider seeks to establish an agreement with the service customer. During the operation phase, the service provider and service customer collaborate with respect to their contract. Lastly, the result phase evaluates customer satisfaction with respect to the delivered service.

As opposed to traditional software systems or product systems, service systems involve value co-creation of various service actors. In our approach, the service lifecycle designates the ad hoc collaboration process, which is divided into five phases: the initialization phase, the situation assessment phase, the establishing agreement phase, the designing service phase and the service provisioning and evaluation phase respectively.

In service systems, almost all the service co-design and value co-creation processes involve these five phases. When co-designing a service, if service providers and customers can refer to some abstract patterns to collaborate with one another, they can accomplish the service design more efficiently and quickly. In other words, they can obtain more value addition if they can refer to and reuse collaboration patterns.

Patterns (Gregor and Woolf 2003) are common in many areas of Computer Science and are used to describe how to solve a problem in many different situations. They are general reusable solutions to commonly occurring problems. Collaboration patterns can be understood in terms of how users act in dialogues by using artifacts (Krange et al. 2002). de Moor (2006) views collaboration patterns as conceptual structures to capture socio-technical knowledge to optimize collaboration processes. In this study, we adopt the definition of collaboration patterns from the work of Verginadis et al. (2009a) whereby they are "*a prescription which addresses a collaborative problem that may occur repeatedly in the environment. It describes the forms of collaboration and the proven solutions to a collaboration problem and appears as a recurring group of actions that enable efficiency in both the communication and the implementation of a successful solution. The collaboration pattern can be used as is in the same application domain or it can be abstracted and used as a primitive building block beyond its original domain*".

We generalize the following benefits of collaboration patterns to ad hoc and dynamic collaboration for value addition:

1- Capture best practices about recurring collaborative problems and solutions (Papageorgiou et al. 2009).

2- Facilitate informal, dynamic and ad hoc knowledge-intensive collaboration (Verginadis et al. 2009b).

3- Capture knowledge exchanges taking place during collaborations and enable the automation of knowledge flows across an organization (Sarnikar and Zhao 2007).

4- Trigger human or machine operations within a specific context and in response to particular events (de Moor 2006).

In this study, we present a set of collaboration patterns to model the cyclic and spiral service lifecycle. Each collaboration pattern has a

common goal that is achieved by a set of interactions. Each interaction activity is viewed as a directed data flow from one service actor and to other service actors to exchange business artifacts. In this study, a collaboration pattern is mainly composed of the following elements: information describing a context, referred service actors, referred business artifacts, collaboration goal, collaboration process, and business rules controlling the modification of artifact attributes and state transitions.

In the remainder of this section, we first discuss some collaboration patterns throughout the five-phase service lifecycle and then we present a structure of collaboration pattern. The service lifecycle is illustrated in Figure 5.5.

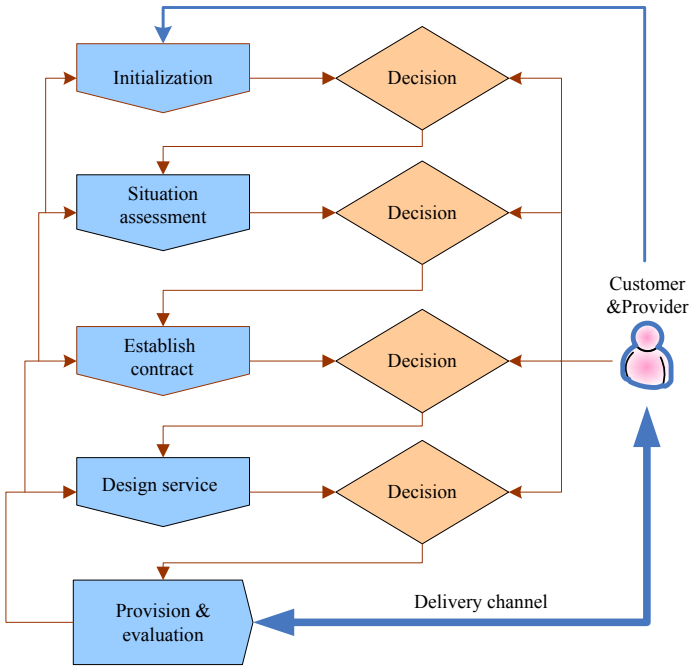


Figure 5.5 Customer's role in service lifecycle

Initialization phase (Phase 1): describes the solicitation of a customer asking for a service in order to solve a problem, improve a situation, or satisfy certain needs.

Collaboration goal: establish a connection between providers and customers through which providers successfully satisfy customer needs.

General collaboration process: in this phase, a potential customer (called visitor) contacts the provider and asks for information about the services provided by the provider. In the meanwhile, the provider queries customer needs from the customer. The visitor perhaps simply asks for information about services and does not consume any service at the moment. If the visitor really wants to consume services at the moment, the visitor

will become a customer. A customer artifact will thus be created. In this phase, at least three artifacts may be identified: including visitor artifact, customer artifact, and customer needs.

Referred artifacts: i.e., visitor, customer, customer needs, etc.

Referred service actors: i.e., customers, requirement Engineer, etc.

Situation assessment phase (Phase 2): determines the nature and scope of the requested service by analyzing the business needs and requirements in measurable goals. In this phase, providers assess the client situation and collect detailed requirement information (i.e., preference or habit).

Collaboration goal: service providers collect detailed information in order to understand customer needs and to evaluate if they are competent enough to satisfy customer needs.

General collaboration process: in order to achieve the collaboration goal, in this phase, providers and customers have to jointly specify customer needs into customer wants with a graphical model (i.e., the *i**model). Providers then specify customer wants to customer demands and describe it with SRML. After that, providers evaluate if they can satisfy customers' expected service characteristics. In this phase, providers collaborate at length with their customers in the way of dialog to collect their information and globally evaluate if they can satisfy customer needs. In this phase, the referred artifacts may include 'customer', 'customer needs', 'customer wants', 'customer demands', and 'assessment'. The 'assessment' artifact is used to preliminarily evaluate if providers are competent to satisfy customer needs.

Referred artifacts: i.e., customer, customer needs, customer wants, customer demands, assessment, etc.

Referred service actors: i.e., customers, requirement Engineer, IT specialists, etc.

Establishing agreement phase (Phase 3): formally or informally records a common understanding about priorities, requirements, responsibilities, expected service characteristics, expected customer inputs, guarantees, warranties, plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. During this phase, providers and customers establish close relationships with each other.

Collaboration goal: service providers and customers reach an agreement on expected service characteristics and expected customer inputs and sign a contract that prescribes guarantees, warranties, plan time, cost, or QoS, etc.

General collaboration process: 1) finally identify and verify expected service characteristics through collaboration; 2) providers and customers negotiate with one another about the contract's details; and 3) sign a contract. In this phase, customers make their order of services from providers and pay for their orders. In addition, an 'order' artifact may include at least one item, in this sense, the artifact 'OrderItem' is used to model the ordered item in the 'order' artifact. The 'bill' artifact is used to record payment information. Another possible artifact is the 'coupon' artifact held by customers for discount.

Referred artifacts: i.e., customer artifact, customer demands, customer requirements, contract, order, OrderItem, bill, or coupon, etc.

Referred service actors: i.e., customers, requirement Engineer, IT specialists, accountant, bank, suppliers, or vendors, etc.

Designing service phase (Phase 4): in this phase, in terms of identified expected service characteristics, service providers design required services by using an appropriate combination of competencies, resources, technologies, ingenuity, skills, and knowledge to bring about specific benefits for the service consumer.

Collaboration goal: design customized service based on expected service characteristics.

General collaboration process: in order to achieve the goal of the collaboration, various service actors (business experts, IT specialists, customers, and developers) collaborate with one another to enrich business ontology, to generate business artifacts, to design business rules, to design collaboration process, and to develop web services. As described in Chapter 4, if there is any change in customer needs, this change can be propagated to customer requirements. This phase completely conforms to the proposed requirement model in Chapter 4 to integrate various service actors that adopt different models or languages.

The detailed design process includes several steps: 1) providers restructure customer demands into customer requirements described by SBVR; 2) design high-level service system ontology; 3) in terms of the obtained customer requirements, enrich service system ontology, identify artifacts (attributes, state, and lifecycle), design web services to update artifacts and performs tasks, and design business rules to invoke web services; 4) discover if existing internal business services can satisfy customer requirements--if there is no match, then take into account external business services; 5) if neither internal nor external service providers can satisfy customer requirements, then create a new service; 6) if the decision is to create a new service, two ways could be followed: either increment existing service system or create a completely new service system; 7) if the first

choice, then increment existing service and select appropriate service system partners (if necessary) and set up collaboration with them, then update the interface (if necessary) to deal with identified artifacts; and 8) If selecting the second choice, a new service system will be created from scratch. The detailed steps may include the following: a) identify service system partners (internal or external) by matching service characteristics; b) design artifacts and business rules; c) transform into implementation rules; d) update interfaces with clients and partners; and e) set up collaboration with them by exchanging artifacts.

In this phase, besides customer artifact, customer needs, customer wants, customer demands, and customer requirements, other artifacts may also be involved, such as 'schedule', 'design plan', 'design evaluation' and 'requirement changes'. The artifact 'schedule' is used to record the schedule of service design. The artifact 'design plan' is used to record detailed design information. The artifact 'design evaluation' is used to evaluate the efficiency of the current design strategy. The artifact 'requirement changes' is used to record changes to customer requirements at design time.

Referred artifacts: i.e., customer, customer needs, customer wants, customer demands, customer requirements, requirement changes, design schedule, design plan, and design evaluation.

Referred service actors: i.e., customers, requirement Engineer, IT specialists, accountants, or external service actors, etc.

Provisioning service and evaluation phase (phase 5): this phase consists of the processes used to perform the service defined in the previous phase to accomplish the customer's requirements. The service consumer collaborates and co-produces value with the service provider. This phase also observes service delivery and customer satisfaction so that potential problems can be identified in a timely manner and corrective action can be proposed to remedy the situation.

Collaboration goal: deliver the service, evaluate customer satisfaction and improve service experience based on evaluation results.

General collaboration process: 1) providers deliver services to customers, and 2) customers consume services and give their feedback to providers. In this phase, possible artifacts include the 'evaluation', 'claim', and 'refund'. The 'evaluation' artifact records customers' evaluation to service experience. The 'claim' artifact is used to represent the case in which some service actor breaks the signed contract or does not respect the terms of the contract. The 'refund' artifact is used to denote the case in which providers refund customers what is due from their claim.

Referred artifact: i.e., evaluation, claim, refund, and other specific artifacts required by the service to accomplish the service goals (i.e., order, vendor, supplier, exam, or course, etc.).

Referred service actors: i.e., customers, requirement Engineer, IT specialists, accountants, or external service actors, etc.

Papageorgiou et al. (2009) argue that a collaboration pattern should include attributes specifying what a specific collaboration is, where it is applicable and under which circumstances it may be initiated. They also presented the structure of a collaboration pattern: including name, category, problem, lifecycle phase, application area, precondition, triggers, triggers of exceptions, roles, input information, output information, duration, exception, post-condition, related collaboration patterns, and solution. In this study, we present a structure for artifact-based collaboration patterns by reusing some of these attributes, as shown in Table 5.3.

Table 5.3 Artifact-based collaboration pattern structure

Artifact-based collaboration pattern		
Field	**Description**	**Example**
Name	The name of collaboration pattern	confirm a claim
Problem/ goal	A description of the problem a collaboration expects to deal with or the goal a collaboration pattern expect to achieve	Evaluate if the customer's claim is eligible
Application domain	Declares the domain where the collaboration is applicable	Any service system
Precondition	The list of the states and conditions that must be satisfied before the execution of a collaboration pattern	Substract(Claim.date,Order.finishedDate)<=30∧Order.completed==true
triggers	Event to trigger the collaboration pattern	(Order.delayDays>=20) ∨ (OrderItem.lowQuality==true)
Triggers of exceptions	Events that can raise an exception during the execution of a collaboration pattern	Substract(Claim.date,Order.finishedDate)>30
exception	A description of an exception to the pattern (e.g. termination of the specific collaboration pattern and execution of another one).	Cancel a claim
Referred roles	Service actors that take part in the collaboration	Customers, providers, generic web service, web service
Post condition	Effects after the execution of a collaboration pattern	(Claim.confirmed==true)∧ Create a refund artifact
Referred artifacts	Record referred artifacts exchanged in the collaboration pattern	Customer, Order, claim, OrderItem, refund
Interaction pattern	Declare our proposed interaction pattern to which the referred artifacts conform	Claim artifact: Shared business artifact and jointly manage business rules between provider and customers

delivery channel for artifacts	Record information about delivery channel to exchange business artifacts	i.e., Channel for the claim artifact ==(email ∨mail∨phone)
Related collaboration patterns	Include three parts of information: 1) Collaboration patterns that optionally can be executed in parallel or after its termination. 2) Alternative collaboration patterns that can be used instead of the described one. 3) Conflicting collaboration patterns that cannot be executed concurrently with the described one.	1) subsequent collaboration pattern: Create a refund 2) alternative collaboration pattern: none 3) conflicting collaboration patterns: Refuse a claim
Solution	Generic process used to describe how the collaboration pattern to deal with its problem. 1) A customer creates a claim artifact. 2) The generic web service establishes the connection between the customer and the provider. 3) The generic web service for the claim artifact sends the claim artiafct to the provider. 4) Based on shared rules of the claim artifact, appropriate web service is invoked to evaluate if the claim is eligible. 5) If it is eligible, then confirm the claim and create a refund artifact. Else refuse the claim and send back the customer the reasons why it is refused.	

As opposed to the process-centric collaboration pattern, our collaboration pattern is data-centric or artifact-based. In other words, their approach has to take into account the inputs and outputs of collaboration patterns, whereas our structure makes professionals focus on collaboration rather than integrating business processes. In addition, we also introduce three new attributes including referred artifact, interaction pattern, and referred delivery channel(s) for artifacts.

Additionally, as shown in Figure 5.5, in KIBS firms, the participation of customers is an essential part of the service lifecycle. Customers not only collaborate to assess the situation but also participate in the service design and provisioning. They are both co-producers and consumers at the same time. Knowledgeable workers are authorized to skip and re-do activities based on the current states of their service consumers. The exchanged data is the main driver by which activities can be chosen and their sequence of execution can be scheduled during the collaboration. Decisions made for customers will be managed by business rules used to invoke computational resources (i.e., Web services) to update business artifacts and to perform tasks/actions to determine the appropriate strategy to guide collaboration, thereby finally achieving customer satisfaction.

To conclude, our proposed data-centric collaboration patterns can not only be used to represent recurred collaboration activities, but also to make business professionals focus on their businesses rather than integrating business processes. In other words, it helps to establish sustainable

business collaboration regardless of current business processes and also to improve the design efficiently by reusing collaboration patterns.

5.7 Conclusion

In this chapter, we examined some key research issues related to modelling collaboration in service systems to achieve business interoperability. Traditional approaches mainly adopted process-centric or activity-driven solutions, which do not conform to the way people do business. In addition, this process-centric approach makes it quite difficult to deal with dynamic and ad hoc collaboration in service systems.

In terms of these issues, some researchers have recently proposed to use data-centric approaches or a unified approach to model dynamic collaboration. However, due to the lack of common understanding of collaboration and restricted access to data resources, these approaches make it difficult to achieve high level business interoperability. In addition, these data-centric approaches do not discuss how to ensure privacy and security when exchanging business artifacts. Further, since in different service systems, throughout the service lifecycle, there are lots of recurred collaboration activities, they could be reused or formulated as patterns to improve design efficiency. The current data-centric approach does not takes into account recurred collaboration activities.

In this sense, in this chapter, we propose a hybrid method to integrate activity-driven business processes and a data-driven method to model ad hoc collaboration. We first presented an artifact model which consisted of self-contained business records composed of attributes, states and life cycles, to represent exchanged explicit knowledge in collaboration. Then we presented the collaboration ontology to provide a common understanding to collaboration. Each collaborative activity has a common goal which is achieved by a set of interaction activities. Each activity is represented by a directed data flow with its major contents of business artifacts. Also, as opposed to the traditional method for interoperability by composing various Web services, we introduce a 'generic' web service' to be limited to establishing a connection between service providers and customers. Additionally, we introduce business rules to manage dynamic decision making in collaboration. In other words, business rules are used to invoke appropriate web services to update artifact attributes and transit from its current state to another state. In addition, we presented a structure for artifact-based collaboration patterns, which could be reused in different projects throughout the service lifecycle to improve design efficiency.

Chapter 6

Case Study: Designing e-Learning Service System

Abstract: In this chapter, we present a case study in the e-learning domain to explain how to use our proposed middle-out methodology and create an e-learning service system driven by requirements and collaboration. Our case study takes into account two situations: 1) the need to design a new service system based on customer requirements, and 2) to extend the current service system by adding or updating service characteristics to supporting requirement changes.

6.1 Introduction

In Chapter 3, we presented the service system design framework and also discussed the general process to create a service system. That general process is mainly composed of eight steps: 1) design generic ontology; 2) design domain ontology; 3) concretize customer requirements and represent service characteristics and expected service characteristics; 4) evaluate if the current system or external system can satisfy expected service characteristics; 5) identify service actors by matching service characteristics and business artifacts; 6) enrich specific ontology and update knowledge base the business rule base; 7) set up collaboration; and 8) adapt to dynamic requirements.

In order to further explain how to use our presented middle-out methodology based on reference model, requirement model, and collaboration model in order to create an intelligent IT-enabled service system, we provide a case study in the e-learning domain. In this case study, we expect to demonstrate how to refer to the reference model to create a service sys-

tem; how to concretize customer requirements, how service actors from multiple disciplines collaborate with each other; how ontology is used to describe this domain, and how to enrich service system ontology and to generate business rules and business artifacts based on structured and precise customer requirements. The case study also shows how to choose delivery channels to exchange business artifacts; how the e-learning service system adapts to new customer needs; and how the e-learning service system provides more value to each service actors compared to traditional e-learning systems.

We distinguish two ways to create an e-learning service system:

1- Design a new service system from scratch based on customer requirements.

2- Adapt the existing e-learning service system to dynamic requirements.

In this chapter, we will first discuss the first way to create a new service system based on customer needs and then in terms of the designed service system, we will take into account the second way by adding new service characteristics to extend current service system to illustrate the system's intelligence.

6.2 Creating an New E-learning Service System

The first step for designers to create a service system is to refer to our proposed reference model to formalize various kinds of knowledge from different domains to enable collaboration. In the following, we elaborate how to design a new e-learning service system from scratch to satisfy customer needs based on the Needs.

Needs: a group of professional people express their need for 'flexible and cost-efficient online e-learning services to improve their English language'.

Wants: based on the above needs, requirement engineers use the *i**model to concretize needs into a set of service characteristics. As mentioned before, we first assume that an e-learning service system will be created from scratch. In this sense, in the beginning of the design process, there are no existing internal service system components. Service providers will thus begin with identifying possible service characteristics. Assume that they preliminarily identify the following potential service characteristics for the professional people needs:

> high-level teachers, freely access online education resources, discoverable online education resources, large quantities of trusted and authoritative education resources, the freedom to adapt, and develop online education resources, flexible teaching delivery, group-oriented e-learning, multiple online interactions, the availability of social learning tools, multiple teaching methods (i.e., first and foremost lectures, con-

tent-based learning through individual work and group work, Guest lecturers, Case studies, Exercises and group discussions, etc), study in entertainment, predominance of an independent learning style, flexible study time and location, flexible schedule, linking theory with practice, online Q&A, rating Teachers, Comment Classes, Online self-test support, online Exam, online quiz, and flexible payment, …

Service providers will then specify these service characteristics with a set of internal characteristics using the *i**model. After that, providers and customers will negotiate with one another to set requirements on well-identified internal characteristics. The corresponding internal characteristics for each of these service characteristics are listed in Table 6.1.

Table 6.1 The specification of service characteristics and corresponding service system components

Service characteristic	Internal service system components
High-level teachers	Teacher resource
Freely access online education resources	Access _Resources technology
Discoverable online education resources	Search_Engine technology
large quantities of trusted and authoritative education resources	Scholor course resource && Open course resource || external resources
the freedom to adapt, and develop online education resources,	Access_Resource Technology && Adapt_Resource technology && Distribute_Resource Technology
flexible teaching delivery	Video channel && Audio channe l&& Text channel
group-oriented e-learning	Teacher resource && Audio_Conference technology|| Video_Conference technology
multiple online interactions	Audio_Conference Technology && Video_Conference Technology && Discussion_Board Technology && Social tool Resources && E_emal technology && Bulletin Technology
the availability of social learning tools	MicroBlog && Blog
multiple teaching methods	Teaching_Via_Lectures technology && Teaching_Via_Individual_Work Technology && Teaching_Via_Group_Work technology && Teaching_Via_Guest_Lecturers technology && Teaching_Via_ Case_Studies technology && Teahing_Via_Project technology
study in entertainment	Video_Game resource || OnlineRobot technology && SelfTestViaOnlineGame technology
flexible study time and location	Online_Course Resources && Internet channel || Mobile Channel
flexible schedule	TeachingPlan resource && Email Channel ||

	DiscussBoard channel
linking theory with practice	Lecture resource && Project resource
online Q&A,	Q&A Technology
rating Teachers	Rating_Teacher technology
Comment Classes	Comment_Class technology
Online self-test support	Excercises resource && Online_Test technology
online Exam	Exam resource && Online_Exam technology
flexible payment	Payment_Via_CreditCard technology

For example, the service characteristic 'FreelyAccessOnlineEducationResources' can be satisfied by the 'AccessResource' technology that is further specified by a set of features and constraints, as shown in Figure 6.1. The 'Access resource' technology is measured by five features: respectively 'availability'; 'accessTime'; 'accessLocation'; 'accessChannel'; and 'authority' that can be further specified into four features: including 'readAuthority' and 'updateAuthority'.

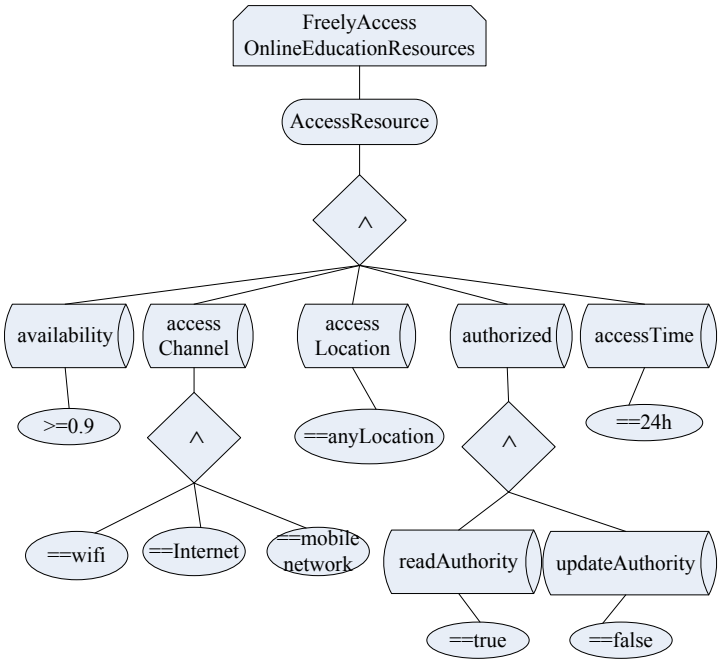


Figure 6.1 Specification to the service characteristic 'freely access online education resources'

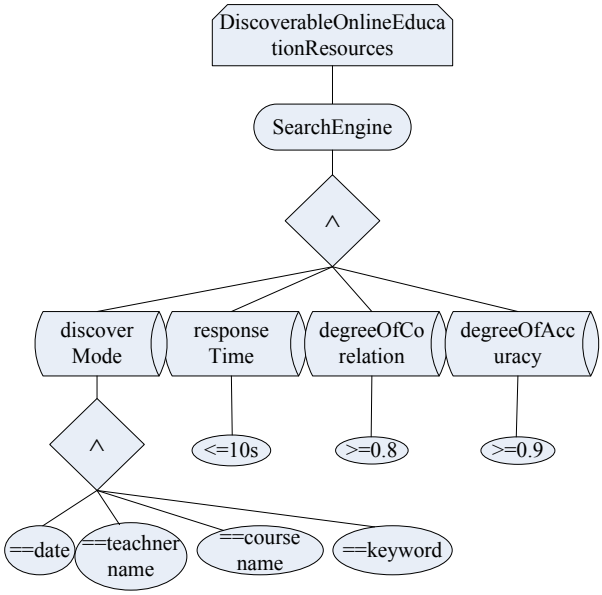


Figure 6.2 Specification to the service characteristic 'discoverable online education resource'

The specification of the service characteristic 'DiscoverableOnlineEducationResources' is shown in Figure 6.2. Another example for the specification to the service characteristic 'study in entertainment' is shown in Figure 6.3.

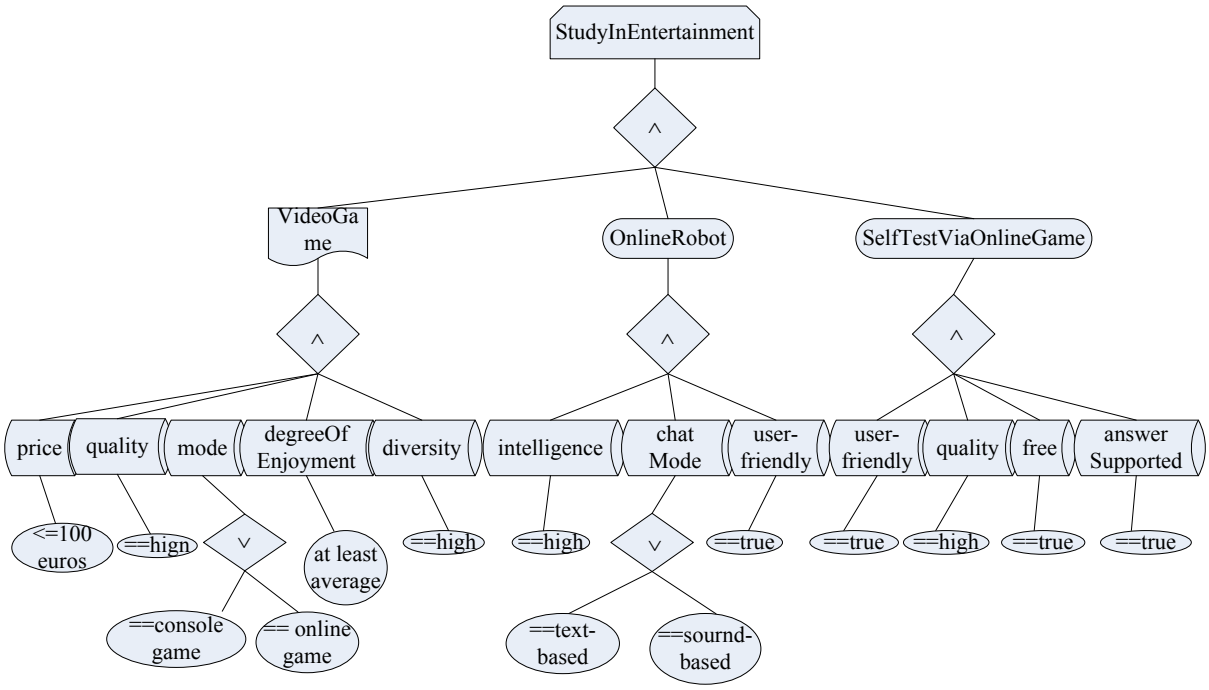


Figure 6.3 Specification of the service characteristic 'study in entertainment

Demands: as mentioned before, the graphical representation of customer wants requires semantics, which may lead to different interpretations. To this end, we present in a formal structured language, namely SRML, to transform the graphical customer wants into structured and clear customer demands. For example, the graphical representation shown in Figure 6.1 can be interpreted SRML expressions below:

```
Type AccessResourceType= TechnologyCharacteristics
{
accessTime: numeric hours;
accessLocation:string;
accessChannel: set{Internet, wifi, Mobile network};
readAuthrity: boolean;
updateAuthority: boolean;
availability: numeric;
};
FreelyAcessOnlineEducationResource_Service = serviceCharacteristics{
AccessResourceType{};
};
AccessResource= AccessResourceType expectedCharacterisics
{ accessTime==24hours;
accessLocation== unlimited;
accessChannel==set{Computer, wifi, Mobile network};
readAuthority==true;
updateAuthority==false;
availability>0.9;
};
FreelyAcessOnlineEducationResource_ExpectedService = expectedSer-
viceCharacteristics{
AccessResource{…};
};
```

Correspondingly, the graphical representation for the service characteristic ‘discoverable online education resource’ shown in Figure 6.2 can be denoted as the following SRML expressions:

```
Type SearchResourceType= TechnologyCharacteristics{
discoverMode: set{keyword ,teacher name, course name, date};
responsTime: numeric s;
authorized: Boolean;
degreeOfCorelevance: numeric;
degreeOfAccuracy: numeric;
};
DiscoverableOnlineEducationResource_Service= serviceCharacteristics{
SearchEngineType{};
};
SearchResource = SearchResourceType expectedCharacteristics{
discoverMode==set {keyword ,teacher name, course name, date}
responseTime<=10s;
authorized==true;
degreeOfCorelavance>=0.8;
degreeOfAccuracy>=0.8;
};
DiscoverableOnlineEducationRe-
source_ExpectedService=expectedServiceCharacteristics{
SearchEngine{};
};
```

Similarly, the graphical representation for the service characteristic 'study in entertainment' shown in Figure 6.3 can be denoted as the following SRML expressions:

Type VideoGameType= **ResourceCharacteristics**{
Price: numeric euro;
Quatiy: increasing enum{High, average, low} with order {high>average>low};
Mode: enum{console game, online game};
Degree of enjoyment: increasing enum{High, average, low} with order {high>average>low};
Diversity: increasing enum{High, average, low} with order {high>average>low};
};
VideoGame=VideoGameType **expectedCharacteristics**{
Price<=100 euros;
Quality ==high;
Mode = =online game || console game;
Degree of enjoyment >=average;
Diversity==high;
};
Type SelfTestViaOnlineGameType = **ResourceCharacteristics**{
free: Boolean;
Quality: increasing enum{High, average, low} with order {high>average>low};
User-friendly: Boolean;
Answer-supported: Boolean;
};
SelfTestViaOnlineGame=SelfTestViaOnlineGameType **expectedCharacteristics**{
Free==true;
Quality==high;
User-friendly==true;
Answer-supported==true;
};
Type OnlineRobotType = **TechnologyCharacteristics**{
intelligence: increasing enum{High, average, low} with order {high>average>low};
chat mode: set{text-based, sound-based};
User-friendly: Boolean;
};
OnlineRobot=OnlineRobotType **expectedCharacteristics**{
Intelligence == high;
chat mode== text-based || sound-based
User-friendly: Boolean;
};

Requirements: as mentioned before, SRML is quite quite effective to describe structured service characteristics, but it is not expressive enough to describe customer requirements and it is too technical to be understood by non-professional people. SBVR can be used to connect customers, business experts and technical people. In this study, SBVR is used to describe specified customer requirements that are composed of business vocabularies and business rules relateted to service characteristics.

For example, the snapshot of SBVR-based customer requirements related to the service characteristic 'FreelyAccessOnlineEducationResources' is:

> It is necessary that a student can access online education resources everyday. It is obligatory that the accessTime of online education resources is 24 hours. The accessLocation should be unlimited. Students can access to the online education resources via the accessChannel of Internet and Mobile network. If a student accesses to online multimedia resources via Internet, the bandwidth of Internet is at least 20Mbits/s, else the bandwidth is at least 10Mbits/s. The format of online resource can be multimedia and text. It is possible that the availability of an education resource is at least 90%. If the student is registered, then the student has readAuthority and he can freely access online resources. If the updateAuthority of an online education resource is allowed, then students can not update the online education resource…

Identify service actors: based on SBVR requirements, designers at first evaluate if current service systems or external service systems satisfy the obtained expected service characteristics by comparing their respective service characteristics and the expected service characteristics.

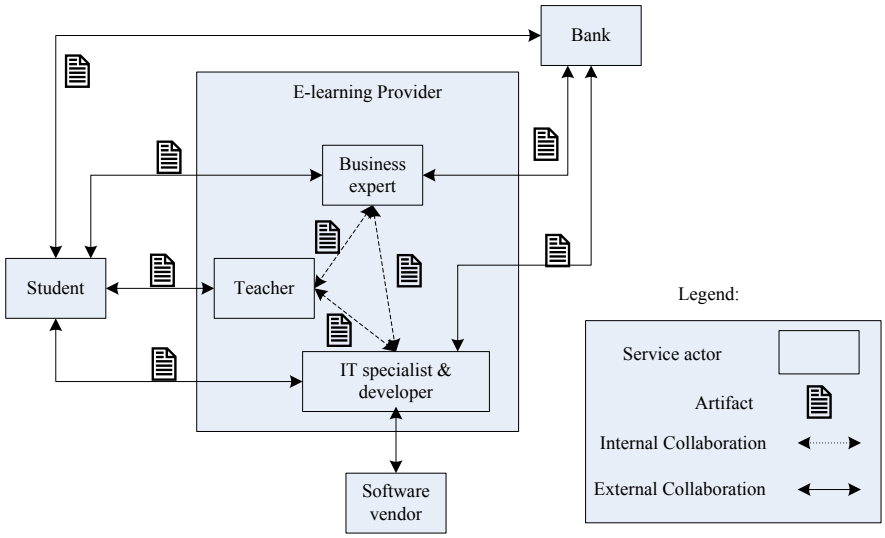


Figure 6.4 Value/collaboration network of an e-learning service system (1)

Herein, we assume that designers have to design a new service system. The following service actors could be identified to satisfy the internal service components listed in Table 6.1: internal employees for service providers (i.e., teachers, business experts, IT specialists, software and developers), customers, software vendors, and banks. Their collaboration network is shown in Figure 6.4, where IT specialists mainly take charge of designing the service system ontology, developing business rules and artifacts. Businesses experts mainly focus on formulating business strategy, identifying and concretizing customer requirements. Software developers take part in implementing service systems. Teachers provide courses and pedagogical services for students. Software vendors sell software or appli-

cations to service providers. Banks collaborate with the service providers to facilitate students' payments.

Enrich service system ontology: after identifying possible service actors, designers have to design an ontology to describe the e-learning domain based on customer requirements and the high-level service system ontology.

Since a service system is mainly composed of service actors, resources, technologies, competencies, and delivery channels, the service system domain ontology to be designed will mainly enrich these components.

In this example, designers may identify the following concepts: 'EducationResources', 'FreelyAccessOnlineEducationResources', 'AccessChannel', 'Internet', 'Wifi', and 'MobileNetwork', where the concept 'Education_Resources' belongs to the category of resource, the concept 'FreelyAccessOnlineEducationResources' is a technology, and the concepts 'AccessChannel', 'Internet', 'Wifi', and 'Mobile_Network' are delivery channels.

The possible object property is 'hasAccessChannel'. Possible data properties include 'hasAccessTime', 'hasAccessLocation', 'hasReadAuthority', 'hasUpdateAuthority', and 'hasAvailability'. The possible instances in this example may include 'Internet', 'Wifi', and 'MobileNetwork'. The axioms or business rules include:

1- FreelyAccessOnlineEducationResources is a subclass of technology characteristics.

2- If a student has readAuthority to an education resource, then the student can access the online educational resource.

Generate business artifacts and business rules: as mentioned before, the granularity of an artifact is quite important. If it is too large, it may lose some levels or aspects of information to describe business logics. On the contrary, it will refer to too many details thereby leading to increasing complexity and reducing efficiency. Also as mentioned before, due to the limited expressive power of the collection of OWL 2 and SWRL, some SBVR expressions cannot be denoted by them (for example, the negation of properties). In this sense, this phase requires the manual work of business people to identify the granularity of business artifacts and related business rules.

In this example, we manually identify the following artifacts: 'Visitor', 'Customer', 'CourseList', 'Course', 'Bill', 'TeachingPlan', 'QuizPerClass', 'ExamResult', 'FinalExam', 'Evaluation' and 'Certificate' based on service characteristics listed in Table 6.1. The 'Visitor' artifact is used to record information concerning anonymous visitors that could be potential customers. The 'Customer' artifact is use to record a customer's col-

laboration activities with other service actors. The 'CourseList' is used to represent the process of selecting courses. The 'Course' artifact represents the course items in a 'CourseList' artifact. The 'Bill' artifact represents business records about the payment process. The artifact 'TeachingPlan' denotes the process of updating the teaching plan to adapt to the progress made by students. The artifact 'QuizPerClass' is used to describe the process of doing a quiz. The artifact 'ExamResult' is used to represent the process from finishing an exam to obtain the result of exams. The artifact 'FinalExam' is introduced to evoke final exams in traditional learning. The 'Evaluation' artifact is used to record students' comments on teachers or courses. The 'Certificate' artifact is used to represent that fact that students end their courses in a satisfactory way.

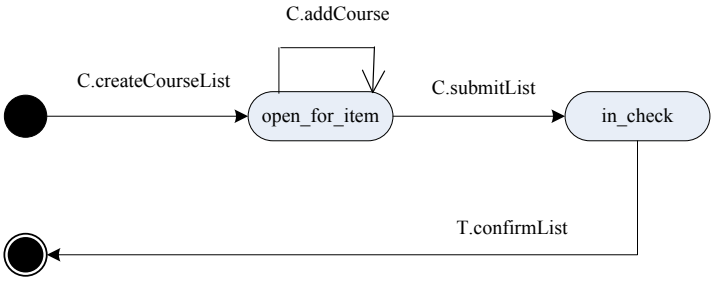
For example, the collaboration between students and teachers may refer to the following artifacts: 'Customer', 'Course', 'TeachingPlan', 'QuizPerClass', 'FinalExam', 'ExamResult', 'FinalExam', and 'Evaluation'.

Herein, we present two artifacts as examples: including the 'CourseList' artifact (See Table 6.2) and the 'Course' artifact (See Table 6.3).

As shown in Table 6.2, a student first creates a 'CourseList' artifact and then the student begins to add courses to the 'CourseList' artifact. The service provider has to guarantee the availability of the selected courses for the student. If some courses are not available, the service provider has to notify the student.

Table 6.2 Example of 'CourseList' artifact

CourseList artifact:
Attributes: CouseListID, StudentID, CourseNum
States: init, open_for_item, in_check, completed
Lifecycle:



Legend: C: student T: teacher

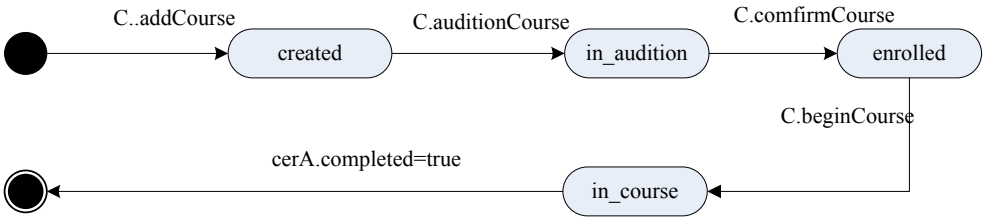
Business rules: e.g.,
Event(?Var_Event_AddCourse), hasEventID(?Var_Event_AddCourse, "AddCourse"), hasEventValue(?Var_Event_ AddCourse,true),
CourseListArtifact(?Var_CourseList), hasState(?Var_CourseList, "open_for_item_CourseList"), hasCourseNum(?Var_CourseList, ?courseNum)
->Add(?courseNum, ?courseNum, 1)

As shown in Table 6.3, a student first creates a course artifact and adds it to the CourseList if the course is available. Then the student can decide if he/she will enrol to the course. If the student decides to enrol the

course, he/she will pay for it and start the course. The 'Course' artifact will not be complete until the student obtains the certificate for the course.

Table 6.3 Example of 'Course' artifact

Course artifact:
Attributes: CoureListID, courseID, teacherID, hours, startDate, finishDate
States: init, created, in_audition, enrolled,in_course, completed
Lifecycle:



Legend: cerA: certificate artifact C: student

Business rules: e.g.,
Event(?Var_Event_AuditionCourse), hasEventID(?Var_Event_AuditionCourse, "AuditionCourse"), hasEventValue(?Var_Event_ AuditionCourse,true),
CourseArtifact(?Var_Course), hasState(?Var_Course, "Created_Course")
->
hasState(?Var_Course, "in_enrolled _Course")

After identifying artifacts, IT specialists have to collaborate with developers, business experts, and customers to design the lifecycle for each of them and to develop business rules to invoke appropriate web services to update them.

Set up collaboration: as opposed to the traditional approach to describe various business activities as web services and then compose them together to execute business process, the artifact-driven approach uses business artifacts to represent exchanged explicit business knowledge in collaboration and introduce a generic web service for each artifact to set up collaboration and to send or receive artifacts. Before collaboration, service actors have to reach an agreement to share artifacts and business rules to jointly contribute towards their lifecycle. In the runtime, when a generic web service receives an artifact, it will invoke the inference engine that reads shared business rules and reasons on them in terms of the artifact's current state to invoke appropriate web services to update the artifact. Under the traditional approach, various service actors may use different concepts to describe the same inputs or outputs of web services, which makes it difficult to achieve semantic interoperability when composing them together. Our approach frees designers from complex web service composition by matching the output of a web service to the expected input of another web service and helps to achieve semantic interoperability.

Since business artifacts could be saved in the format of XML files, or in databases, it is important to choose appropriate delivery channels to exchange them. When exchanging an artifact, a customer should not care about the internal structure of the service providers or how the service pro-

viders mobilize their internal resources, competencies, technologies, and delivery channels to provide services. The customer only requires guidelines from the service providers about what to do in the next step.

6.3 Propagate Requirement Changes

As discussed before, our approach supports the propagation of changes due to customer requirements from a high level (i.e., customer level and business level) to a low level (i.e., technical level). When customer requirements change, various service actors follow the requirement model to make corresponding changes to low-level service system components (such as introducing/developing new service system components, or to decompose current components or recomposing components), which may lead to the introduction of new service actors, new business artifacts, and new business rules. Herein, we intend to explain how our approach propagates requirement changes through this case study.

Requirement changes could occur in original customer needs, service characteristics, or features or constraints of service characteristics. For example, changes to customer needs may be 'an effective and cost-efficient e-learning service that supports learning through mobile networks'. Customers can also add new service characteristics or update current service characteristics (i.e., supports the use of external open course resources) in terms of original customer needs. They can reset the value of constraints (i.e., the frequency of exams and the level of teachers) as well.

6.3.1 Requirement Changes Due to External Services

In the case that students expect to attend a special English course that is not available in current e-learning services but it can be provided by external e-learning providers (i.e., MIT Open Course University) (MIT 2011), the e-learning provider first look for external service providers with respect service characteristics.

Assuming that the MIT Open Course University is identified as the service partner, the service provide will collaborate with it to include its freely access to its special English course, which could provide value for all actors involved. From the perspective of students, they can easily attend the special course as they attend courses from the current system neglecting how the service provider collaborates with the external service provider to integrate external courses into existing courses. In other words, in order to realize these two service characteristics for customers, the service provider and the external e-learning provider have to be bundled together to construct a value network. From the perspective of the current service pro-

vider, a new solution is found to satisfy the customer's new needs and obtain their satisfaction.

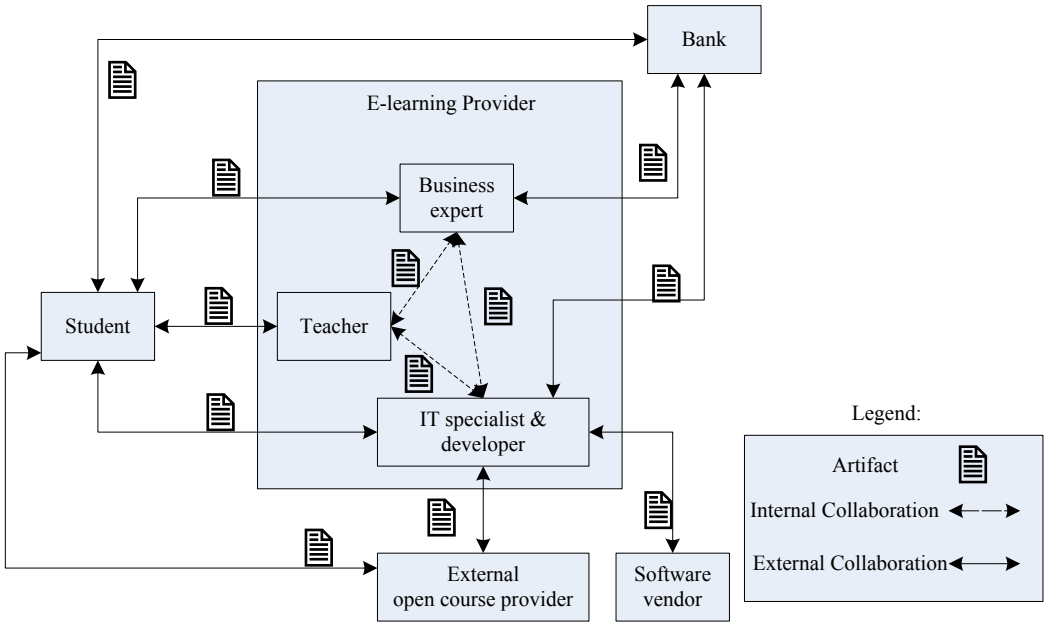


Figure 6.5 Value/collaboration network of an e-learning service system (2)

As for the external service provider, benefits are also gained by offering quality courses and supporting their dessimantion policy. The collaboration network is shown in Figure 6.5.

The introduction of courses from external e-learning providers will inevitably lead to the updating of some existing service characteristics, such as 'discoverable online education resources', 'freely access online education resources, and 'download online education resources'.

Business experts, IT specialists, developers and customers then collaborate with one another to specify these three additional requirements with the *i**model; As shown in Figure 6.6, the service provider introduces large quantities of external educational resources and re-mobilize/re-compose internal service system components including internal course resources, external course resources and access resource technology, to satisfy the new service characteristic 'freely access internal and external education resources'.

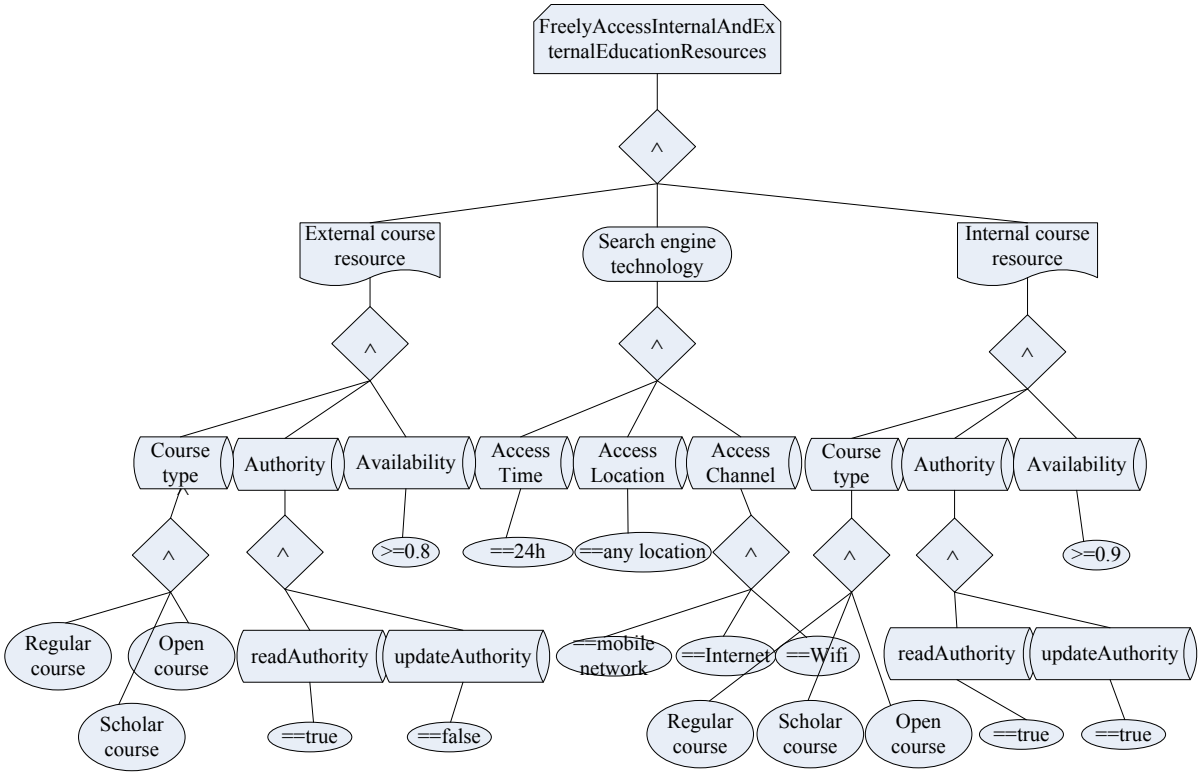


Figure 6.6 The specification for 'freely access online internal and external education resources'

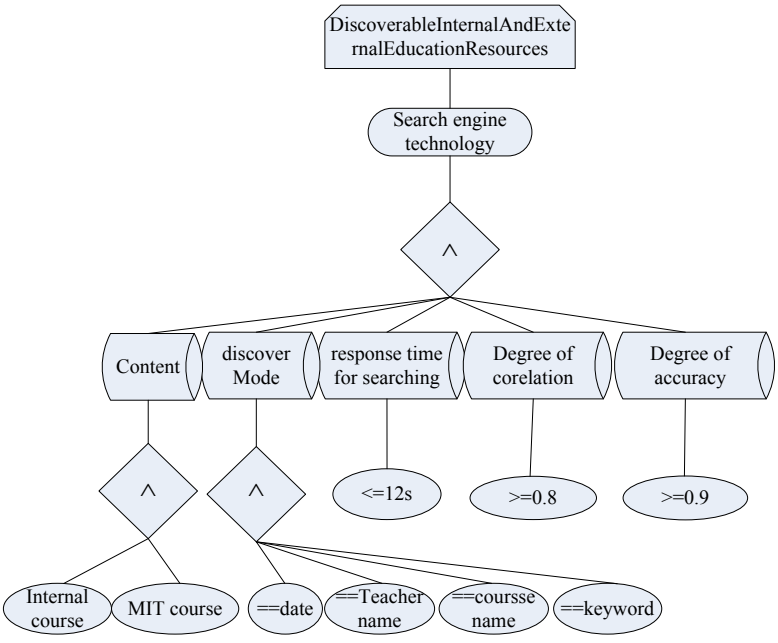


Figure 6.7 Discoverable internal and external education resources'

As shown in Figure 6.7, due to the increase of education resources (MIT Open Courses), current Search Engine technology has to be extended to support the search of both internal resources and external MIT Open Courses.

As shown in Figure 6.8, different price policies are formulated to download internal and external education resources. The price unit for downloading external education resources is no more than 2 credits/100Mbits, which is different from that for downloading internal education resources (<=1 credit/100Mbits).

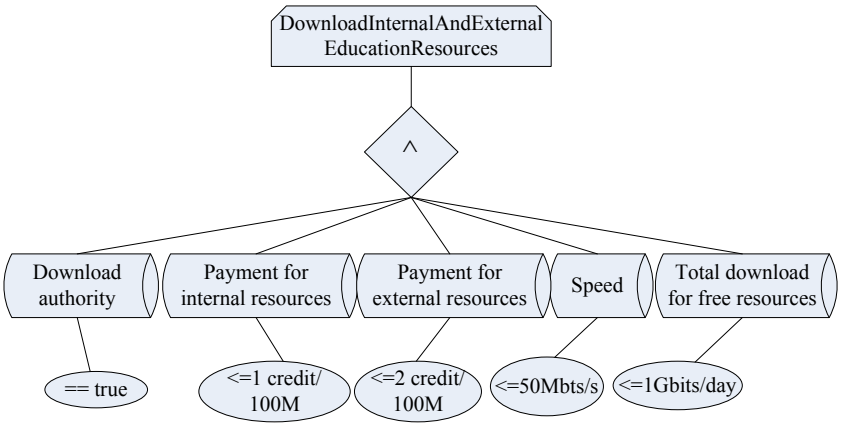


Figure 6.8 Download internal and external education resources

Correspondingly, the new service characteristics specified in the *i**model will lead to new SRML expressions. For example, the SRML expressions for the new service characteristic 'FreelyAccessOnlineInternalAndExternalResources' can be described below:

```
Type SearchEngineType= TechnologyCharacteristics{
accessTime: numeric hours;
accessLocation:string;
accessChannel: set{Internet, wifi, Mobile network};
};
Type InteralCourseType= ResourceCharacteristics{
courseType: set{Regular Course, Scholar Course, Open Course};
readAuthority: boolean;
updateAuthority: boolean;
availability: numeric;
};
Type ExternalCourseType= ResourceCharacteristics{
courseType: set{Regular Course, Scholar Course, Open Course};
readAuthority: boolean;
updateAuthority: boolean;
availability: numeric;
};
Freely_Access_Online_Internal_And_External_Education_Resources_
Service = serviceCharacteristics{
SearchEngineType{}; &&
```

InternalCourseType{};&&
ExternalCourseType{};
};
SearchEngine= SearchEngineType **expectedCharacterisics**{
accessTime==24hours;
accessLocation== any location;
accessChannel==Computer && wifi && Mobile network;
};
InternalCourse = InternalCourseType **expectedCharacteristics**{
readAuthority==true;
updateAuthority==true;
availability>=0.9;
};
ExternalCourse = ExternalCourseType **expectedCharacteristics**{
readAuthority==true;
updateAuthority==false;
availability>=0.9;
};
Freely_Access_Online_Internal_And_External_Education_Resources_
Expected_Service= **expectedServiceCharacteristics**{
SearchEngine{}; &&
InternalCourse{};&&
ExternalCourse{};
};

Service providers can reconstruct expressions for the service characteristic 'FreelyAccessOnlineInternalAndExternalResources' into SBVR expressions on the basis of the new SRML, as shown below:

> It is necessary that a student can access online education resources everyday. It is obligatory that the accessTime of online education resources is 24 hours. It is necessary that the accessLocation is unlimited. Students can access to the online education resources via the accessChannel of Internet and Mobile network. If a student accesses to online multimedia resources via Internet, the bandwidth of Internet is at least 20Mbits/s, else the bandwidth of Internet is at least 10Mbits/s. The format of online resource can be multimedia or text. It is necessary that If a student requires an education resource that is from external e-learning service system, it is possible that the current e-learning service system provides the external online resource to the student. It is possible that the availability of an internal education resource is at least 90%. Student should have two types of internal courses: regular course and scholar course. If a student is registered, then the student has readAuthority and updateAuthority and he can freely access or update any internal online resource. It is possible that the availability of an external education resource is at least 80%. If a student is registered, then the student has readAuthority of external education resources and he can freely access any internal online resource. If the updateAuthority of an external online education resource is false, then students can not update the online education resource.

According to the new SBVR expressions, IT specialists can identify new concepts (i.e., InternalCourse and ExternalCourse), new object properties (i.e., hasInternalCourse and hasExternalCourse), new data properties (i.e., readAuthorityForExternalResource and updateAuthorityForExternalResources), new instances (i.e., the special English course), and new class assertions (i.e., the data value of the updateAuthorityForExternalRe-

sources feature) to enrich high level service system ontology. For example, specialists can also identify the following new rules:

1- If a course is an internal resource, then students can read and update the course.

2- If a course is an external resource, then students can read the course.

3- If a course is an external resource, then students cannot update the course.

Originally, the 'CourseList' (see Table 6.2) and the 'Course' artifacts (see Table 6.3) were used to represent the collaboration between current e-learning provider and students to facilitate the selection of courses by students. Now due to the changes to customer requirements and the introduction of the external e-learning provider, these two artifacts have to be adjusted. In this new situation, if a course in the 'CourseList' is not available in current system, the service provider has to introduce external e-learning providers who offer the course. In this sense, the three kinds of service actors construct a value network to satisfy the student's requirements (see Figure 6.5).

The student creates a 'CourseList' artifact to select courses and add new courses into the 'CourseList'. If all the courses are available in current e-learning system, the student can directly enrol in them and attend the courses. If one of the courses is not available, the provider can help the student to look for external e-learning service providers that provide the course and identify the most appropriate one from them by matching the student's expected service characteristics to the exposed service characteristics by these external providers. The service provider, the student and the identified external provider will then collaborate with one another to satisfy the student's new requirements for value co-creation.

6.3.2 Requirement Changes Due New Features

Taking another situation, the student also expects two additional service characteristics from the e-learning service system: 1) to provide shortcut for buying books for courses from external e-commerce providers (such as Amazon and Ebay), and 2) to support the ordering and comparison of books from different e-commerce providers by prices and public opinion.

Therefore, the e-learning service provider has to collaborate with new service actors such as e-commerce providers, suppliers, and shippers. External e-commerce providers, for instance, are integrated into the e-learning service system to facilitate students' purchase of study materials. Suppliers supply tangible products for external e-commerce providers.

Shippers ship ordered products to students. Their collaboration network is illustrated in Figure 6.9.

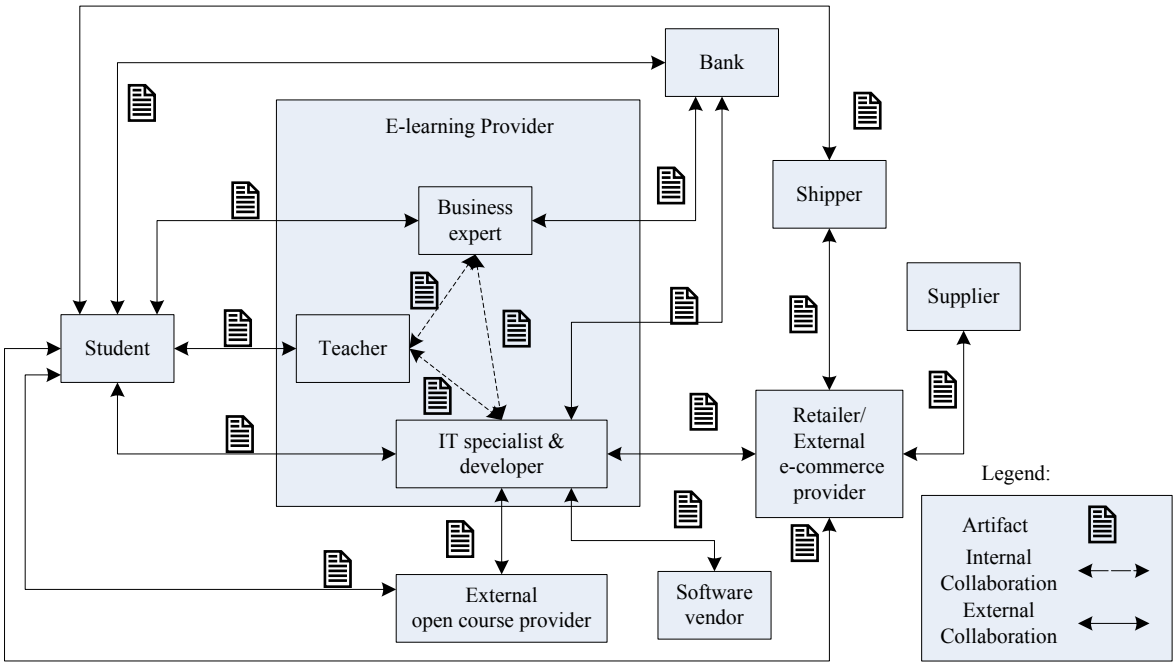


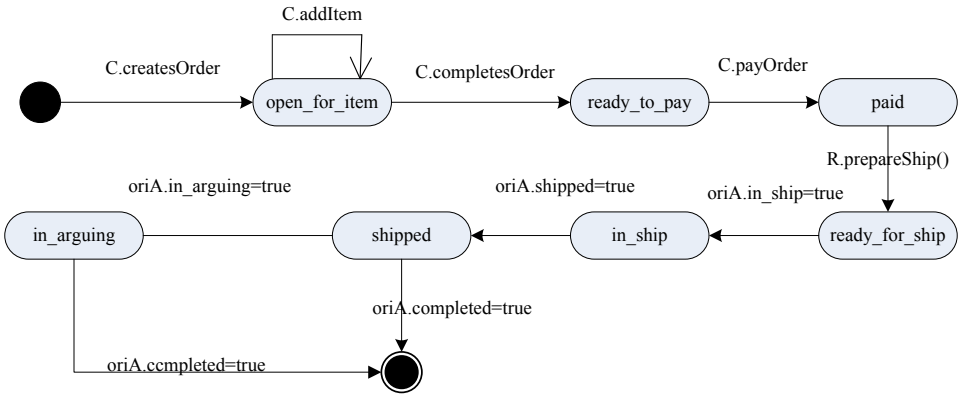
Figure 6.9 Value/collaboration network of an e-learning service system (3)

We are not expected to repeat the process of concretizing customer requirements, identifying service actors, and enriching service system ontology. We directly discuss the possible seven additional business artifacts based on this new customer requirement: respectively 'Order', 'OrderItem', 'Stock', 'Shipment', 'ReturnedItem', and 'Bill'. The 'Order' artifact is used to record the process of a student's purchase of study materials. The 'OrderItem' artifact is used to describe item products in an 'Order' artifact. The 'ReturnedItem' artifact is used to describe the process of returning items that are ordered beforehand. The 'Stock' artifact is used to record the stock information for ordered products. The 'Shipment' artifact is used to describe the process of shipping products to buyers. The 'Bill' artifact represents business records about the payment process. We simply present here the 'Order' artifact as an example shown in Table 6.4. Additional details about other artifacts (attributes, states, lifecycle, and business rules) can be found in Appendix B.

As shown in Table 6.4, these two examples for business rules are used to calculate discount information for the 'Order' artifact. In practice, one possible solution to calculate the discount information can be realized as Web services and in the run time, they are invoked by business rules.

Table 6.4 Example of 'Order' artifact

Order artifact:
Attributes: OrderID, CustID, Bid, ScheduleDate, Discount
States: init, open_for_item, ready_for_ship, paid, ready_for_ship, in_ship, shipped, in_arguing, completed
Lifecycle:



Legend: C: student R: retailer oriA: OrderItem

Business rules : e.g.,
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, "ready_to_pay"), hasCustCredit(?Var_StudentArt, ?Var_CustCredit),
greaterThanOrEqual(?Var_CustCredit, 2500)
->
hasDiscount(?Var_Order, 0.10)

In addition, based on the lifecycle model for the 'Order' artifact, a student first creates an 'Order' artifact and then adds products to the 'Order' artifact if they are available until the student finishes to adding new products to the 'Order' artifact. Before a new product is successfully added to the 'Order' artifact, the e-commerce provider has to create a 'Stock' artifact to evaluate if the ordered product is available. If the product is in stock, then it can be added to the 'Order' artifact. If not, the provider must ask the suppliers to add the product to the 'Stock' artifact.

When the student finishes his/her shopping, the student will pay for the ordered products and wait for their delivery. The retailers of the e-commerce providers will create a shipment artifact to collaborate with the shippers to ship those ordered products if the 'Bill' artifact and the 'Order' artifact arepaid. When receiving those products, if no quality problem arises, the student will complete the 'Order' artifact. If there is any quality concern regarding one or more products or if the student wants to return one or more products, the student will create a 'ReturnItem' artifact to ask for returning these products.

In this case, the collaboration activities among various service actors are represented by the exchange of these business artifacts. These service actors can agree on their exchanged business artifacts and shared business rules before their collaboration for semantic interoperability.

6.4 Conclusion

In this chapter, we presented a case study in the e-learning domain to illustrate how to use our proposed models and approaches to create an e-learning service system that takes into account requirement changes and achieves semantic interoperability.

As illustrated in the case study, our models and approaches support the creation of a new service system and the extension of current service systems by adding or updating service characteristics. Requirement changes are satisfied by remobilizing various service system components.

In designing a service system, unlike the traditional approach which mainly emphasizes collaboration between designers and customers and focuses on composing of web services or applications to model business processes in terms of customer requirements, our approach integrates various service actors from multiple disciplines by providing a more general description of business knowledge with ontologies to deal with their conceptual ambiguities. Our approach also takes into account multiple perspectives to concretize unstructured, unclear and imprecise customer needs into clear and precise customer requirements and describes it with SBVR that are understandable by different service actors due to its natural language like interface. SBVR facilitates the collaboration of both high-level strategic service actors and low-level technical service actors.

Our artifact-driven approach helps to address the issue of semantic interoperability encountered by traditional integration approaches in software system design or information system design that focus on implementing business activities in business processes as Web services and then on composing various Web services together to execute them.

Chapter 7

Technical Architecture and Experiment

Abstract: In this chapter, we present a technical architecture to implement IT-enabled service systems in terms of our proposed models and approaches and we also develop a prototype to simulate the decision makings based on business artifacts.

7.1 Introduction

In Chapter 3, we have presented a design framework and identified the design process to create IT-enabled service systems. A system architecture is generally viewed as the conceptual model to provide different views to describe a system, such as its static system components, structure, dynamic behaviours, common understanding of concepts and business knowledge in systems. In this sense, our framework and the design process support the system architecture for designing IT-enabled service systems. We argue that the service system architecture can reifies the reference model and guides different service actors from different disciplines to create a service system driven by customer requirements at design time and guide their collaboration in the run time.

However, it is not enough for technical specialists, especially system implementers to technically implement an IT-enabled service system. To this end, in this chapter, we provide a technical architecture for techni-

cal specialists to implement IT-enabled service systems in terms of our proposed models and approaches. The technical architecture presents implementation details about software components, hardware components, and how they are connected and exchange data to provide services.

7.2 Technical Architecture

In terms of our models and approaches, the proposed technical architecture should implement the following utilities: ontology development, requirement specification and representation, service characteristics matching for selecting service partners, specific ontology generation and enrichment, business artifact generation, the development of a knowledge base and business rule base, merging various different ontologies into one composite ontology, reasoning based on established knowledge-base and rule-base, detecting events from service actors' decision-made activities, providing guidelines for service actors' decision makings.

Conforming to our proposed service system design framework, the proposed technical architecture is illustrated in Figure 7.1 and is mainly composed of the following modules: 1) ontology developer module; 2) requirement specification module; 3) artifact module; 4) service characteristic registry; 5) ontology merger module; 6) decision-make module; and 7) Graphical User Interface.

During the completion of this thesis, we have implemented some modules to demonstrate the feasibility of the proposed IT-enabled service system architecture. Roughly speaking, we have implemented the requirement generation component in the artifact module in order to map SBVR requirements to OWL ontology by introducing an intermediary, namely an APE engine. The ACE Parsing Engine (APE) is a parser to transform ACE text to various kinds of formal logic languages, such as first-order logic, OWL and SWRL (APE Help 2011). The basic logic is to at first rewrite SBVR with ACE text and then to use the APE engine to implement the mapping from ACE text to OWL ontology. We also have developed a prototype to simulate data-driven collaboration based on ontology, artifacts and business rules. In this prototype, we automatically implement the collaboration platform in Figure 7.1, which mainly refers to three modules: the ontology developer module, the ontology merger module, and the decision-made module. As for the other two modules (the requirement specification module and service characteristic registry), albeit unimplemented in our prototype, we provide some guidelines about their implementations.

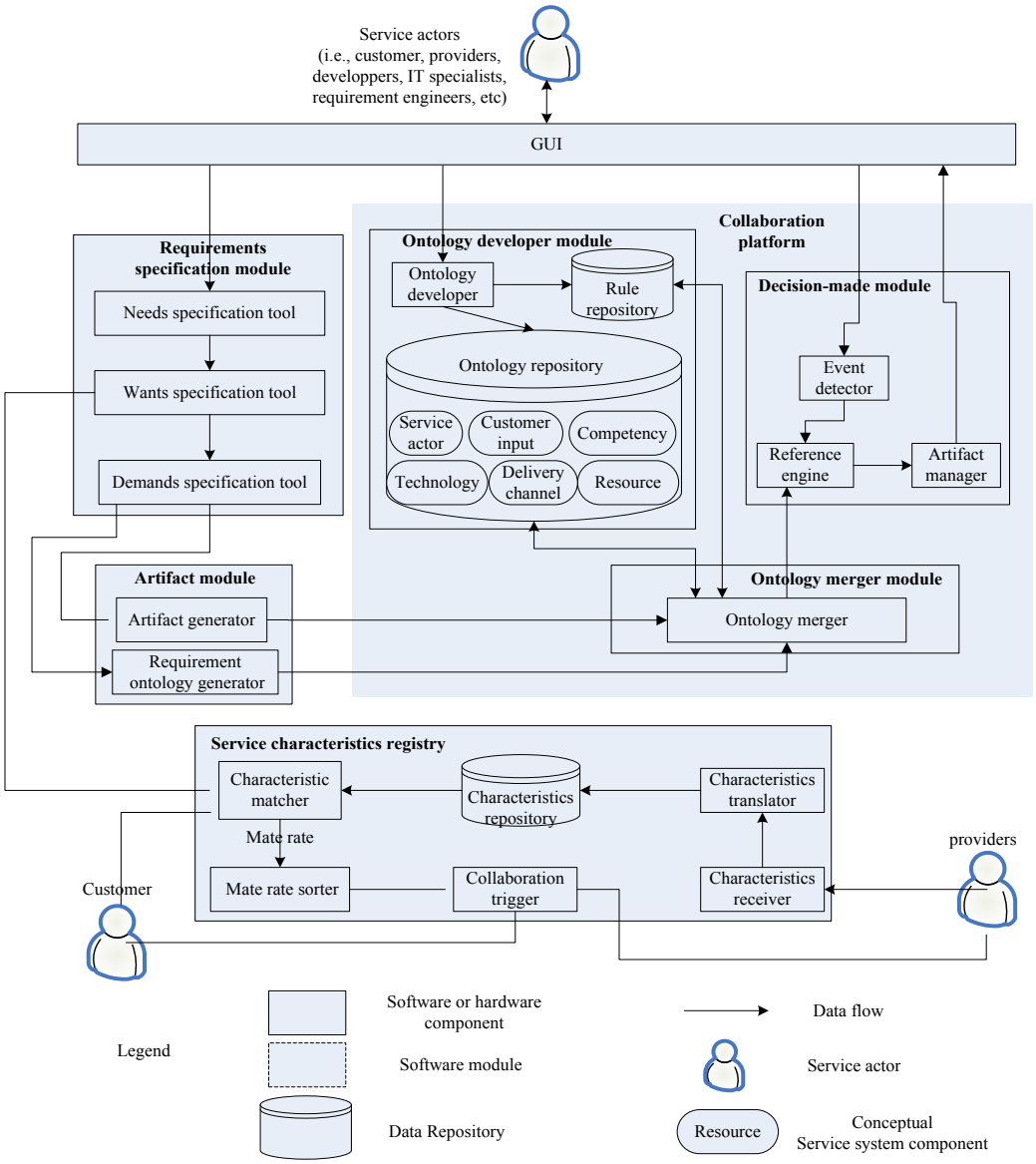


Figure 7.1 Technical architecture for creating an IT-enabled service system.

7.2.1 Ontology Developer Module

This module is used to develop service-based ontologies and SWRL rules. It is mainly composed of three components: the ontology developer, ontology repository and rule repository.

So far, various tools have been developed to design ontologies, such as Protégé (Protégé 4 2011) and Kaon (Kaon 2011). In addition, ontology can be developped APIs, such as OWL 2 API (OWL2API, 2011) and Protégé OWL API (ProtégéOWLAPI, 2011).

Protégé is widely used since it provides graphical interfaces for developers and also has large quantities of plug-ins to facilitate the development of ontology. Protégé also provides Java API for developers to develop Java applications, namely Protégé OWL API that is developed based on OWL API. The OWL API provides a set of classes and functions to support the development of ontology-based applications by using OWL ontology. It can also be integrated with inference engines to reason on an established knowledge base. In this implementation, we adopt OWL API 3.2.4 rather than Protégé OWL API to develop the prototype based on OWL ontologies.

We built three kinds of ontologies with OWL 2 API: including generic ontology, requirements/domain ontology from customer requirements, and artifact ontology for describing business artifacts. All these three kinds of ontologies are saved in the format of OWL/XML files. As shown in the Figure 7.1, the ontology repository is mainly composed of formalized knowledge about service system components (i.e., resources, technology, competency, delivery channel, service actor, and customer input) and their relationships.

Ontology repository: this repository is used to store OWL 2 DL ontology in the format of OWL/XML. In Table 7.1, we present a snapshot of this generic service system ontology. Interested reader may refer to Appendix D.1 for its complete syntax. An ontology file is self-described. It is delimited with the tag <Ontology> and </Ontology>, which includes subtags (<SubClassOf>…</SubClassOf>) describing concepts, properties, and axioms.

Table 7.1 Snapshot of the generic ontology

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://ontology/test/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://ontology/test/">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <EquivalentClasses>
      <Class IRI="#Non_FunctionalCharacteristics"/>
      <Class IRI="#ServiceCharacteristics"/>
</EquivalentClasses>
…
   <SubClassOf>
      <Class IRI="#AdvicerelatedServiceSystem"/>
      <Class IRI="#SystemType"/>
</SubClassOf>
…
   <SubClassOf>
```

```
      <Class IRI="#AtomicServiceSystem"/>
      <Class IRI="#ServiceSystem"/>
   </SubClassOf>
  …
<ObjectPropertyRange>
      <ObjectProperty IRI="#technologyCharacteristicsOf"/>
      <Class IRI="#Technology"/>
</ObjectPropertyRange>
…
  </DataPropertyDomain>
     <DataPropertyDomain>
      <DataProperty IRI="#comHasDescription"/>
      <Class IRI="#Competency"/>
   </DataPropertyDomain>
   …
</Ontology>
```

Rule repository: this repository is used to save SWRL rules. In a similar to the ontology repository, SWRL rules are structured with XML documents by respecting the OWL format. In Table 7.2, we present an example of business rules in the OWL/XML format. Each SWRL rule mainly includes two parts: body and head. The body part denotes the preconditions and the head part represents the effects, which confirms to the syntax of SWRL that was presented in Chapter 4.

Table 7.2 Example of business rule file

```
<DLSafeRule>
      <Body>
         <ClassAtom>
            <Class IRI="http://ontology/test/#Events"/>
            <Variable IRI="http://ontology/test/#Var_Event1"/>
         </ClassAtom>
         <ClassAtom>
            <Class IRI="http://ontology/test/#VisitorArtifact"/>
            <Variable IRI="http://ontology/test/#Var_Visitor1"/>
         </ClassAtom>
         <DataPropertyAtom>
            <DataProperty IRI="http://ontology/test/#hasEventID"/>
            <Variable IRI="http://ontology/test/#Var_Event1"/>
            <Literal
datatypeIRI="http://www.w3.org/2001/XMLSchema#string">isVisited</Literal>
         </DataPropertyAtom>
         <DataPropertyAtom>
            <DataProperty IRI="http://ontology/test/#hasEventValue"/>
            <Variable IRI="http://ontology/test/#Var_Event1"/>
            <Literal
datatypeIRI="http://www.w3.org/2001/XMLSchema#boolean">true</Literal>
         </DataPropertyAtom>
         <DataPropertyAtom>
            <DataProperty IRI="http://ontology/test/#hasStates"/>
            <Variable IRI="http://ontology/test/#Var_Visitor1"/>
            <Literal
datatypeIRI="http://www.w3.org/2001/XMLSchema#string">Init_Visitor</Literal>
         </DataPropertyAtom>
      </Body>

      <Head>
         <DataPropertyAtom>
```

```
            <DataProperty IRI="http://ontology/test/#hasStates"/>
            <Variable IRI="http://ontology/test/#Var_Visitor1"/>
            <Literal
datatypeIRI="http://www.w3.org/2001/XMLSchema#string">Created_Visitor</Literal>
         </DataPropertyAtom>
         <DataPropertyAtom>
            <DataProperty IRI="http://ontology/test/#hasVisitorID"/>
            <Variable IRI="http://ontology/test/#Var_Visitor1"/>
            <Literal
datatypeIRI="http://www.w3.org/2001/XMLSchema#string">Visitor_e4SLjg</Literal>
         </DataPropertyAtom>
      </Head>
   </DLSafeRule>
```

In later of this chapter (Section 7.3), we will present an example to show how various service actors collaborate with one another based on ontologies, business artifacts and business rules.

7.2.2 Requirement Specification Module

The main function of this module is to concretize unclear and imprecise customer needs in natural language into clear and precise customer requirements expressed in SBVR. As mentioned in Chapter 4, service providers and customers collaborate with each other to specify customer wants using the graphical *i**model from customer needs. Service providers further specify demands with the SRML formal language from customer wants. Customer demands expressed with the SRML are finally transformed into SBVR to enable collaboration among IT-specialists, customers, business experts and developers. This module is composed of the following three components:

Needs specification tool: facilitates the specification from natural-language-base customer needs into customer wants in the graphical *i**model. The inputs of these components are customer needs in natural language and its outputs are customer wants in *i**model, which can be saved in graph format.

Wants specification tool: supports the specification from wants in the *i**model to customer demands in SRML. The inputs of this component are customer wants in *i**model, and its outputs are formal customer demands in SRML, which can be saved as a XML file as we deal with other formal languages (i.e., OWL and SWRL).

Demands specification tool: facilitates the specification from customer demands in SRML into customer requirements in SBVR. The inputs of this component are formal customer demands in SRML, and its outputs are customer requirements in SBVR sentences. SBVR sentences can be saved as texts due to its natural-like interfaces.

7.2.3 Service Characteristic Registry

This module is used to help service actors look for their partners at design time. We describe service characteristics with a structured language, namely SRML. At design time, a customer service system can identify its providers by matching its expected service characteristics with its potential providers' service characteristics. This module mainly includes the following components:

Characteristic receiver: this component is used to receive a service provider's exposed service characteristics. We suggest that SRML represent respective service characteristics due to its structured, formal, and QoS-based specification.

Characteristic repository: is used to save SRML-based service characteristics in XML files.

Characteristic matcher: this component is used to match a customer's expected service characteristics or requirement characteristics in SRML with a set of service characteristics in the characteristic repository to identity the appropriate service provider for the customer.

Mate rate sorter: this component is used to descendingly or ascendingly sort the mate rate between various service characteristics in the characteristic repository and the customer's requirement characteristics. The component then identifies the service provider that ranks first or last as the appropriate provider for the customer.

Collaboration trigger: this component is used to trigger the collaboration between the identified provider and the customer.

7.2.4 Artifact Module

As mentioned in Chapter 4, service actors can enrich service system ontology, identify business artifacts and business rules from SBVR-based customer requirements. This module mainly realizes these functions and is composed of two components:

Requirement ontology generator: this component is used to enrich service system ontology and to identify business rules in terms of SBVR expressions. Customer requirements contain large quantities of concepts, object properties, data properties, instances, axioms, and business rules to describe domain-related policies or contexts.

Artifact generator: is used to generate business artifacts from SBVR-based customer requirements. This component holds the input of customer requirements in SBVR and output of the obtained artifact model described in the OWL/XML format.

In Chapter 6, we illustrated how to manually enrich service system ontology and to identify business artifacts through a case study. In Chapter 4, we proved the possibility to transform SBVR vocabularies to OWL ontology and SBVR rules into SWRL rules. We also theoretically elaborated their mappings. However, since SBVR is more expressive than either OWL or SWRL, not all SBVR expressions can be mapped into them without losing information and semantics.

In our prototype, we do not directly implement the mapping from SBVR to OWL 2 and SWRL. For sake of simplicity and proof of concept, we adopt another relatively simple way to enrich service system ontology and business rules by formulating SBVR expressions with the ACE syntax (APE Help 2011, Fuchs and Schwitter 1996). Both of SBVR and ACE are languages to express natural-like syntaxes. SBVR is proposed to as a formal business rule language and it provides logical keywords to describe business logics and rules. We found that some keywords in SBVR are not supported by the current version of ACE, such as recommendation (e.g., should), possibility (e.g., can), necessity (e.g., it is necessary), or sentence subordination, etc. Without lost of generality, we validate our mapping approach by using the ACE Parsing Engine (APE) to map ACE expressions representing SBVR expressions into OWL ontology.

The principal of APE lies on the fact that the ACE context is firstly transformed to Discourse Representation Structure (DRS) and then DRS is mapped to various formal logic languages as described by the DRS technical report (DRS 2011). We also directly call for the APE API to transform ACE text into OWL ontology. For example, assume that customer requirements are shown in Table 7.3.

Table 7.3 Example of customer requirements in ACE text

Yong is a freshman. He v:failsIn TofelTest. He v:looksFor a n:Course that improves the English. The n :AvailableTime that v:isOn Monday v:isFrom 15 and v:isTo 18. The n :AvailableTime that v:isOn Tuesday v:isFrom 15 and v:isTo 18. The n :AvailableTime that v:isOn Wednesday v:isFrom 15 and v:isTo 18. The n :AvailableTime that v:isOn Thursday v:isFrom 15 and v:isTo 18. The n :AvailableTime that v:isOn Friday v:isFrom 15 and v:isTo 18. The n :AvailableTime that v:isOn Saturday v:isFrom 15 and v:isTo 18. The n :AvailableTime that v:isOn Sunday v:isFrom 15 and v:isTo 18. Yong has Freely_Access_Online_Education_Resources. Freely_Access_Online_Education_Resources is a n:TechnologyCharacteristic. Access_Resources is a n:InternalTechnology and it achieves Freely_Access_Online_Education_Resources. The n:TechnologyCharacteristic has five features. AccessTime is a feature. Availability is a feature. AccessLocation is a feature. The n:TechnologyCharacteristic has a n:deliveryChannel. The n:TechnologyCharacteristic has a n:readAuthroty. The n:TechnologyCharacteristic has a n:updateAuthroty. The n:accessTime of Access_Resource is 24 hours. Yong v:studiesVia Web and v:studiesVia the n:MobileNetwork. The n:accessLocation is a:unlimitedLocation.

Figure 7.2 shows our developed GUI for service providers/customers captures clear and precise customer requirements and displays the mapped OWL file with two possible formats: OWL functional

syntax or OWL/XML format. The ontology file is listed in the "output area". The "Merge" functionality in the GUI allows to select, open and show multiple ontology files in the output area (see Appendix C.1).

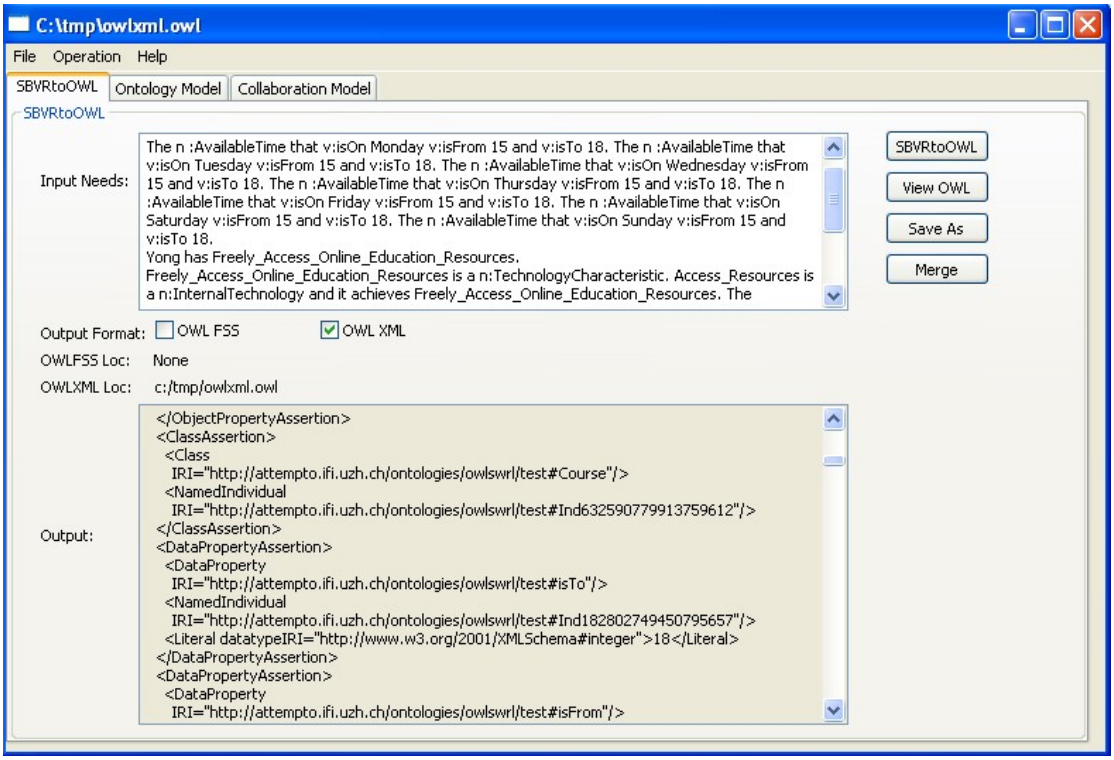


Figure 7.2 Snapshot to realizing the mapping from SBVR to OWL

Although APE Engine supports input as natural-like English expressions, it also imposes some restrictions on input phases to make the engine understand phases. In Table 7.3 for example, a student, 'Yong,' sets requirements to one service characteristic, 'Freely_Access_Online_Education_Resources,' thereby leading to an expected service characteristic. The symbol 'n:' denotes a noun (either concatenated phase or single word) whereas the symbol 'v:' denotes a verb (either concatenated phase or single word). Others symbols such as 'a:', and 'p:' respectively denote adverbs (either concatenated phase or single word) and proper names (either concatenated phase or single word). More details can be referred to (APE Help 2011).

In our implementation, we relied on the APE engine to realize the mapping from SBVR-based customer requirements to ACE expressions and then to OWL ontologies.

The input customer requirements expressed in the ACE-based syntax in Table 7.3 is mapped to ontology expressed in OWL/XML, as shown in Table 7.4 and the complete OWL expressions are shown in the Appendix D.2.

Table 7.4 Snapshot of requirements ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<Ontology
   xml:base="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.w3.org/2002/07/owl#"
   ontologyIRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test">
  <ObjectPropertyAssertion>
   <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#failsIn"/>
   <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Yong"/>
   <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#TofelTest"/>
  </ObjectPropertyAssertion>
  <ClassAssertion>
   <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#freshman"/>
   <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Yong"/>
  </ClassAssertion>
  ……
  </Ontology>
```

The mapped requirement ontology can be viewed as the domain ontology to enrich the generic ontology and business rules. As mentioned before, IT specialists can deduce artifacts from the requirement ontology. The general deduction rule assumes that each noun concept in customer requirements can be considered a business artifact and its attributes can be viewed as its data or object properties. Constraints applied to business artifacts and their relationships with other artifacts are considered business rules.

It is worth noting that, in practice, we have to identify the appropriate granularity for a business artifact. If the granularity is too large, it will lose information and be difficult to reflect practical business logics. If the granularity is too little, it will increase workload and reduce efficiency. Since the granularity of artifacts requires the collaboration and negotiation of IT specialists, business experts, and developers, the generation of business artifacts from SBVR requirements needs some aspects or levels of intellectual work.

For the sake of simplicity, as per customer requirements, we manually identify some artifacts to simulate our artifact-driven collaboration model, such as 'ReturnedItem', 'Stock', 'Shipment', 'CourseList', 'Course', 'Bill', 'Evaluation', and 'Certificate'. As mentioned before, each business artifact is composed of a set of attributes, states, and lifecycles. In Section 7.3, we present two artifacts as examples to simulate how the system guides service actors' decision making. More details about these artifacts can be found in Appendix B.

As for the artifact ontology, we separate business artifact descriptions into two parts: an ontological description and related business rules.

The ontological description is mainly based on concepts and properties in generic ontology and requirement ontology whereas the related artifact rules are developed through the implementation of the *developArtifactOntology()* function. More details about this function can be found in Appendix C.2.

7.2.5 Ontology Merger

This module is used to merge various ontologies, such as generic ontology, requirements/domain ontology, and artifact ontology in a same ontology and finally provides the merged ontology into the inference engine.

As mentioned before, the generic service system ontology, domain ontology, and the artifact ontology are saved in different files. A key issue is merging concepts from different ontologies. This module is expected to deal with concepts in different domains and then automatically merge their ontology files in a single file using functions in OWL 2 API.

7.2.6 Decision-making Module

This module is used to reason on the established ontology repository and the rule repository to provide guidelines to service actors' collaboration and decision making. It is mainly composed of three components:

Event detector: this component is used to detect events that result from the changes to service actors' contexts, their environments or requirements. One example of the Event detector is Sensor that records the changes of service actors' context (i.e., from one location to another location). The Event detector will forward its captured events to the inference engine.

Inference engine: this component is used to reasons on established ontology and SWRL rules to generate inferred OWL facts. The reasoners that support OWL 2 and SWRL rules mainly include Pellet (Pellet 2011), FaCT++ (Fact++ 2011), and HermiT (Hermit 2011). In our implementation we adopt Pellet 2.3.0 as our inference engine. Pellet is an open source OWL reasoner in Java which supports OWL 2 profiles including OWL 2 EL. It incorporates optimizations for nominals, conjunctive query answering, and incremental reasoning. Pellet has full SWRL support, including the core built-ins. Pellet interprets SWRL using the DL-Safe Rules notion which means rules will be applied to only named individuals in the ontology. At this point, the latest version is Pellet 2.3.0 that was updated in August 2011.

Based on the Pellet libraries, we develop an API including several functions; the function *reasoner*()implement the reasoning on established ontology base and rule base. The function *generateReasonedOntology*() generates the inferred OWL model. The function *generateReasonedOntology*() provides flexibility to customize the types of inferred axioms to obtain. For example, the following code initializes the engine before using our functions.

```
List<InferredAxiomGenerator<? extends OWLAxiom>> gens = new ArrayList<InferredAxiomGenerator<? extends OWLAxiom>>();
gens.add(new InferredSubClassAxiomGenerator());
gens.add(new InferredEquivalentClassAxiomGenerator());
gens.add(new InferredSubClassAxiomGenerator());
gens.add(new InferredDisjointClassesAxiomGenerator());
gens.add(new InferredPropertyAssertionGenerator());
gens.add(new InferredClassAssertionAxiomGenerator());
gens.add(new InferredSubDataPropertyAxiomGenerator());
gens.add(new InferredSubObjectPropertyAxiomGenerator());
gens.add(new InferredDataPropertyCharacteristicAxiomGenerator());
gens.add(new InferredEquivalentDataPropertiesAxiomGenerator());
gens.add(new InferredEquivalentObjectPropertyAxiomGenerator());
gens.add(new InferredInverseObjectPropertiesAxiomGenerator());
gens.add(new InferredObjectPropertyCharacteristicAxiomGenerator());
```

Artifact manager: based on the inferred OWL model, this component is used to update business artifacts and to send event messages to service actors to guide their decision making. Finally, the function *saveInferredOntology*() saves the inferred OWL model into an OWL file and the function mergeOntology() allows merging inferred ontology and current knowledge base for updating the established knowledge base.

7.2.7 Graphic User Interface

We have developed a graphical user interface (GUI) to facilitate the interactions with the developed prototype. The interface allows to capture customer requirements expressed in ACE syntax and to generate from them the OWL ontology. The interface also views OWL ontologies and business rules repository, merge ontologies, separate axioms and concepts in an OWL ontology files. In addition, in the GUI, present an interface to simulate the reasoning based on ontologies, business rules and events in order to provide guidelines to help a service actor make a decision.

As shown in the snapshots in Figure 7.3, we list a set of actions to be performed by various service based on our case study in the e-learning service system presented in Chapter 6. When a service actor performs an action, the Event Detector perceives the event and then forwards it to the inference engine which sets up the reasoner and triggers the reasoning processes on the established ontology repository and business rule reposi-

tory. After the execution of business rules, the inference engine returns an inferred OWL model in order to update business artifacts and to provide guidelines for service actors about what to do in the next step through notification messages as shown in the message notification Area (see Figure 7.3).

From a technical perspective, we have developed the GUI with Eclipse 3.4.1 and Java JDK 1.6 as the development platform and we used SWT (SWT 2011) plug-ins and its widget toolkit. In Appendix C.3, we provide a snapshot of code to capture the event "become a customer" and execute business rules as well as reason on established knowledge.

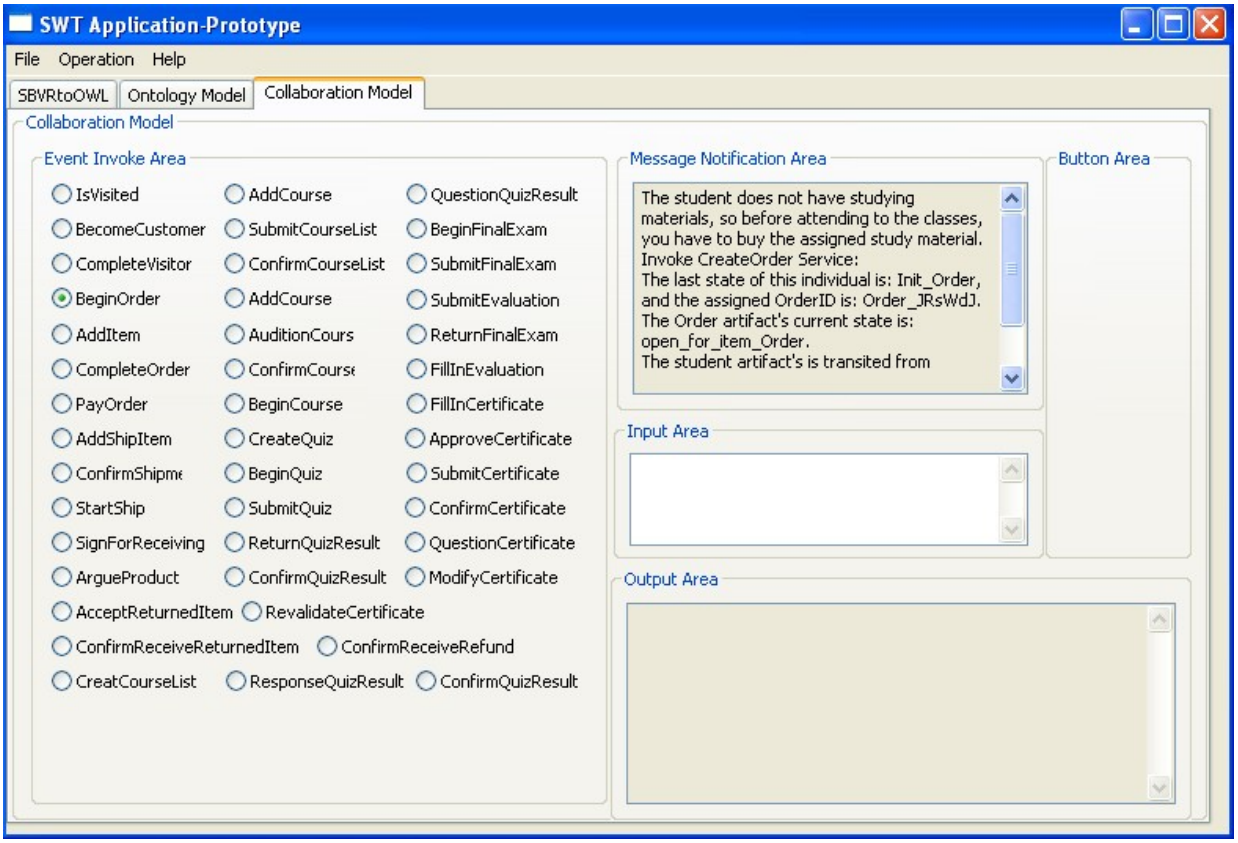


Figure 7.3 Snapshot to simulate decision making

7.3 Supporting Business Artifacts and Their Exchange

Based on our case study presented in Chapter 6, we herein present two artifacts, respectively the 'Visitor' artifact and the 'Order' artifact, to simulate how the prototype provides guidelines to facilitate collaboration among students, e-learning providers and external-ecommerce providers/vendors. More details about these two artifacts can be found in Appendix B.1 and B.2.

7.3.1 Visitor Artifact

The 'Visitor' artifact aims at recording information concerning visitors that access to the online e-learning system. We identify two attributes for this artifact: VisitorID and visitingTime and three states: Created_Visitor, Created_Customer, and QuittedVisitor. Its lifecycle model is illustrated in Figure 7.4 and its referred business rules are shown in Table 7.5.

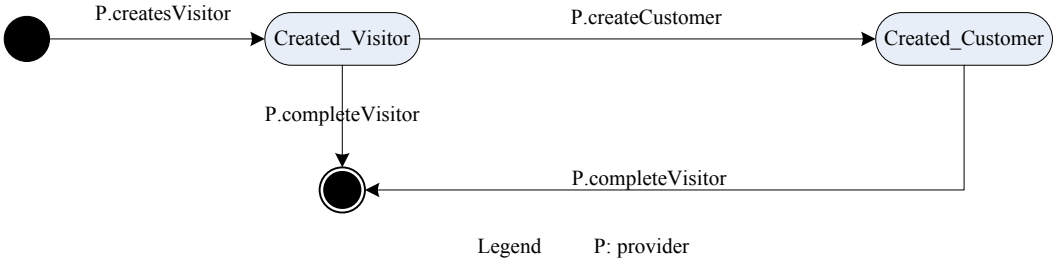


Figure 7.4 Lifecycle for visitor artifact

Table 7.5 Business rules for visitor artifact

Rule 1: Events(?Var_Event1), VisitorArtifact(?Var_Visitor1), hasEventID(?Var_Event1, "isVisited"), hasEventValue(?Var_Event1, true), hasStates(?Var_Visitor1, "Init_Visitor") -> hasStates(?Var_Visitor1, "Created_Visitor"), hasVisitorID(?Var_Visitor1, "Visitor_6v2ngi") **Rule 2**: Events(?Var_VisitorBeCus), StudentArtifact(?Var_Customer), VisitorArtifact(?Var_Visitor1), hasEventID(?Var_VisitorBeCus, "becomeCustomer"), hasEventValue(?Var_VisitorBeCus, true), hasStates(?Var_Customer, "Init_Customer"), hasStates(?Var_Visitor1, "Created_Visitor") -> hasCustomerID(?Var_Customer, "Customer_0pYs6b"), hasStates(?Var_Customer, "Created_Customer"), hasStates(?Var_Visitor1, "QuitedVisitor") **Rule 3**: Events(?Var_Event_Quit), VisitorArtifact(?Var_Visitor_Quit), hasEventID(?Var_Event_Quit, "completeVisitor"), hasEventValue(?Var_Event_Quit, true), hasStates(?Var_Visitor_Quit, "Created_Visitor") -> hasStates(?Var_Visitor_Quit, "QuitedVisitor")

When a visitor accesses the e-learning services (i.e., check the catalogue), The system generates the event of 'CreateVisitorArtifact' and then it randomly assigns the visitor a "VisitorID" and records its accessing time with the attribute "visitingTime". The output message is shown in Table 7.6.

Table 7.6 The output message when a visitor visits the e-learning platform

Invoke CreateVisitorArtifact Service: The last state of this individual is: Init_Visitor, and the assigned visitorID is: Visitor_OhOE7w. The visitor artifact's current state is:Created_Visitor.

Assuming that the student decides to become a customer, it will then invoke the "CreateCustomerArtifact" service, which will assign the student a "CustomerID". As shown in Table 7.7, the visitor becomes a customer and the visitor artifact's state is transformed from "Init_Customer" to

"Created_Customer". The output message also informs the student what to do in the next step.

Table 7.7 The output message when the visitor becomes a customer

Invoke CreateCustomerArtifact Service: The last state of this individual is: Init_Customer, and the assigned CustomerID is: Customer_W7MNqm. The visitor artifact's current state is:Created_Customer.

If the visitor is just a visitor and does not expect to become a customer, the visitor can choose to quit the system, which automatically invokes the "CompleteVisitorArtifact" service and its state is transformed from "Init_Visitor" to "QuittedVisitor", as shown in Table 7.8.

Table 7.8 The output message when the visitor quites the e-learning platform

Invoke CompleteVisitorArtifact Service: The last state of this individual is: QuitedVisitor. The visitor artifact's current state is:QuitedVisitor.this individial has visitor ID:["Visitor_thd73v"^^xsd:string]

7.3.2 Order Artifact

Another example, namely the 'Order' artifact, illustrates decision making support for service actors. This artifact is used to record a student's purchase of studying materials, such as books, handsets, and microphones. This artifact includes the following attributes: OrderID, CustID, Bid, ScheduleDate, and Discount, and has the following states: open_for_item, ready_to_pay, paid, ready_for_ship, in_ship, shipped, in_arguing, and completed. In addition, the artifact lifecycle is illustrated in Figure 7.5 and the interception of business rules for this artifact is illustrated in Table 7.9.

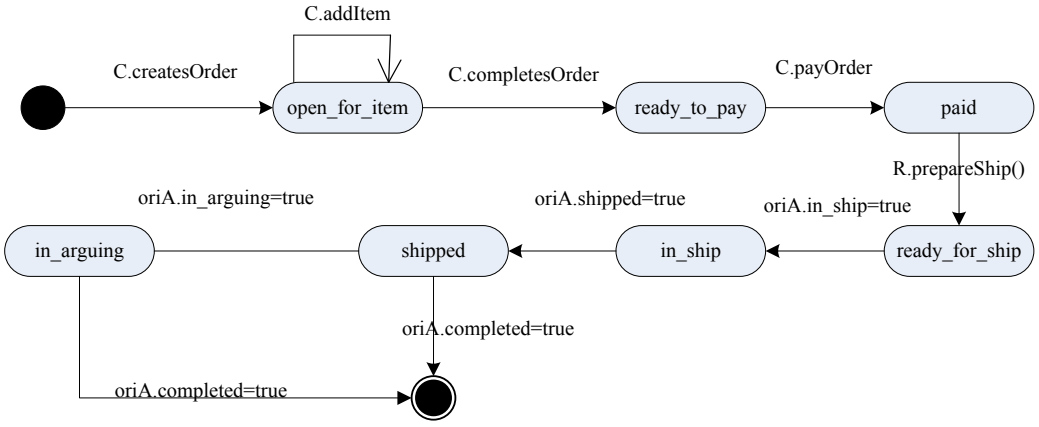


Figure 7.5 Lifecycle for order artifact

As shown in Figure 7.3, service actors communicate their decisions through the "Event Invoke Area" in the Graphic User Interface.

Table 7.9 Intercept business rules for order artifact

Rule 1: C.createsOder Event(?Var_Event_BeginOrder), OrderArtifact(?Var_Order), hasEvent-tID(?Var_Event_BeginOrder, "BeginOrder"), hasEvent-Value(?Var_Event_BeginOrder,true), hasState(?Var_Order,"Init_Order"),StudentArtifact(?Var_Stu),hasCustomerID(?Var_Stu, "Cust_ID"), -> hasState(?Var_Order, "Open_for_Item_Order"),hasOrderID(?Var_Order, "Or-der_ID"),hasCustomerID(?Var_Order, "Cust_ID"), **Rule 2: C.addItem** Event(?Var_Event_AddItem), hasEventID(?Var_Event_AddItem, "AddItem"), hasEvent-Value(?Var_Event_AddItem,true), OrderArtifact(?Var_Order), OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, "Init_OrderItem"), hasState(?Var_Order, "Open_for_Item"), hasQuantity(?Var_OrderItem, ?int_quantity),hasPrice(?Var_OrderItem, ?float_price) -> hasOrderID(?Var_OrderItem, "Order_ID"), hasItemAmount(?Var_OrderItem, ?float_price*?int_quantity), hasState(?Var_OrderItem, "newly_added") **Rule3: C. completeOrder** Event(?Var_Event_CompleteOrder), hasEventID(?Var_Event_CompleteOrder, "Com-pleteOrder"), hasEventValue(?Var_Event_CompleteOrder,true), OrderArtifact(?Var_Order), hasState(?Var_Order, "open_for_item") -> hasState(?Var_Order, "ready_to_pay_Order")

The messages shown in the "Message Notification Area" indicate details about the state transitions for various artifacts and inform service actors what are the possible actions in their next step. For example, if the customer invokes the event "BeginOrder", the inference engine will reason on the established ontology base and business rules base and output inferred OWL model. Then based on the inferred OWL model, the artifact manager returns the details about the updating of artifacts and indicates to what to do in the next step. The output messages are shown in Table 7.10.

Table 7.10 The output message to Create 'order' artifact

The student does not have studying materials, so before attending to the classes, you have to buy the assigned study material. Invoke CreateOrder Service: The last state of this individual is: Init_Order, and the assigned OrderID is: Or-der_JRsWdJ. The Order artifact's current state is: open_for_item_Order. The student artifact's is transited from in_audition_Customer to in_order_Customer.

Based on the guidelines in Table 7.10, the customer can add order items and "addItem" event will be invoked to add items into the order artifact. As shown in Table 7.11, the system will assign an item ID for the item the customer newly adds to the order artifact.

Table 7.11 The output message to add item to 'order' artifact

Invoke AddItem Service: the orderITem artifact is created and transites from Init_OrderItem to newly_added_orderItemthe assigned OrderItemID is: OrderItem_JvOVFQ. The Order artifact's current state is: open_for_item_Order and it does not change.

Based on the above simulation resscenarios, the artifact-based system provides effective guidelines for various service actors to make appro-

priate decisions in collaboration, which is advantageous to ensure semantic interoperability in collaboration.

7.4 Conclusion

In this chapter, we have presented the technical architecture of IT-enabled service systems and discussed the implementation of a simple prototype to test our proposed artifact-based model. We at first manually specify customer needs in natural-like language into customer requirements in SBVR and then we transform SBVR requirements into the OWL format. We then generate some business artifacts based in customer requirements and describe them in the OWL/XML format. After that, we merge our developed generic ontology, requirement ontology and artifact ontology together to feed the inference engine. Ontology and business rules are respectively saved in the ontology base and rule base. When an event is invoked, the inference engine (Pellet) reasons on the current ontology base and rule base and invokes the appropriate services to update artifacts and informs service actors with potential actions to do in the next step. Our experiments demonstrate the effectiveness of our proposed artifact-based approach to provide service actors with guidelines for their decision making in a collaboration context.

Chapter 8

Conclusions and Future Research

8.1 Introduction

In this chapter, we present the main research results in this study and the outlook of our future work. We review the key challenges and present our principal contributions, research results, as well as the key value of our models and approaches. We finally provide a broader view of our future work.

8.2 Key Issues and Conclusions

Throughout this thesis, we shed the light on the design of IT-enabled service systems driven by requirements and collaboration for value addition. We argue that the main added value of creating an IT-enabled service system lies on the following aspects:

1- **Multiple perspectives on IT-enabled services**: we integrate views from the business domain (Service Marketing, Service Operation and Service Management) and technical domain (Information Science and Computer Science) in our study of IT-enabled services. In our study we dealt with the main problems of traditional business services and the challenges related to service characteristics, which cannot be processed by software thereby lacking of automation.

2- **Formalize business knowledge and provide a common understanding of IT-enabled service systems**: we employ Knowledge Management, Requirements Engineering, and Semantic Web techniques to add

a layer of formalism to existing business knowledge on services and developing a common understanding of concepts in IT-enabled service systems based on ontologies. After that, Customers, business experts, IT specialists, developers, software and other service actors in different levels of business can share the common understanding to specify their requirements and collaborate.

3- **A top-down requirements engineering approach to concretize ambiguous customer needs into clear and precise customer requirements and a side-way approach to facilitate the collaboration of various service actors**.

4- **Semantic interoperability and Data-driven collaboration:** after the formalizing of business knowledge and the identification of customer requirements, various service actors have to model their adhoc collaboration. The traditional approaches prove inappropriate as it adopts business processes to interconnect business activities and reduce flexibility in dynamic collaboration. In this our work, we promote an artifact-driven approach to deal with the dynamic collaboration and achieve semantic interoperability.

5- **Make service actors focus on their core businesses and relationships with their service partners rather than interested in underlying service systems**: During collaboration, service actors do not expect to take into account their partners' back office (internal complex components and structure) but instead expect their partners to inform them what to do rather than how to do it. In this sense, we advocate a mechanism to make service actors focus on their own business and the interactions with their service partners for exchanging explicit business knowledge (front office) without taking into consideration their partners' internal components and how their partners support their interactions.

8.3 Value of Research and Contributions

Our main contributions lie on the following five aspects: 1) the study of services from different domains and in particular the added-value of IT-enabled services , 2) the introduction of the service system reference model to describe service system from different perspectives, 3) A top-down approached based on traditional Requirement Engineering and the side-way approach to make it possible to integrate service actors from different levels to co-create a service system, and 4) A data-driven approach to model ad hoc and dynamic collaboration to enable semantic interoperability and flexibility. 5) A middle-out methodology to create IT-enabled service systems. We respectively summarize each of these aspects in the following section.

8.3.1 Value Addition for IT-enabled Services

In Chapter 2, we survey different interpretations of services in different disciplines including business services, e-services, web services, and IT services. In this thesis, we focused on IT-enabled services that integrate traditional services in the business domain and e-services/IT services/web services in Information Science and/or Computer Science. We argue that IT-enabled services cover the economic benefits and as well as the intangible value of e-services/IT services/web services (such as automation, composition, delivery via e-channels, and bundling). In this sense, the designing of IT-enabled services is not an easy task since it refers to the participation of service actors from different domains.

When designing an IT-enabled service, business knowledge must first be formalized in the business domain and then integrated with business knowledge in the technical domain in order to provide a common understanding for various service actors in different disciplines.

Our work contributes to existing service research in both the business domain and the Information Science and Computer Science domain since we present a middle level between them, which makes our work neither business-oriented nor technical.

8.3.2 Service System Reference Model

Due to the complexity of internal service system components and their relationships, dynamic behaviours and the interdependencies between service systems, it is quite difficult to create a service system for added-value services.

Our proposed service system reference model includes three views to facilitate the understanding and design of service systems. The systematic view presented a graphical view to illustrate service systems' components and the representation of interactions between providers and customers. The ontological view presented a common understanding of concepts in service systems and described dynamic behaviours. We identify the following advantages of the ontological model: 1) the use of common understandings of concepts and vocabulary for service actors that need to share information; 2) the reuse of domain knowledge; 3) the possibility to address conceptual ambiguities for service actors in collaboration and to improve interoperability; 4) to provide a knowledge base managed by business rules to support service actors' decision makings.

The characteristics view separates the complex internal service system components from the exposed service utilities, and represents both

services and internal components with a set of qualitative and quantitative features and constraints. In this sense, the characteristics view helps to find potential partner relationships between two service systems without taking into account their internal complex structure and components.

With the three different views of service systems, our proposed service system reference model provides the following benefits: 1) a middle-out approach to enable the collaboration of various service actors; 2) illustration of the basic service system components and their relationships; 3) an ontological model to deal with conceptual ambiguities in multi-disciplinary context; and 5) its extensibility to encompass other views of service systems. Finally, the characteristics view makes it possible to formulate innovation since the modification of internal service system component characteristics (i.e., technology characteristics) will lead to different service characteristics that may imply innovation.

8.3.3 Requirement Model

The requirement model is mainly proposed to deal with the following two kinds of gaps: 1) the gap between services as perceived by providers and services as expected by customers, and 2) the collaboration of service actors that may use different models and/or languages.

The first gap can be partially addressed by traditional top-down requirement engineering approaches. However, these approaches overemphasize the role of designers or requirement engineering in identify customer needs and do not completely take into account the collaboration of various service actors to co-identify and co-concretize customer needs. The second gap requires a common structured language that is close to natural language, close to business-level language (e.g., UML), as well as close to structured formal language (i.e., SWRL) and it also requires a middle-out approach to facilitate the co-work of various service actors to identify clear and precise customer requirements. Another important engineering requirement is that the approach propagates changes to customer needs into customer requirements for adaptability.

In this thesis, we propose a top-down and side-way approach to deal with these issues. Due to the lack of domain knowledge, customers have difficulty in identifying what they need. Generally, they express their needs in their own languages (i.e., natural language). In this sense, providers have to help them to identify their needs and concretize their unstructured, unclear, and imprecise needs into structured, clear, and precise requirements. The top-down requirement engineering approach can help to achieve that. Different from the traditional top-down requirement engineer-

ing approach, we hierarchically distinguish fours levels of customer needs including customer needs, customer wants, customer demands and customer requirements and take into consideration negotiation and trade-offs between service actors to concretize customer needs. When specifying customer needs, we use a graphical model (*i**model) to facilitate co-concretizing customer needs into customer wants. Due to the lack of formal logic that could lead to different interpretations, we introduce a structured SRML formal language to specify customer wants into customer demands. The main value of SRML lies in the fact that it represents a set of quantitative and qualitative features and constraints on services, which can be used at design time to ensure service quality and measure customer satisfaction in the run time.

When creating a service system, some core service actors such as customers, business experts, IT specialists, and developers use different models and/or languages, which make them difficult to collaborate with each other for value co-creation. We present a side-way approach to use SBVR to represent customer requirements as a common structured language that could be understood by all service actors due to its natural-like interface but internally with formal logics and notations. In this sense, SBVR is feasible to be mapped to formal implementation language (i.e., FL, OWL, R2ML, and SWRL) understood by machines.

We argue that: 1) the top-down requirement approach guarantees that the service is driven by customer needs, their requirements and all underlying resources of service providers are mobilized towards this end; and 2) the side-way requirement approach facilitates interactions among service actors of the same discipline (i.e., domain of expertise) in order to specify their requirements in an expressiveness language and make possible propagation of further changes through the top-down requirement approach.

8.3.4 Collaboration Model

Traditional approaches adopt activity-driven approaches to model business processes and implement each activity with Web services. Workflow techniques are then used to execute the modelled business processes. However, with these approaches it is difficult to achieve semantic interoperability at the business by comparing the output of a Web service to the input of its precedent Web service. Even though we adopt semantic description languages (i.e., WSDL 2.0) to describe web services, activities in business process are quite predefined with respect to the underlying business logic, which means it lacks flexibility to be modified to adapt to dynamic customer requirements.

In this thesis, we proposed an artifact-driven approach to deal with dynamic collaboration in service systems. Each collaborative activity is modelled as a set of interactions and each interaction is denoted as a data flow. A business artifact is a mechanism used to record chunks of information that can be used by business persons to manage their businesses. In addition, a business artifact not only describes a business entity, but also encompasses the knowledge about what to do without explaining how to do it. We also developed a set of SWRL rules based on a developed OWL 2 ontology to invoke developed web services to update business artifacts.

In addition, in order to achieve semantic interoperability and security, different from other artifact-driven approaches, we distinguished public artifacts and private artifacts and presented four interaction patterns: 1) partners have respective artifacts and manage them separately; 2) partners share a common public artifact and jointly manage it; 3) partners share a common public artifact but manage it separately and; 4) partners share a common public artifact provided by a contractor and other partners play the role of sub-contractor and mange the artifact separately. In this thesis, we assumed that service actors in service systems have enough trust of each other to make it possible for them to share common public artifacts and jointly manage them. At last, in order to reuse recurred collaboration activities and to improve design efficiency, we formulated a common structure for collaboration patterns.

The main value of our proposed collaboration model lies in the fact that it is 1) appropriate to model dynamic and ad hoc collaboration; 2) flexible enough to adapt to dynamic customer requirements; 3) helps to achieve semantic interoperability and security in collaboration and; 4) in collaboration, various participants do no have to know details about collaboration, such as respective complex internal structure and components and how their collaboration are supported and enabled.

8.3.5 How to Create an IT-enabled Service System?

We finally re-examine our proposed general process (see Chapter 3) to create an IT-enabled service system with these aforementioned models and approaches.

Step 1: Design generic service system ontology by referring to the systematic view and ontological view of our reference model.

Step 2: Design domain ontology.

Step 3: In terms of our proposed requirement model, concretize customer needs into customer requirements and obtain service characteristics and expected service characteristics.

Step 4: Evaluate if current service system or external service systems can satisfy well-structured customer requirements.

Step 5: Identify service actors by matching their service characteristics and corresponding requirement characteristics or expected service characteristics and business artifacts.

Step 6: Enrich designed generic and domain ontology based on customer requirements and design the knowledge base and business rule base.

Step 7: Set up collaboration in terms of identified business artifacts, developed business rules, ontology and web services.

Step 8: If there are changes in customer requirements, they will be propagated from unclear and informal customer needs to formal and structured customer requirements. Accordingly, based on the new customer requirements, IT specialists will identify new business artifacts and business rules and the developers will develop new web services or SaaS to update business artifacts.

Similar to the traditional engineering approach to design services that evolve spirally during the service lifecycle (from customer needs to the service design and at last to implementation and evaluation), our approach also evolves spirally through these steps. However, differently, our approach is neither business strategic nor technical and instead integrates knowledge from the business domain and the technical domain (i.e., Information Science and Computer Science). We also presented a common understanding to formalize knowledge and make it possible to be understood by both humans and machines thereby achieving economic benefits and intangible value (i.e., automation, e-channel, composition, and bundling). In addition, at design time, our approach supports the collaboration of service actors from different domains and different levels of business (i.e., customers, business experts, IT specialists, and developers) to co-create an IT-enabled service system in terms of customer requirements and also enables the collaboration of different service actors in a value network to co-innovate services in the run time.

The characteristics view of the reference model ensures the service quality and performance at design time, but also guarantees that the delivered IT-enabled service can adapt to dynamic customer requirements in the run time and customer satisfaction could be dynamically achieved and measured.

8.4 Future Outlook

Our work attempts to create an IT-enabled service system driven by dynamic requirements and ad hoc collaboration. However, it is still far away

from the destination that automatically integrates various service actors from different domain to co-create an IT-enabled service system for value co-creation. Multi-disciplinary approaches are required to explore the economic-social-technical-systematic nature of service systems. During our research work, some technical and strategic problems require attention in the future:

1- The development of the following tools to facilitate the concretization from customer needs into customer requirements:

a) Tool to facilitate the mapping from natural language to *i** model.

b) Tool to realize the mapping from *i** model to SRML.

c) Tool to achieve the mapping from SRML to SBVR.

d) Tool to achieve the mapping between SBVR and SWRL.

2- As mentioned before, SBVR is more expressive than the collection of OWL 2 and SWRL. As a result, some SBVR expressions cannot be described by neither OWL 2 nor SWRL, which will lead to the loss of business logics during the translation if we fully rely on enriching service system ontology and business rules from SBVR-based customer requirements. One solution is to improve the expressiveness OWL 2 and SWRL. Motik et al. (2009) argue that the use of a description graph can improve the expressiveness of OWL 2 and DL-safe SWRL. In this sense, in our future work, we are interested in integrating OWL 2, DL-safe SWRL, and description graph to represent more complex SBVR expressions. Another research topic is to develop a new rule-based language that is more expressive than OWL 2 and SWRL, for example R2ML.

3- Our service system ontology is just in its infancy and simply includes some high-level concepts. One part of our future work is to enrich the service system ontologies. In addition, in our study, we just use SBVR to describe the final specified customer requirements related to IT resources and service system components based on SBVR-based customer requirements. However, SBVR is an expressive business rules language used to describe business policies, strategies, and logics. Apart from SBVR-based requirements characteristics, we are interested in other kinds of information in order to enrich service ontologies and to generate artifacts for automating collaboration (such as a customer profile, provider profile, customer context, or security policy, etc). In real use case, business people should describe all kinds and all levels of businesses with SBVR. In this study, we do not take into account this kind of information. That is one reason why in our case study, it is impossible for us to automatically generate business artifacts as per SBVR requirements. In our future work, we will further formalize all of this information or knowledge and express them

with SBVR. This could help to automatically or semi-automatically enrich service system ontology and to generate artifacts.

4- When presenting the characteristics view in Chapter 3, we argued that the characteristics view can be used to look for partners for a service system at the design time by comparing its exposed service characteristics/expected service characteristics to its partners' expected service characteristics/exposed service characteristics. However, we are interested in developing the comparison mechanism. It is thus necessary to present a bundling model to discuss how to match two service characteristics.

5- Further work includes the possible extension to our presented service system reference model. So far, the reference model includes three views, respectively, the systematic view, the ontological view, and the characteristics view. In our future work, we are interested in extending reference model with more views as As per Zachman's Enterprise Architecture model.

6- In Chapter 5, we distinguished public/common artifacts and private artifacts and presented four different interaction patterns for the sake of semantic interoperability. In this thesis, we just take into account one of them, namely service actors who share common artifacts and jointly manage them. While in real case, a service system may refer to at least one of them. In this sense, a mechanism to achieve the mapping between private artifact and public artifact must be presented.

Bibliography

Alonso-Rasgado M.T., Thompson G. and Dannemark O.J. (2004) State of the Art in Service Design and Modeling. VIVACE Public, 5(1), pp. 1-59.

Alter S. (2002) The Work System Method for Understanding Information Systems and Information System Research. Communications of the Association for Information Systems, 9(1), pp. 90-104.

Alter S. (2008) Service System Fundamentals: Work System, Value Chain, and Life Cycle. IBM Systems Journal, 47(1), pp. 71-85.

APE Help (2011) APE Help Document. "http://attempto.ifi.uzh.ch/site/docs/ape_zwebclient_help.html". lasted accessed in December 2011.

Aslesen H.W. (2007) Knowledge Intensive Business Services and Urban Industrial Development. The Service Industries Journal, 27(3), pp. 321-338.

Badr Y., Narendra N.C. and Maamar Z. (2011) Business Artifacts for E-Business Interoperability, In: E. Kajan (Ed.), Electronic Business Interoperability: Concepts, Opportunities and Challenges, IGI Global, pp. 15-36. doi:10.4018/978-1-60960-485-1.ch002.

Baida Z. (2006) Software-aided Service Bundling Intelligent Methods & Tools for Graphical Service Modeling. Ph.D thesis. Amsterdam, Netherlands : Vrije Universiteit, 301 p. ISBN-10: 90-810622-1-2.

Baina S. (2006) Interoperabilité Dirigée par les Modèles: Une Approche Orientée Produit pour l'interopérabilité des systèmes d'entreprise. Thèse de doctorat de l'Université Henri Poincarè. Nancy, France: l'Université Henri Poincarè.

Balin S. and Giard V. (2006) A Process Oriented Approach to the Service Concepts. In: Proceedings IEEE SSSM06 (Service Systems and Service Management), October 25-27, Troyes, France. Washington. DC: IEEE Computer Society, pp. 785 – 790.

Barras R. (1986) Towards a Theory of Innovation in Services. Research Policy, 15(4), pp. 161-173.

Bénaben F., Touzi J., Rajsiri V. et al. (2006) Collaborative Information System Design. In: Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.), AIM 2006 - Information Systems and Collaboration: State of the Art and Perspectives, Best Papers of the 11th International Conference of the Association Information and Management (AIM), Luxembourg, June 8-9, 2006. LNI 92 GI 2006, pp. 281-296. ISBN 978-3-88579-186-7

Bernstein A. and Kaufmann E. (2006) GINO - A Guided Input Natural Language Editor. In: Cruz, I.; Decker, S.; Allemang, D.; Preist, C.; Schwabe, D.; Mika, P.; Uschold, M.; Aroyo, L. (Eds.), 5th International Semantic Web Conference (ISWC'06), November 5-9, Athens GA, ETATS-UNIS. Springer, 4273, pp. 144-157.

Berry B.A. and Broduer K. (1999) Quality Values, Market Dominance Through Customer. Satisfaction, San Diego, CA: Aquarius Publishing.

Bettencourt L.A., Ostrom A. L., Brown S. W. et al. (2002) Client Co-Production in Knowledge-Intensive Business Services'. California Management Review, 44(4), pp. 100-128.

Bhattacharya K. et al. (2007) Artifact-centered operational modeling: lessons from customer engagements. IBM Syst. J., 46(4), pp. 703-721.

Biege S., Lay G., Zanker C. et al. (2011), Challenges of Measuring Service Productivity in Innovative, Knowledge-Intensive Business Services. In: Walter Ganz, Florian Kicherer, and Alexander Schletz (Eds.), RESER 2011 Productivity of Services Next Gen - Beyond Output / Input, September 7-10, Hamburg, Germany. Stuttgart : Fraunhofer Verlag. ISBN 9783839602980

Bitran G. and Pedrosa L. (1998) A Structured Product Development Perspective for Service Operations. European Management Journal, 16(2), pp.169-189.

Bostrom R. and Heinen J.S. (1977) MIS Problems and Failures: A Socio-technical Perspective. MIS Quart, 1(3), pp. 17–32.

BPAP (2007) Offshoring and Outsourcing, Philippines Roadmap 2010, Business Processing Association of the Philippines (BPAP), Makati City. "http://bpap.org/bpap/index.asp?roadmap", last visited December 2011.

BPEL4People (2007) BPEL4People. "http://xml.coverpages.org/bpel4people.html". last accessed in December 2011.

Business Rule Group (2001) Business Rules Group, Defining Business Rules ~ What are They really? "http://www.businessrulesgroup.org/first_paper/br01c0.htm". lasted accessed in December 2011.

Cartelli A. (2007) Socio-Technical Theory and Knowledge Construction: Towards New Pedagogical Paradigms? Issues in Informing Science and Information Technology, 4(27), pp. 1-14.

Camarinha-Matos L. M. and Afsarmanesh H. (2006) Collaborative Networks Value Creation in a Knowledge Society. In: Proceedings of PROLAMAT'06, June 15-17, Shanghai, China. Springer, pp. 1-14.

Casati F. et al. (2000) Adaptive and dynamic service composition inEFlow. In: Proceedings of 12th International Conference on Advanced Information Systems Engineering(CAiSE), June 5-9, Stockholm, Sweden. Berlin: Springer-Verlag, pp. 13-31.

Ceravolo P. et al. (2007) Modeling Semantics of Business Rules. In: Proceeding of the Inaugural IEEE International Conference on Digital Ecosystems and Technologies (IEEE-DEST), February 21-23, Cairns, Australia. IEEE press, pp. 171-176.

Chase R.B. (1978) Where Does the Customer Fit in a Service Operation? Harvard Business Review, November-December, 56(6), pp. 137-142.

Chau M., Ball G., Huang J., Chen J. et al. (2011) Global IT and IT-enabled services. Inf Syst Front, 13 (3) pp. 301–304.

Christensen C. and Tan A. (2000) Developing Ideas for Innovative Products. Masters Thesis in Mechanical Engineering. Lyngby : Technical University of Denmark, 43 p.

Clark G., Johnston R. and Shulver M. (2000) Exploiting the Service Concept for Service Design and Development. In: Fitzsimmons, J., Fitzsimmons, M. (Eds.), New Service Design, Sage: Thousand Oaks, CA, pp. 71-91.

Cohn D. and Hull R. (2009) Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. IEEE Data Engineering, 32(3), pp. 3-9.

Col P. and Beveridge T. (2003) Guide to Enterprise IT Architecture. New York, NY: Springer. ISBN: 0-387-95132-6.

Congram C. and Epelman M. (1995) How to Describe Your Service Paper. International Journal of Service Industry Management, 6(2), pp. 6-23.

de Man H. (2009) Case Management: A Review of Modeling Approaches. "http://www.bptrends.com/publicationfiles/01-09-ART-%20Case%20Management-1-DeMan.%20doc--final.pdf". last accessed in October 2011.

de Mazancourt C. et al. (2005) Understanding Mutualism When There is Adaptation to the Partner. Journal of Ecology, 93(2), pp. 305-314.

de Moor A. (2006) Community Memory Activation with Collaboration Patterns. In: L. Stillman & G. Johanson (Eds.), Proceedings 3rd International Community Informatics Research Network (CIRN) Conference, October 9-11, Prato, Italy. Italy: Monash University, pp. 1-22.

Demuth B. and Liebau H. B. (2007) An Approach for Bridging the Gap Between Business Rules and the Semantic Web. In: Proceedings of RuleML, Orlando, Florida, pp. 119-133.

Den Hertog P. (2000) Knowledge-Intensive Business Services as Co-Producers of Innovation. International Journal of Innovation Management, 4(4), pp. 491-528.

de Ruyter K., Wetzels M. and Kleijnen, M. (2001) Customer Adoption of e-service: an Experimental Study. International Journal of Service Industry Management, 12(2), pp. 184–207.

Deutsch A., Hull R., Patrizi F. et al. (2007) Automatic Verification of Data-Centric Business Processes. In: Proceedings of the 12th International Conference on Database Theory (ICDT), March 23-25, St. Petersburg, Russia. New York : ACM , pp. 252-267.

de Vries E.J. (2006) Innovation in Services in Networks of Organizations and in the Distribution of Services. Research Policy, 35(7), pp. 1037-1051.

Donzelli P. (2004) A Goal-driven and Agent-based Requirements Engineering Framework. Requirements Eng, 9(1), pp. 16–39.

DRS (2011) DRS Technical Report. "http://attempto.ifi.uzh.ch/site/docs/". lasted accessed in December 2011.

Du A.Y., Gopal R.D. and Ramesh R. (2005) Value Chains in IT-Enabled Services: Towards a Reference Model. In K. Kumar and S. Krishna (Eds), 1st International Conference on Management of Globally Distributed Work, 28-30 December 2005, Bangalore, India. India: GDW (Globally Distributed Work) Consortium, 25 p.

Durand F. (2010), Mapping SBVR and F-logic: An Approach for Modelling Ontologies using Business Rule Paradigms. Thesis in Knowledge Management. Karlsruhe : Karlsruhe Institute of Technology, 100 p.

Edvardsson B. and Olsson J. (1996) Key Concepts for New Service Development. The Services Industries Journal, 16(2), pp. 140-164.

Edvardsson B., Gustafsson A. and Roos I. (2005) Service Portraits in Service Research: a Critical Review. International Journal of Service Industry Management, 16(1), pp.107 – 121

EIA/IS 632 (1994) Systems Engineering. Washington, DC: Electronic Industries Association (EIA).

Eiglier P. and Langeard E. (1975) Une nouvelle Approche Nouvelle pour le Marketing des Services. Revue Française de Gestion, Spring, (2), pp. 97-114.

Elvesæter B., Hahn A., Berre A. and Neple T. (2006) Towards an Interoperability Framework for Model-Driven Development of Software Systems. In: Konstantas, D.; Bourrières, J.-P.; Léonard, M.; Boudjlida, N. (Eds.), Proceeding of 1st international conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA05), February 23-25, Geneva, Switzerland. Springer, pp. 409-420.

Fact++ (2011) Fact++ Reasoner. "http://code.google.com/p/factplusplus/". lasted accessed in December 2011.

FEA Document (2006) FEA Consolidated Reference Model Document Version 2.1. December 2006, published by the Federal Enterprise Architecture Program Management Office, Office of Management of Budget.

FEAF (1999) Federal Enterprise Architecture Framework Version 1.1. "http://www.cio.gov/documents/fedarch1.pdf". lasted accessed in December 2011.

FEA Practice Guide (2006) FEA Practice Guidance. "http://www.whitehouse.gov/sites/default/files/omb/assets/fea_docs/FEA_Practice_Guidance_Nov_2007.pdf". lasted accessed in December 2011.

Fitzsimmons J.A. and Fitzsimmons M.J. (2004) Service management: Operations, Strategy and Information Technology. McGraw-Hill: London.

Free Dictionary (2011) Definition of e-service. "http://encyclopedia2.thefreedictionary.com/e-services". last visited in September 2011.

Fritz C., Hull R. and Su J. (2009) Automatic Construction of Simple Artifact-based Workflows. In: Proceedings of the 12th International Conference on Database Theory (ICDT), March 23-25, St. Petersburg, Russia. New York, NY: ACM Press, pp. 225-238.

Frolund S. and Koistinen J (1998) QML: A Language for Quality of Service Specification. Technical Report HPL-98-10, Palo Alto. California : Hewlett-Packard, 63 p.

Fuchs N.E. and Schwitter R. (1996) Attempto Controlled English (ACE), In:, First International Workshop on Controlled Language Applications (CLAW 96), March 26-27, University of Leuven, Belgium. "http://arxiv.org/PS_cache/cmp-lg/pdf/9603/9603003v1.pdf", lasted accessed in December 2011.

Fujii K. and Suda T. (2009) Semantics-based Context-aware Dynamic Service Composition. ACM Trans. Auton. Adapt. Syst., 4(2), pp. 1-31.

Gall N., Newman D., Allega P. et al. (2010) Introducing Hybrid Thinking for Transformation, Innovation and Strategy. White paper, ID Number: G00172065. Gartner Group.

"http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1352013". lasted accessed in December 2011.

Gallouj F. and Weinstein O. (1997) Innovation in Services. Research Policy, Elsevier, 26(4-5), pp. 537-556.

Gartner Group (2009) Gartner Identifies New Approach for Enterprise Architecture. "http://www.gartner.com/it/page.jsp?id=1124112". lasted accessed in December 2011.

Gartner Group (2010a) Gartner Says Hybrid Thinking for Enterprise Architecture Can Help Organisations Embrace Transformation, Innovation and Strategy "http://www.gartner.com/it/page.jsp?id=1368613". last accessed in December 2011.

Gartner Group (2010b) Gartner Predicts 95 Percent of Organisations Will Support Multiple Approaches to Enterprise Architecture by 2015. "http://www.gartner.com/it/page.jsp?id=1358913". lasted accessed in December 2011.

Glushko R. J. (2010) Seven Contexts for Service System Design, in Maglio, P. P., Kieliszewski, C, and Spohrer, J (Eds.), Handbook of Service Science. New York: Springer, pp. 219-249.

Goldkuhl G. and Rostlinger A. (2000) Beyond goods and services - an elaborate product classification on pragmatic grounds, In: proceedings of The Seventh International Research Symposium on Service Quality (QUIS 7), July 13-16, Karlstad university, Sweden. Emerald Group Publishing, 10 p.

Goldstein S.M., Johnston R., Duffy J. et al. (2002) The Service Concept: the Missing Link in Service Design Research? Journal of Operations Management, 20(2), pp. 121-134.

Gou J. Q. et al. (2008) On-demand Integration of Digital Service System. In: Service Operations and Logistics, and Informatics (IEEE/SOLI 2008), October 12-15, Beijing, China. IEEE Xplore, pp. 2774-2778.

Govindarajan K., Karp A., Kuno H. et al. (2001) Conversation Definitions: Defining Interfaces of Web Services. In HP Position Papers for the World Wide Web Consortium (W3C) Workshop on Web Services, Hewlett-Packard report HPL-2001-73. HP : Hewlett-Packard.

"http://www.hpl.hp.com/techreports/2001/HPL-2001-73.html", last visited in December 2011.

Grandison T. and Thomas J.O. (2008) Asset Reuse and Service Science: The Delicate Balance. PICMET 2008 Proceedings, July 27-31, Cape Town, South Africa, pp. 2384-2389.

Gregor H. and Woolf B. (2003) Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, 683 p. ISBN 0-321-20068-3

Groeneweg J. and van de Kar E. (2007) A Service System Design Approach for iTV Banking. In: Gini, Maria L., Kauffman, Robert J., Sarppo, Donna, Dellarocas, Chrysanthos and Dignum, Frank (eds.) Proceedings of the 9th International Conference on Electronic Commerce - ICEC 2007 August 19-22, 2007, Minneapolis, MN, USA. ACM Press, pp.369-378.

Gronroos C. (1990) Service Management and Marketing： Managing the Moments of Truth in Service Competition. USA: Lexington Books, Massachusetts, Massachusetts/Toronto, pp. 71-240.

Gummesson E. (1994) Making Relationship Marketing Operational. International journal of service industry management, 5(5), pp. 5-20.

Haase P., Hitzler P., Krtzsch M. et al. (2006) Practical Reasoning with OWL and DL-safe Rules. Tutorial in Half-day tutorial at the 3rd European Semantic Web Conference (ESWC)'2006.

Hamdouch A. and Moulaert F. (2006) Knowledge Infrastructure, Innovation Dynamics, and Knowledge Creation, Diffusion, Accumulation Processes: A comparative institutional perspective. The European Journal of Social Science Research, 1469-8412, 19(1), pp. 25-50.

Hax and Majluf (1991) The Strategy concept and Process: A Pragmatic Approach. Englewood Cliffs, NJ: Prentice Hall.

Hendryx S. (2005) Model-Driven Architecture and the Semantics of Business Vocabulary and Business Rules. "http://www.semanticcore.org/Docs/MDA%20and%20SBVR%20v2.pdf". lasted accessed in October 2011.

Hermit (2011) HermiT Reasoner. "http://hermit-reasoner.com/". lasted accessed in December 2011.

Hill T.P. (1977) On Goods and Services. Review of Income and Wealth, 23(4), pp. 315–338.

Horrocks I. et al. (2004) SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C. "http://www.w3.org/Submission/SWRL/". lasted accessed in December 2011.

Hull R. (2008) Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: Robert Meersman, Zahir Tari (Eds.): On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I. Lecture Notes in Computer Science 5331 Springer 2008, pp. 1152-1163. ISBN 978-3-540-88870-3

IBM (2004), Service Science: a New Academic Discipline? "http://www.almaden.ibm.com/asr/resources/facsummit.pdf ". lasted accessed in December 2011.

ISO (1991) International Organisation for Standardisation (ISO, 9004-2, 1991), Quality Management and Quality Systems Element- Part 2 Guidelines for services.

Issa H., Assi C. and Debbabi M (2006) QoS-Aware Middleware for Web Services Composition - A Qualitative Approach. In: ISCC Proceedings of the 11th IEEE Symposium on Computers and Communication, June 26-29, Cagliari, Sardinia, Italy. Washington, DC : IEEE Computer Society pp. 359-364.

ITIL V3 (2007) The Definition of Business Service. "http://www.best-management-practice.com/gempdf/ITIL_Glossary_V3_1_24.pdf". last accessed in December 2011.

Izumi N. et al. (2009) Service System Development Based on Web Process Ontology. In: Kenneth Boness, João M. Fernandes, Jon G. Hall, Ricardo Jorge Machado, Roy Oberhauser (Eds.): The Fourth International Conference on Software Engineering Advances, ICSEA 2009, 20-25 September 2009, Porto, Portugal. IEEE Computer Society, pp. 222-228. ISBN 978-0-7695-3777-1

Jackson M.C. (2003) Systems Thinking: Creative Holism for Managers. Chichester, UK: John Wiley & Sons, pp. 1-352, ISBN 0 470 84522 8, 24.95

Jasper R. and Uschold M. (1999) A Framework for Understanding and Classifying Ontology Applications. In: Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Methods(KRR5), August 2, Stockholm, Sweden, 37 p.

Javalgi R.G., Martin C.L. and Todd P.R. (2004) The Export of E-services in the Age of Technology Transformation. Journal of Services Marketing, 18(7), pp. 560-573.

Jurisica I., Mylopoulos J. and Yu E. (1999) Using Ontologies for Knowledge Management: An Information Systems Perspective, In: Proceedings of the 62nd Annual Meeting of the American Society for Information Science(ASIS '99), Oct. 31 – Nov 4, Washington, D. C. Medford : American Society for Information Science by Information Today, pp. 482-496;

Kalakota R. and Robinson M. (2003) Services Blueprints: Roadmap for Execution. Boston: Addison Wesley, pp. 1-384. ISBN: 0-321-15039-2

Kaon (2011) Kaon Ontology Editor, "http://kaon.semanticweb.org/". lasted accessed in December 2011.

Kaplan D. (2000) Structural Equation Modeling: Foundations and Extensions. SAGE Advanced Quantitative Techniques in the Social Sciences series, 10, pp. 1-272, Newbury Park, Calif.: Sage Publications, ISBN 0-7619-1407-2.

Kattenstroth H. et al. (2007) Combining OWL with F-Logic Rules and Defaults. In Axel Polleres, D. Pearce, Edna Ruckhaus, Gopal Gupta, Stijn Heymans (Eds), Proc. ALPSWS, September 13, Porto, Portugal. CEUR Workshop Proceedings, 287, pp. 60-75.

Kautz H., Selman B. and Coen M. (1994) Bottom-Up Design of Software Agents. Communication of the ACM, 37(7), pp. 143-147.

Kethers S. and Schoop M. (2000) Reassessment of The Action Workflow Approach: Empirical Results. In: M. Schoop and C. Quix (eds). Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling, LAP 2000. September 2000, Aachen, Germany. Aachener Informatik Berichte, pp. 151-169.

Kim Y.J. and Nam K (2009) Service Systems and Service Innovation: Toward the Theory of Service Systems. In: Proceedings of the Fifteenth Americas Conference on Information Systems, August 6-9, San Francisco, California. Atlanta : Association of Information Systems (AIS), 2009, pp. 1-6.

Ko R. K. L. et al. (2009) Business Process Management (BPM) Standards: A Survey. Business Process Management Journal, 15(5), pp. 744 – 791.

Kosanke K. (2005) ISO Standards for Interoperability: a Comparison. In: Konstantas, D.; Bourrières, J.-P.; Léonard, M.; Boudjlida, N. (Eds.), Proceeding of 1st international conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA05), February 23-25, Geneva, Switzerland. Springer-Verlag, pp. 55-64.

Kotler P. and Bloom P.N. (1984) Marketing Professional Services. Englewood Cliffs, NJ.,: Prentice-Hall.

Krange I. et al. (2002) Describing Construction of Knowledge Through Identification of Collaboration Patterns in 3D Learning Environments. In: G. Stahl (Ed.), Proceedings of CSCL 2002, January 7-11, Boulder, Colorado, USA. Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates, pp 82-91.

Krishna V. et al. (2007) Intelligent Document Gateway - A Service System Analysis. In: IEEE International Conference on Services Computing (SCC 2007), July 9-13, Salt Lake City, Utah, USA. Piscataway, N.J. : IEEE Xplore, pp. 636-643.

Kumaran S., Liu R., and Wu F.Y. (2008) On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'2008), June 18-20, Montpellier, France. Berlin: Springer-Verlag, pp. 32-47.

Lancaster K.J. (1966) A New Approach to Consumer Theory. Journal of Political Economy, 74(2), pp. 133-157.

Lammermann S. (2002). Runtime Service Composition via Logic-Based Program Synthesis. Ph.D thesis. Kista: Department of Microelectronics and Information Technology, Royal Institute of Technology, 188 p.

Late Middle English (2011) The Definition of Method. "http://dictionary.reference.com/browse/method". last accessed in December 2011.

Lauras M., Parrod N. and Telle O. (2003) Proposition de référentiel pour la notion d'entente industrielle: trois approches dans le domaine de la gestion des chaînes logistiques. Revue Française de Génie Industriel, 22(4), pp. 5-29.

Lee M.J. (1996) Foundations of the Win-Win Requirements Negotiation System. Ph.D. Thesis. California : University of South California, 192 p.

Leist S. and Zellner G. (2006) Evaluation of Current Architecture Frameworks. In: ACM Symposium on Applied Computing, April 23-27, Dijon, France. France: ACM, pp. 1547-1553.

Lengrand L. et al. (2002) Innovation Tomorrow, Innovation Policy and the Regulatory Framework: Making Innovation an Integral Part of the Broader Structural Agenda. A report for the DG Enterprise produced by Lois Legrand & Associes, Prest (University of Manchester) and ANRT France. Innovation papers, Luxemburg: Directorate-General for Enterprise, European Commission, (28), ISBN: 9289445491.

Lienhard H. and Künzi U. M. (2005) Workflow and Business Rules: a Common Approach, BPTrends September 2005. "http://www.bptrends.com/publicationfiles/09-05%20WP%20Workflow%20and%20Business%20Rules%20%20Lienhard%20-%20Kunzi.pdf". last accessed in October 2011.

Lovelock C. (2001) Services Marketing: People, Technology and Strategy. 2nd edition, Upper Saddle River: Prentice Hall, pp. 1- 717, ISBN: Number = 0-13-544701-1.

Lovelock C. H. and J. Wirtz (2004) Services Marketing: People, Technology, Strategy. 5th ed. Upper Saddle River, NJ: Pearson Prentice Hall.

Maglio P.P, Srinivasan S., Kreulen J.T. et al. (2006) Service Systems, Service Scientists, SSME, and Innovation. Communications of the ACM, 49(7), pp. 81-85.

Martin D., et al. (2004) OWL-S: Semantic Markup for Web Services. Technical Report, Member Submission, W3C. "http://www.w3.org/Submission/OWL-S/", last accessed in December 2011.

Martyn A.O. (2005) Business Process Management: A Rigorous Approach. Meghan-Kiffer Press, BCS (North America), 338 p.

Manes A.T. (2011) Service Modeling: Making Sure Services Deliver Value. In: 4th International SOA Symposium & 3rd International Cloud Symposium, April 27-28, Brasilia, Brazil. Gartner Group.

Mager B. (2008) Service Design. In M. Erlhoff & T. Marshall (Eds.), Design Dictionary: Perspectives on Design Terminology, Basel: Birkhäuser, pp. 354-357.

Maglio P.P. and Spohrer J. (2008) Fundamentals of Service Science. Journal of the Academy of Marketing Science, 36(1), pp. 18-20.

Mai T.V. and Bosch O.J.H. (2009) Systems Thinking Approach as a Unique Tool for Sustainable Tourism Development: A Case Study in the Cat Ba Biosphere Reserve of Vietnam. In: Proceedings of the 54th Meeting of the International Society for the Systems Sciences(ISSS), Wilfrid Laurier University, Waterloo, ON, Canada. Waterloo : ISSS, pp. 1444-1463.

Martinez-Fernandez C., Soosay C.A., Bjorkli M. et al. (2004) Are Knowledge-Intensive Service Activities Enablers of Innovation Processes? A study of Australian Software Firms. In: the 5th International Proceedings of Continuous Innovation: Strategic Priorities for the Global Knowledge Economy, September 22-25, Sydney, Australia. Adelaide: Causal Productions, pp. 986-1000.

Merriam-Webster (2006) The Definition of Paradigm. "http://www.merriam-webster.com/dictionary/paradigm". last accessed in December 2011.

Meyer K. and Fahnrich K.P. (2009) How to 'Engineer' IT-Enabled Services. In: Proceedings First International Symposium on Services Science ISSS'09, March 23-25, Leipzig, Germany. Berlin : Logos Verlag, 1999, pp. 137-148.

Miles I. (2003) Knowledge Intensive Services' Suppliers and Clients. In: Studies and Reports of Ministry of Trade and Industry of Finland, Helsinki. Finland: Ministry of Trade and Industry of Finland, pp. 1-81, ISBN 951-739-740-2

Miles I., Kastrinos N., Flanagan K. et al. (1995) Knowledge-Intensive Business Services: Users, Carriers and Sources of Innovation. A report to DG13 SPRINT-EIMS, Luxembourg: EC, "http://research.mbs.ac.uk/INNOVATION/Portals/0/docs/KIBSEIMS1995shortreport.pdf", lasted accessed in December 2011.

Miozzo, M. and Grimshaw, D. (2005) Modularity and innovation in knowledge-intensive business services: IT outsourcing in Germany and UK. Research Policy, 34 (9), pp 1419-1439.

Mora M. et al. (2011) Onto-ServSys: A Service System Ontology. In: H. Demirkan et al. (eds.), the Science of Service Systems, Service Science: Research and Innovations in the Service Economy, Springer Science+Business Media, LLC, pp. 151-173.

Morelli N. (2006) Developing new PSS, Methodologies and Operational Tools. Journal of Cleaner Production, 14(17), pp. 1495–1501.

Moritz S. (2005). Service Design: Practical Access to an Evolving Field. "http://stefan-moritz.com/welcome/Service_Design_files/Practical%20Access%20to%20Service%20Design.pdf". last visited December 2011.

MIT (2011) MIT Open Course. "http://ocw.mit.edu/index.htm". lasted accessed in December 2011.

Mo T. et al. (2007) A Service System Theory Frame Based on Ecosystem Theory. Wireless Communications. In: Networking and Mobile Computing, WiCom International Conference, September 21-25, Shanghai, China. IEEE, pp. 3184-3187.

Mo T. et al. (2009) A Virtualization-Based Service System Development Method. Journal of Service Science and Management (JSSM), 2(1), pp. 1-9.

Motik B., Grau B. C., Horrocks I. et al. (2009) Representing Ontologies Using Description Logics, Description Graphs, and Rules. Artif. Intell, 173(14), pp. 1275-1309.

Mullera E. and Doloreuxb D. (2009) What We Should Know about Knowledge-intensive Business Services. Technology in Society, 31(1), pp. 64-72.

Muller D., Reichert M. and Herbst J. (2008) A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. In: Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'2008), Montpellier, France. Berlin: Springer-Verlag, pp. 48-63.

Narendra N.C., Badr Y., Thiran P. and Maamar Z. (2009) Towards a Unified Approach for Business Process Modeling Using Context-based Artifacts and Web Services. In: Proceedings of the IEEE International Conference on Services Computing (SCC'2009), September 21-25, Bangalore, India. Washington, DC : IEEE Computer Society, pp.332-339.

Nigam A. and Caswell N. S. (2003) Business Artifacts: An Approach to Operational Specification. IBM Systems Journal, 42(3): 428- 445.

Nitto E.D., Ghezzi C., Metzger A. et al. (2008) A Journey to Highly Dynamic, Self-adaptive Service-based Applications. Automated Software Engineering, 15(3-4), pp. 313-341.

Nof S.Y. (2007) Collaborative Control Theory for e-Work, e-Production, and e-Service. Annual Reviews in Control, 31(2), pp. 281-292.

Noy N.F. and McGuinness D.L. (2001) Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report SMI-2001-0880. Toronto: University of Toronto, 25 p.

OCL (2000) Object Constraint Language Specification, Chapter 7 of OMG Unified Modeling Language Specification, Version 1.3, March 2000 (first edition).

OWL2 (2009) OWL 2 Web Ontology Language Document Overview. "http://www.w3.org/TR/owl2-overview/". last accessed in December 2011.

OWL2API (2011) OWL2API. "http://owlapi.sourceforge.net/". lasted accessed in December 2011.

Papageorgiou N., Verginadis G., Apostolou D. et al. (2009) A Collaboration Pattern Model for Virtual Organisations., in Luis M. Camarinha-Matos; Iraklis Paraskakis & Hamideh Afsarmanesh (eds.), 'PRO-VE', Springer, pp. 61-68.

Parag Y. and Janda K. (2010) Midstream and Sideways: Considering a Middle-out Approach to Changing Energy Demand. In: Sussex energy group conference, Oxford, U.K., Oxford University, February 2010, Proceedings: Oxford, United Kingdom, Environmental Change Institute, 16 p.

Parasuraman A., Berry L.L. and Zeithaml V.A. (1991) Understanding Customer Expectations of Service. Sloan Management Review, 32(3), pp.39-48.

Parsia B., Sirin E., Grau B.C. et al. (2005). Cautiously Approaching SWRL. Technical report, University of Maryland.

Pellet (2011) Pellet Reasoner. "http://clarkparsia.com/pellet/". lasted accessed in December 2011.

Peng Y., Badr Y. and F. Biennier (2009) A Generic Service System for Knowledge-Intensive Service Firms. In: Richard Chbeir, Youakim Badr, Epaminondas Kapetanios, Agma J. M. Traina (Eds.), MEDES '09: International ACM Conference on Management of Emergent Digital EcoSystems, October 27-30, Lyon, France. ACM 2009, pp. 422-426. ISBN 978-1-60558-829-2

Peng Y., Badr Y. and F. Biennier (2010a) Dynamic Representation Model for Service Systems, In: Service Sciences (ICSS), 2010 International Conference on, 13-14 May 2010, Hangzhou, China. IEEE, pp. 335 – 340.

Peng Y., Badr Y. and F. Biennier (2010b) Design Data-driven Collaboration in Service Systems, In: New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on, 11-13 May 2010, Gyeongju, Korea. IEEE, 2010, pp. 47-53

Pinhanez C. (2008) Service Systems as Customer-Intensive Systems and Its Implications for Service Science and Engineering. In: HICSS '08 Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences, January 7-10, Waikoloa, HI. Washington DC: IEEE Computer Society, p. 117.

Pohl K. (1993) The Three Dimensions of Requirements Engineering. In Rolland C., Bodart F., Cauvet C. (Eds), Fifth International Conference on Advanced Information Systems Engineering (CAiSE'93), June 8-11, Paris, France. Paris: Springer-Verlag, pp.175-292.

Ponsignon F., Smart P.A. and Maull R.S. (2007) Service Delivery Systems: the Transformational Context., Discussion Papers in Management, Paper number 07/17. University of Exeter. ISSN 1472-2939

Protégé 4 (2011) The Protégé Ontology Editor and Knowledge Acquisition System framework. "http://protege.stanford.edu/". lasted accessed in December 2011.

ProtégéOWLAPI (2011) ProtégéOWLAPI. "http://protege.stanford.edu/plugins/owl/api/". lasted accessed in December 2011.

Qiu et al. (2007) Editorial: Towards Service Science, Engineering and Practice. International Journal of Services Operations and Informatics, 2(2), pp. 103-113.

Qiu (2009) Computational Thinking of Service Systems: Dynamics and Adaptiveness Modeling. Service Science, 1(1), pp. 42-55.

Quinn J.B., Baruch J.J. and Paquette P.C. (1987) Technology in Services. Scientific American, 257(6), pp. 50-58.

R2ML (2008) R2ML Portal. "http://hydrogen.informatik.tu-cottbus.de/wiki/index.php/Portal:R2ML", lasted accessed in December 2011.

Raccoon L.S.B. and Puppydog P.O.P. (1998) A Middle-out Concept of Hierarchy (or the Problem of Feeding the Animals). SIGSOFT Softw. Eng, 23(3), pp. 111-119.

Räisänen V. et al (2005) Service Management Evolution. In : 14th IST Mobile and Wireless Communications Summit, June 19-23, Dresden, Germany.

Ramesh B. and Jarke M (2001) Toward Reference Models for Requirements Traceability, IEEE Trans. Softw. Eng, 27 (1), pp. 58 -92.

Rathmell J. M. (1966) What Is Meant by Services? Journal of Marketing, 30(4), pp. 32-36.

Rempel P., Katt B. and Breu R. (2010) Supporting Role Based Provisioning with Rules Using OWL and F-Logic. In: Robert Meersman, Tharam S. Dillon, Pilar Herrero (Eds.): On the Move to Meaningful Internet Systems:

OTM 2010 - Confederated International Conferences: CoopIS, IS, DOA and ODBASE, Hersonissos, Crete, Greece, October 25-29, Proceedings, Part I. Lecture Notes in Computer Science 6426 Springer 2010, pp.600-618. ISBN 978-3-642-16933-5,

Rittgen P. (2006) Supporting Planned and Ad-Hoc Changes of Business Process. In: Gil Regev, Pnina Soffer, Rainer Schmidt (Eds.), Proceedings of the CAISE06 Workshop on Business Process Modelling, Development, and Support BPMDS '06, Luxemburg, June 5-9, 2006. CEUR Workshop Proceedings 236 CEUR-WS.org 2007.

Rolland C. (1993) Modeling the Requirements Engineering Process. In: 3rd European-Japanese Seminar on Information Modelling and Knowledge Bases, May 31- June 3, Budapest, Hungary. Amsterdam : IOS Press, pp. 86-97.

Roger Sessions (2007) Comparison of the Top Four Enterprise Architecture Methodologies. "http://msdn.microsoft.com/en-us/library/bb466232.aspx". lasted accessed in December 2011.

Rooze E.J., Paapst M. and Sombekke J. (2007) eCase Management. An International Study in Judicial Organizations. "http://www.rechtspraak.nl/NR/rdonlyres/67AEE8E1-7A06-4E05-BB64-C64245D13049/0/84120052_eCaseManagement.pdf". last accessed in October 2011.

Roth A.V. and Menor L.J. (2003) Insights into Service Operations Management: a Research Agenda. Production & Operations Management, 12(2) pp. 145-164.

Royce W.W. (1970). Managing the Development of Large Software Systems. In: Proceedings of IEEE WESCON, August, Los Angeles. IEEE Press, 26, pp. 328-338.

RuleML (2011) RuleML. "http://ruleml.org/". lasted accessed in December 2011.

Rust R.T. and Kannan P. (2003) E-service: a New Paradigm for Business in the Electronic Environment. Communications of the ACM, 46(6), pp. 36–42.

Rust R. (2004) A Call for a Wider Range of Service Research. Journal of Service Research, 6 (3), pp. 211.

SaaS (2008) Saas Data Architecture, an Oracle White Paper. "http://www.oracle.com/technology/tech/saas/pdf/saas-data-architecture-whitepaper.pdf/". last accessed in October 2011.

Sabatier P. (1986) Top-Down and Bottom-Up Approaches to Implementation Research: a Critical Analysis and Suggested Synthesis. Journal of Public Policy, 6(1), pp. 21-48.

Sampson S.E. and Froehle C.M. (2006) Foundations and Implications of a Proposed Unified Services Theory. Production and Operations Management, 15(2), 2006, pp. 329-343.

Sampson S.E. (2007) (Why we need) An Operations Paradigm for Services. In: POMS/CSO Conference, July 12-13, London business school, U.K.

Sarnikar S. and Zhao J.L. (2007) Pattern-based Knowledge Workflow Automation: Concepts and Issues. Information Systems and E-Business Management, 6(4), pp. 1-18.

Sasser W.E., Olsen R.P. and Wyckoff D. (1978) Management of Service Operations. Boston, MA: Allyn & Bacon.

Saviotti P.P. and Metcalfe J.S. (1984) A Theoretical Approach to the Construction of Technological Output Indicators. Research Policy, 13 (3), pp. 141-151.

Schmitt J.R. (1993) Product Modeling for Requirements Engineering Process Modeling. In: IFIP WG 8.1 Conf. on Information Systems Development Process, September 1-3, Como, Italy. Amsterdam : North-Holland, pp. 231-245.

SBVR (2008) SBVR 1.0 documentation. "http://www.omg.org/spec/SBVR/1.0/". lasted accessed in December 2011.

Science Dictionary (2005) The definition of Model. In: The American Heritage Science Dictionary, Houghton Mifflin Company. "http://www.thefreedictionary.com/model". last accessed in December 2011.

Sector Future (2005) Sector Futures - The knowledge-intensive business services sector.

"http://www.eurofound.europa.eu/emcc/publications/2005/ef0559en.pdf". last visited December 2011.

Shroff V. (2001) Requirements Management Metrics. Calgary, Alberta, Canada: University of Calgary.

Shostack L.G (1984) Design Services that Deliver. Harvard Business Review (84115), 62(1), pp. 133-139.

Shostack G.L. (1987) Service Positioning Through Structural Change. Journal of Marketing, 51(1), pp. 34-43.

Silver M.S., Markus M.L. and Beath C.M. (1995) The Information Technology Interaction Model: A Foundation for the MBA Core Course. MIS Quarterly, Special Issue on IS Curricula and Pedagogy, 19(3), pp. 361-390.

Smedlund A. and Toivonen M. (2007) The Role of KIBS in the IC Development of Regional Clusters. Journal of Intellectual Capital, 8(1), 159-170.

Smith B. (1998) Basic Concepts of Formal Ontology. In: Guarino N (Ed.), Proceedings of Formal Ontology in Information Systems (FOIS'98), 6-8 June Trento, Italy. Amsterdam: IOS Press, pp. 19-28.

Song I.Y., Evans, M. and Park E. K. (1995) A Comparative Analysis of Entity-Relationship Diagrams. J. Comput. Softw, Eng. 3(4), pp. 427– 459.

Southern G. (1999) A Systems Approach to Performance Measurement in Hospitality. International Journal of Contemporary Hospitality Management, 11(7), pp. 366-376.

Sowa J.F. and Zachman J.A. (1992) Extending and Formalizing the Framework for Information Systems Architecture. IBM Systems Journal, 31(3), pp.590-616.

Spohrer J. and Maglio P.P. (2008) The Emergence of Service Science: Toward Systematic Service Innovations to Accelerate Co-Creation of Value. Production and Operations Management, 17(3), pp. 238-246.

Spohrer J., Maglio P.P, Bailey J. et al. (2007) Steps toward a Science of Service Systems. IEEE. Computer, 40 (1), pp. 71-77.

Spohrer J., Vargo S., Maglio P.M. et al. (2008) The Service System Is the Basic Abstraction of Service Science. In: Proceedings of the 41st Hawaii International Conference on System Sciences, January 7-10, Waikoloa, HI. Washington DC: IEEE Computer Society, pp.104.

Stanicek Z. and Winkler M. (2010) Service Systems through the Prism of Conceptual Modeling. Service Science, 2(1/2), pp. 112 – 125.

Stencil Group (2001) Defining Web Services, The Stencil Group. "http://www.perfectxml.com/Xanalysis/TSG/TSG_DefiningWebServices.pdf". last visited in September 2011.

Sudan R., Ayers S., Dongier P. et al. (2010) The Global Opportunity in IT-Based Services Assessing and Enhancing Country Competitiveness. Washington, D.C. : World Bank,, pp. 1-97, ISBN: 978-0-8213-8192-2.

Sue K., Nicola G., Christopher H., et al. (1999) Focus Issue on Legacy Information Systems and Business Process Engineering: a Business Perspective of Legacy Information Systems. Communications of the AIS, 2 (7), pp. 1–27.

SWRL (2011) SWRLLanguageFAQ. "http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ". lasted accessed in December 2011.

SWT (2011) SWT: The Standard Widget Toolkit. "http://eclipse.org/swt/". lasted accessed in December 2011.

Tang F.L., Guo S., Guo M.Y. et al. (2011), Towards Context-Aware Ubiquitous Transaction Processing: a Model and Algorithm. In: IEEE ICC 2011 proceedings, July 05-09, Kyoto, Japan. IEEE Computer Society, pp. 1-5.

Tian C.H. et al. (2008) BEAM: A Framework for Business Ecosystem Analysis and Modeling. IBM Systems Journal, 47(1), pp. 101-113.

Tien J.M. and Berg D. (2003) A Case for Service Systems Engineering. Journal of Systems Science and Systems Engineering, 12(1), pp. 13-38.

Tidwell D. (2000) Web Services: The Web's Next Revolution. IBM Tutorial. "http://www.cn-java.com/download/book/wsbasics-a4.pdf". last accessed in October 2011.

Toffolon C. and Dakhli S. (1999) Requirements Elicitation Process: An Iterative Approach based on the Spiral Model. In: XVIth International Symposium on Computer Information and Sciences (ISCIS'99), 1999 October 18-20, Izmir, Turquie. New York : Elsevier Science Publishers, 13 p.

TOGAF (2009) Welcome to TOGAF Version 9 -- The Open Group Architecture Framework. "http://pubs.opengroZup.org/architecture/togaf9-doc/arch/". lasted accessed in December 2011.

Touzi J., Bénaben F. and Pingaud H. (2007) Interoperability Through Model Based Generation: the Case of the Collaborative IS. In Doumeingts G., Müller J., Morel G., and Vallespir B. (Eds.), Enterprise Interoperability: New Challenges and Approaches. Springer, pp. 407-416.

Touzi J. (2007) Aide à la Conception de Système d'Information Collaboratif Support de l'interopérabilité des Enterprises. Ph.D Thesis of Industrial Engineering. Toulouse : INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE - Ecole des Mines d'Albi Carmaux, 208 p.

Tung W.F. and Yuan S.T. (2007) iDesign: An Intelligent Design Framework for Service Innovation, System Sciences. In: HICSS 40th Annual Hawaii International Conference, January 3-6, Waikoloa, HI. IEEE Computer Society, pp. 64-74.

Udaipur (2000) IT Enabled Services. Paper presented at Udaipur Chamber of Commerce, Udaipur, On August 10, 2000, "http://www.sphconsultants.com/papers/itenabpapr1.pdf". lasted accessed in December 2011.

van de Kar E.A.M. (2004) Designing Mobile Information Services, An approach for Organisations in a Value Network. Ph.D Thesis. Delft : Delft University of Technology, 245 p.

van Der Aalst W. M. P. and Ter Hofstede A. H. M. (2005) YAWL: Yet Another Workflow Language. Information Systems Journal, 30(4), pp. 245-275.

Vargo *S.* and Lusch R. (2004) Evolving to a New Dominant Logic for Marketing. Journal of Marketing, 68, pp. 1-17.

Vargo S., Maglio P. and Akaka M. (2008) On Value and Value Co-Creation: A Service Systems and Service Logic Perspective. *European Management Journal*, 26(3), pp. 145-152.

Vasarhelyi M. and Greenstein M. (2003) Underlying Principles of the Electronization of Business: a Research Agenda. International Journal of Accounting Information Systems, 4(1), 2003, 1–25.

Verginadis Y. et al. (2009a) Collaboration Patterns in Event-Driven Environment for Virtual Organisations. In: Intelligent Event Processing – Association for the Advancement of Artificial Intelligence (AAAI), Spring Symposium 2009, Stanford, USA (2009)

Verginadis Y. et al. (2009b) An Architecture for Collaboration Patterns in Agile Event-Driven Environments. In: Sumitra Reddy (Ed), 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2009, June 29 - July 1,Groningen, The Netherlands. IEEE Compute Society, pp. 227-230.

Wang J. and Kumar A. (2005) A Framework for Document-Driven Workflow Systems. In: Third International Conference on Business Process Management (BPM 2005), September 5-8, Nancy, France. Lecture Notes in Computer Science Business Process Management, Berlin/Heidelberg: Springer, pp. 285-301.

Wall Q. (2006) Understanding the Service Lifecycle within a SOA. "http://dev2dev.bea.com/pub/a/2006/11/soa-service-lifecycle-run.html/". accessed in October 2011.

W3C (2004) The Definition of Web Service, "http://www.w3.org/TR/ws-arch/", last accessed in December 2011.

Wheeler R.D. (1979) Top-down Systems Design. ACM '79 Proceedings of the 1979 annual conference. October 29-31, 1979, Detroit, Michigan. New York : ACM Press, pp. 104. ISBN: 0-89791-008-7

Wolak R., Kalaftis S. and Harris P. (1998) an Investigation into Four Characteristics of Services. Journal of Empirical Generalisations in Marketing Science, 3(2), pp. 22-41.

WSDL 2.0 (2007) Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. "http://www.w3.org/TR/wsdl20/". lasted accessed in December 2011.

WS-HumanTask (2007) WS_Human Task. "http://incubator.apache.org/hise/WS-HumanTask_v1.pdf", last accessed in December 2011.

Zeithaml V.A., Parasuraman A. and Berry L.L. (1985) Problems and Strategies in Services Marketing. Journal of Marketing, 49 (2), pp. 33-46.

Yass A.A. et al. (2010) SSME architecture design in reserving parkingproblems in Malaysia. African Journal of Business Management, 4(18), pp. 3911-3923.

Zachman J.A. (1987) A Framework for Information Systems Architecture. IBM Systems Journal, 26(3), pp. 276-292.

Zeithaml et al. (2006) Services Marketing. 4th edition, New York: McGraw-Hill.

Zhang L.J., Zhang J. and Cai H. (2007) Services Computing, Core Enabling Technology of the Modern Services Industry. Beijing:Tsinghua University Press, pp. 1-452, ISBN: 978-3-540-38281-2.

Appendix A: Examples of Internal Characteristics

// characteristic type or internal characteristics

Type highLevelTeacherType= **ResourceCharacteristics**{

Age : numeric years;

Name: string;

Title: increasing enum{teaching assistant, associate professor, professor, director} with order { teaching assistant< associate professor, associate professor< professor, professor< director};

Major: string;

Availability: numeric;

workingYears : numeric years;

};

Type VideoConferenceType = **TechnologyCharacteristics**{

response time: numeric s;

delay: numeric s;

connectionFailureRate: numeric %;

connectionBlockRate: numeric %;

ConnectionNumber: numeric ;

internetBandwidth: numeric Mbits/s;

};

Type AudioConferenceType = **TechnologyCharacteristics**{

response time: numeric s;

delay: numeric s;

connectionFailureRate: numeric %;

connectionBlockRate: numeric %;

ConnectionNumber: numeric ;

internetBandwidth: numeric Mbits/s;

};

Type VideoType =**ChannelCharacteristics**{

Definition: increasing enum{low, average, high} with order { low<average<high};

SoundQuality: increasing enum{low, average, high} with order { low<average<high};

Authorized : boolean;

Reliability: numeric;

};

Type AudioType= **ChannelCharacteristics**{

Authorized: boolean;

Reliability: numeric;

soundQuanlity: increasing enum{low, average, high} with order { low<average<high};

};
Type TextType= **ChannelCharacteristics**{
Authorized: boolean;
Reliability: numeric;
};
Type OpenCourseType= **ResourceCharacteristics**{
Availability: numeric;
Reliability: numeric;
authoritative: boolean:
};
Type ScolarCourseType = **ResourceCharacteristics**{
{
Reliability: numeric;
Format: set{video, audio, text};
Price: numeric;
accessAuthority: boolearn;
updateAuthority: boolearn;
downloadAuthority: boolearn;
downloadCreditUnit: numeric /100Mbits;
};
Type EmailType = **ChannelCharacteristics**{
responseTime: numeric;
emailLostRate: numeric;
Reliability: numeric;
};
Type IntranetType = **ChannelCharacteristics**{
bandWidth: numeric Mbits/s;
Reliability: numeric;
};
Type SocialLearningToolType = **TechnologyCharacteristics**{
Tools: enum{wiki, blog, podcast, microblog, social network};
Reliability: increasing enum{High, average, low} with order {high>average>low};
Security: increasing enum{High, average, low} with order {high>average>low};
Interface-frendly: boolean;
};
Type FlexibleTeachingPlanType = **ResourceCharacteristics**{
Plan period: numeric times/month;
Public: Boolean;
UpdatingWays: set{student-demanding, teaching-driven};
};
Type OnlineExamType = **ResourceCharacteristics**{

Frequency: numeric times/month
Time: numeric hours;
Result: numeric;
waitingTimeForResult: numeric days;
};
Type ShareResourceWithOtherServiceProviderType = **TechnologyCharacteristics**
{
otherSource: set{MIT OpenCourseWare, China Open Resource for Education, OpenCourseWare Consortium, OpenLearn and OpenCourseWare Universia};
reliability: increasing enum{High, average, low} with order {high>average>low};
authorized: boolean;
};
Type DownloadOnlineResourceType= **TechnologyCharacteristics**{
Authority: boolearn;
Speed: numeric Mbits/s;
amountPerDay: numeric Gbits/day;
};
Type StudyViaVideoGameType = **ResourceCharacteristics**{
Price: numeric euros;
GameQuality: increasing enum{High, average, low} with order {high>average>low};
Enjoyment: increasing enum{High, average, low} with order {high>average>low};
Diversity: increasing enum{High, average, low} with order {high>average>low};
};
Type SelfTestVideoGameType = **ResourceCharacteristics**{
Price: numeric euros;
GameQuality: increasing enum{High, average, low} with order {high>average>low};
Enjoyment: increasing enum{High, average, low} with order {high>average>low};
Diversity: increasing enum{High, average, low} with order {high>average>low};
};
Type OnlineRobotForChattingType = **TechnologyCharacteristics**{
intelligence: increasing enum{High, average, low} with order {high>average>low};
chatWays: set{text, sound};
};
Type OnlineShoppingShortCutType = **TechnologyCharacteristics**{
siteSupport: set{Amazon,Ebay, NewEgg, Taobao,CdisCount};
search: set{keyword};
order: set{price, praise};
searchTime: numeric s;
accuracy: increasing enum{High, average, low} with order {high>average>low};
};
Type OnlineSelfTestType= **ResourceCharacteristics**{

{

Free: boolearn;

answerFollowed: boolearn;

Diversty: increasing enum{High, average, low} with order {high>average>low};

};

…

};

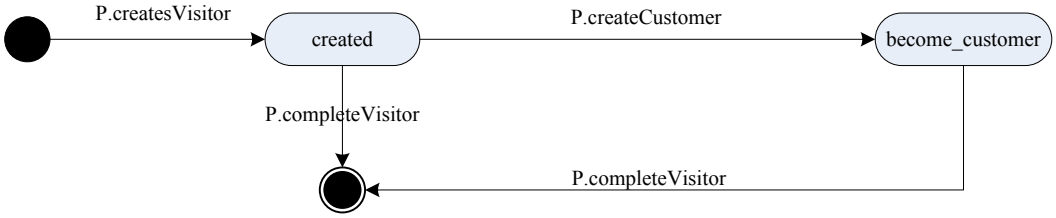
Appendix B: Business Artifact and Domain Rules

Legend:

Artifact	Role	State
cusA: student artifact cerA: certification fexA: final exam ordA: order evaA: evaluation couA: course reiA: returnedItem stoA: stock bilA: bill ori: orderItem colA:courseList shiA: shipment quiA: quiz exrA: exam result tplA: teaching plan	P: provider C: customer T: teacher R: retailer Shi: shipper Sup: supplier	 init  completed

B.1 Visitor Artifact

Attributes: VisitorID,
States: created, become_customer, completed, finished
Lifecycle:

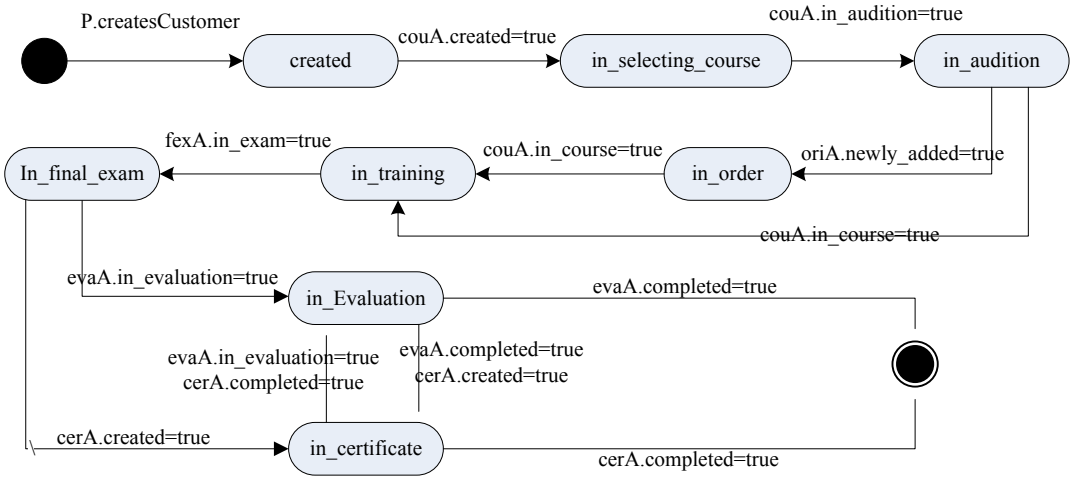


Business rules:
Rule 1:
Events(?Var_Event1), VisitorArtifact(?Var_Visitor1), hasEventID(?Var_Event1, "isVisited"), hasEventValue(?Var_Event1, true), hasStates(?Var_Visitor1, "Init_Visitor") -> hasStates(?Var_Visitor1, "Created_Visitor"), hasVisitorID(?Var_Visitor1, "Visitor_6v2ngi")
Rule 2:
Events(?Var_VisitorBeCus), StudentArtifact(?Var_Customer), VisitorArtifact(?Var_Visitor1), hasEventID(?Var_VisitorBeCus, "becomeCustomer"), hasEventValue(?Var_VisitorBeCus, true), hasStates(?Var_Customer, "Init_Customer"), hasStates(?Var_Visitor1, "Created_Visitor") -> hasCustomerID(?Var_Customer, "Customer_0pYs6b"), hasStates(?Var_Customer, "Created_Customer"), hasStates(?Var_Visitor1, "QuitedVisitor")
Rule 3
Events(?Var_Event_Quit), VisitorArtifact(?Var_Visitor_Quit), hasEventID(?Var_Event_Quit, "completeVisitor"), hasEventValue(?Var_Event_Quit, true), hasStates(?Var_Visitor_Quit, "Created_Visitor") -> hasStates(?Var_Visitor_Quit, "QuitedVisitor")

B.2 Student Artifact

Attributes: CustID, CustCredit, CustType

States: init, created, in_selecting_course, in_audition, in_order, in_training, in_final_exam, in_evaluation, in_certificate, completed
Lifecycle:

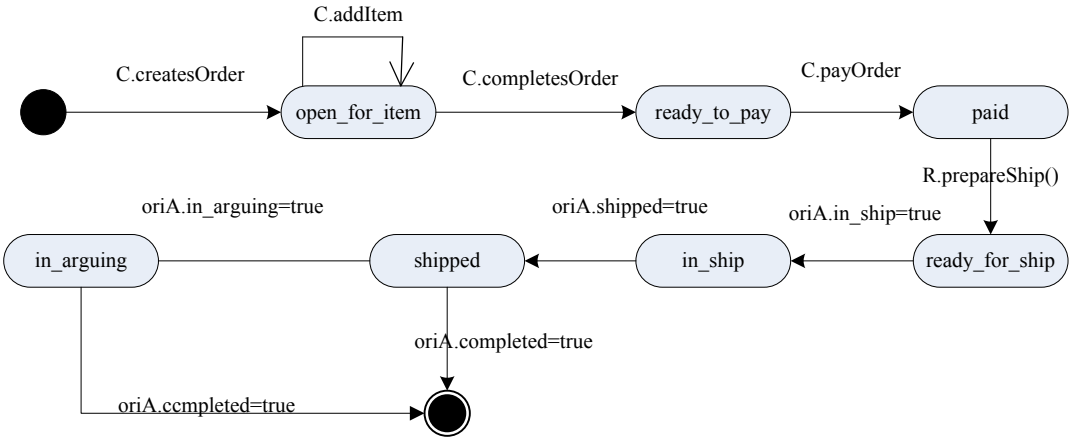


Business rules:
Rule 1:
Events(?Var_VisitorBeCus), StudentArtifact(?Var_Customer), VisitorArtifact(?Var_Visitor1), hasEventID(?Var_VisitorBeCus, "becomeCustomer"), hasEventValue(?Var_VisitorBeCus, true), hasStates(?Var_Customer, "Init_Customer"), hasStates(?Var_Visitor1, "Created_Visitor") -> hasCustomerID(?Var_Customer, "Customer_0pYs6b"), hasStates(?Var_Customer, "Created_Customer"), hasStates(?Var_Visitor1, "QuitedVisitor")
Rule 2:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, "created_Customer")
,CourseArtifact(?Var_Course), hasState(?Var_Customer, “created_Course”)
->
hasState(?Var_Customer, "in_selecting_course_Customer")
Rule 3:
StudentArtifact(?Var_Customer),hasState(?Var_Customer,
"in_selecting_course_Customer"),CourseArtifact(?Var_Course), hasState(?Var_Course, "in_audition_Course")
->
hasState(?Var_Customer, "in_audition_Customer")
Rule4:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, "in_audition_Customer") CourseArtifact(?Var_Course), hasState(?Var_Course, “in_course_Course”)
->
hasState(?Var_Customer, "in_training_Customer")
Rule5:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, "in_audition_Customer"), CourseArtifact(?Var_Course), hasState(?Var_Course, "in_course_Course")
->
hasState(?Var_Customer, "in_training_Customer")
Rule6:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, "in_audition_Customer"), OrderItem(?Var_OrderItem), hasState(?Var_OrderItem, “newly_added_OrderItem”)
->
hasState(?Var_Customer, "in_order_Customer")
Rule7:
StudentArtifact(?Var_Customer), hasState(?Var_Customer,
"in_order_Customer"),CourseArtifact(?Var_Course), hasState(?Var_Course, “in_course_Course”)
->
hasState(?Var_Customer, "in_training_Customer")
Rule 8:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, "in_training_Customer"), FinalExamArtifact(?Var_FinalExam), hasState(?Var_FinalExam, “in_exam_FinalExam”)

->
hasState(?Var_Customer, "in_final_exam_Customer")
Rule 9:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, "in_final_exam_Customer"), EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “in_evaluation_Evaluation)
->
hasState(?Var_Customer, “in_evaluation_Customer)
Rule 10:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, “in_evaluation_Customer), EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “completed_Evaluation), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “completed_Certificate”)
->
hasState(?Var_Customer, “completed_Customer”)
Rule 11:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, “in_evaluation_Customer), EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “completed_Evaluation), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “created_Certificate”)
->
hasState(?Var_Customer, “in_certificate_Customer”)
Rule 12
StudentArtifact(?Var_Customer), hasState(?Var_Customer, “in_certificate_Customer”), EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “completed_Evaluation”), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “completed_Certificate”)
->
hasState(?Var_Customer, “completed_Customer”)
Rule 13:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, "in_final_exam_Customer"), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “created_Certificate”)
->
hasState(?Var_Customer, “in_certificate_Customer)
Rule 14:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, “in_certificate_Customer), EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “completed_Evaluation), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “completed_Certificate”)
->
hasState(?Var_Customer, “completed_Customer”)
Rule 15:
StudentArtifact(?Var_Customer), hasState(?Var_Customer, “in_certificate_Customer), EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “in_evaluation_Evaluation), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “completed_Certificate”)
->
hasState(?Var_Customer, “in_evaluation_Customer”)

B.3 Order Artifact

Attributes: OrderID, CustID, Bid, ScheduleDate, Discount
States: open_for_item, ready_for_ship, paid, ready_for_ship, in_ship, shipped, in_arguing, completed
Lifecycle:

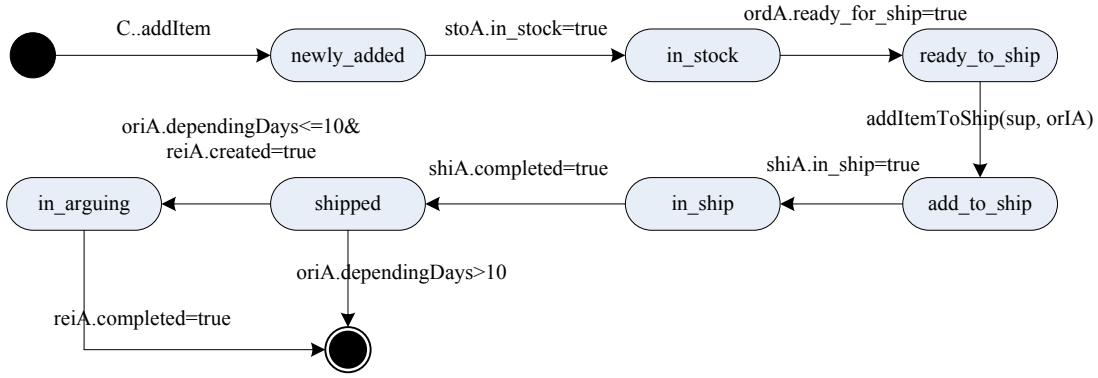


Business rules:
Rule 1: createsOder
Event(?Var_Event_BeginOrder), OrderArtifact(?Var_Order), hasEventID(?Var_Event_BeginOrder, “BeginOrder”), hasEventValue(?Var_Event_BeginOrder,true),
hasState(?Var_Order,“Init_Order”),StudentArtifact(?Var_Stu),hasCustomerID(?Var_Stu, “Cust_ID”),
->
hasState(?Var_Order, “Open_for_Item_Order”),hasOrderID(?Var_Order, “Order_ID”),hasCustomerID(?Var_Order, “Cust_ID”),
Rule 2: addItem
Event(?Var_Event_AddItem), hasEventID(?Var_Event_AddItem, “AddItem”), hasEventValue(?Var_Event_AddItem,true),
OrderArtifact(?Var_Order), OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “Init_OrderItem”),
hasState(?Var_Order, “Open_for_Item”), hasQuantity(?Var_OrderItem,
?int_quantity),hasPrice(?Var_OrderItem, ?float_price)
->
hasOrderID(?Var_OrderItem, “Order_ID”), hasItemAmount(?Var_OrderItem, ?float_price*?int_quantity),
hasState(?Var_OrderItem, “newly_added”)
Rule3: completeOrder
Event(?Var_Event_CompleteOrder), hasEventID(?Var_Event_CompleteOrder, “CompleteOrder”), hasEventValue(?Var_Event_CompleteOrder,true),
OrderArtifact(?Var_Order), hasState(?Var_Order, “open_for_item”)
->
hasState(?Var_Order, “ready_to_pay_Order”)
Rule 4: payOrder
Event(?Var_Event_PayOrder), hasEventID(?Var_Event_PayOrder, “PayOrder”), hasEventValue(?Var_Event_PayOrder,true),
OrderArtifact(?Var_Order), hasState(?Var_Order, “ready_to_pay_Order”),
->
hasState(?Var_Order, “Paid_Order”)
Rule 5:
OrderArtifact(?Var_Order), hasState(?Var_Order, “Paid_Order”),
ShipmentArtifact(?Var_Shipment), hasState(?Var_Shipment, “ready_to_dispatch_Shipment”)
->
hasState(?Var_Order, “ready_to_ship_Order”)
Rule 6:
OrderArtifact(?Var_Order), hasState(?Var_Order, “ready_to_ship_Order”),
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “in_ship_OrderItem”)
->
hasState(?Var_Order, “in_ship_Order”)
Rule 7:
OrderArtifact(?Var_Order), hasState(?Var_Order, “in_ship_Order”),
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “shipped_OrderItem”)
->
hasState(?Var_Order, “shipped_Order”)

Rule 8:
OrderArtifact(?Var_Order), hasState(?Var_Order, “shipped_Order”)
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “completed_OrderItem”)
->
hasState(?Var_Order, “completed_Order”)
Rule 9:
OrderArtifact(?Var_Order), hasState(?Var_Order, “shipped_Order”)
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “in_arguing_OrderItem”)
->
hasState(?Var_Order, “in_arguing_Order”)
Rule 10:
OrderArtifact(?Var_Order), hasState(?Var_Order, “in_arguing_Order”)
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “completed_OrderItem”)
->
hasState(?Var_Order, “completed_Order”)
Rule 11: calculate Discount
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, “ready_to_pay”), hasCustCredit(?Var_StudentArt, ?Var_CustCredit), greaterThanOrEqual(?Var_CustCredit, 2500)
->
hasDiscount(?Var_Order, 0.10)
Rule 12: calculate Discount
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, “ready_to_pay”),hasCustCredit(?Var_StudentArt, ?Var_CustCredit), greaterThanOrEqual(?Var_CustCredit, 1500), lessThan(?Var_CustCredit,2500)
->
hasDiscount(?Var_Order, 0.06)
Rule 13 : calculate Discount
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, “ready_to_pay”),hasCustCredit(?Var_StudentArt, ?Var_CustCredit), greaterThanOrEqual(?Var_CustCredit, 500), lessThan(?Var_CustCredit,1500)
->
hasDiscount(?Var_Order, 0.02)
Rule 14: calculate Discount
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, “ready_to_pay”),hasCustCredit(?Var_StudentArt, ?Var_CustCredit), lessThan(?Var_CustCredit,500)
->
hasDiscount(?Var_Order, 0)
Rule 15 :
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, “ready_to_pay”),hasCustCredit(?Var_StudentArt, ?Var_CustCredit), hasBid(?Var_Order, ?Var_Bid),
greaterThanOrEqual(?Var_Bid, 2000)
->
add(?Var_CustCredit, ?Var_CustCredit, 100), hasCustCredit(?Var_StudentArt, ?Var_CustCredit)
Rule 16 : calculate CustCredit
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, “ready_to_pay”),hasCustCredit(?Var_StudentArt, ?Var_CustCredit), hasBid(?Var_Order, ?Var_Bid),
greaterThanOrEqual(?Var_Bid, 1000), lessThan(?Var_CustCredit,2000)
->
add(?Var_CustCredit, ?Var_CustCredit, 60), hasCustCredit(?Var_StudentArt, ?Var_CustCredit)
Rule 17 : calculate CustCredit
StudentArtifact(?Var_StudentArt), OrderArtifact(?Var_Order),
hasState(?Var_Order, “ready_to_pay”),hasCustCredit(?Var_StudentArt, ?Var_CustCredit), hasBid(?Var_Order, ?Var_Bid),
lessThan(?Var_CustCredit,1000)
->
add(?Var_CustCredit, ?Var_CustCredit, 30), hasCustCredit(?Var_StudentArt, ?Var_CustCredit)

B.4 OrderItem Artifact

Attributes: OrderId, ProdName, ProdType, shipID, quantity, price}, States:newly_added, on_hold, ready_to_ship, added_to_shipping, in_shipping, shipped, pending, returned, closed
Lifecycle:

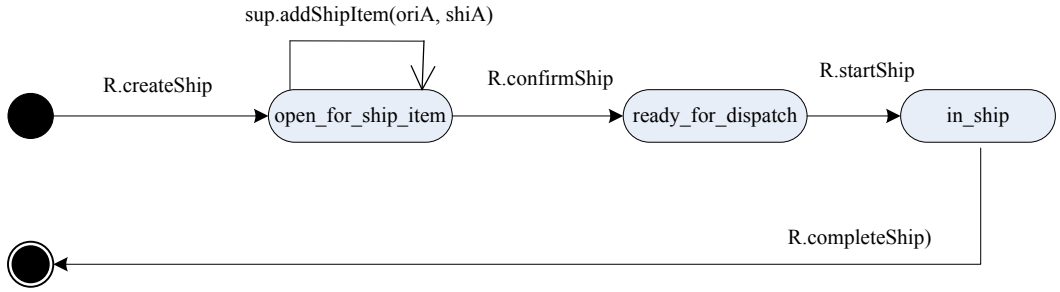


Business Rules:
Rule 1:
Event(?Var_Event_AddItem), hasEventID(?Var_Event_AddItem, “AddItem”), hasEvent-Value(?Var_Event_AddItem,true),
OrderArtifact(?Var_Order), OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “Init_OrderItem”), hasState(?Var_Order, “Open_for_Item”), hasQuantity(?Var_OrderItem, ?int_quantity),hasPrice(?Var_OrderItem, ?float_price)
->
hasOrderID(?Var_OrderItem, “Order_ID”), hasItemAmount(?Var_OrderItem, ?float_price*?int_quantity), hasState(?Var_OrderItem, “newly_added”)
Rule 2:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “newly_added_OrderItem”), StockArti-fact(?Var_StockArtifact), hasState(?Var_StockArtifact, “in_stock_Stock”)
->
hasState(?Var_OrderItem, “in_stock_OrderItem”)
Rule 3:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “in_stock_OrderItem”), OrderArti-fact(?Var_Order),
hasState(?Var_Order, “ready_to_ship_Order”)
->
hasState(?Var_OrderItem, “ready_to_ship_OrderItem”)
Rule 4:
Event(?Var_Event_AddShipItem), hasEventID(?Var_Event_AddShipItem, “AddItem”), hasEvent-Value(?Var_Event_AddShipItem,true),
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “ready_to_ship_OrderItem”),
->
hasState(?Var_OrderItem, “add_to_ship_OrderItem”)
Rule 5:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “add_to_ship_OrderItem”)
ShipmentArtifact(?Var_Shipment), hasState(?Var_Shipment, “in_ship_Shipment”)
->
hasState(?Var_OrderItem, “in_ship_OrderItem)
Rule 6:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “in_ship_OrderItem),
ShipmentArtifact(?Var_Shipment), hasState(?Var_Shipment, “completed_Shipment”)
->
hasState(?Var_OrderItem, “shipped)
Rule 7:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “shipped_OrderItem),

ReturnedItem(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “Init_ReturnedItem”), hasPending-Days(?Var_OrderItem, ?pendingDays), greaterThan(?pendingDays, 10)
->
hasState(?Var_OrderItem, “completed_OrderItem”)
Rule 8:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “shipped_OrderItem), ReturnedItem(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “Created_ReturnedItem”), hasPending-Days(?Var_OrderItem, ?pendingDays), lessThanOrEqual(?pendingDays, 10)
->
hasState(?Var_OrderItem, “in_arguing_OrderItem”)
Rule 9:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “in_arguing_OrderItem), ReturnedItem(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “completed_ReturnedItem”) ->
hasState(?Var_OrderItem, “completed_OrderItem”)

B.5 Shipment Artifact

Attributes: shipID, shipAddress, shipDate, CustomerID, ordered, receiveDate, States:created, ready_to_dispatch, in_shipping, completed,
Lifecycle:

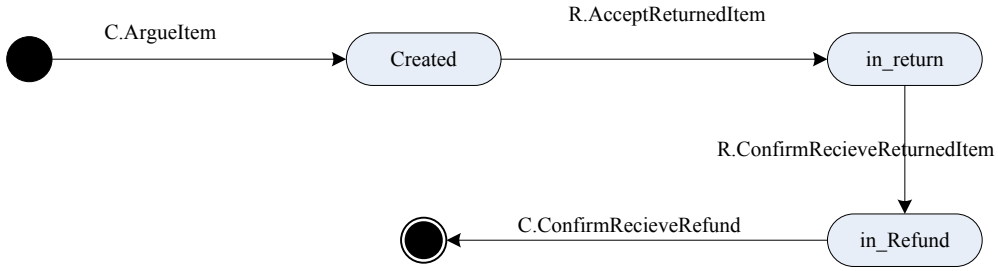


Business rules:
Rule 1:
ShipmentArtifact(?Var_Shipment), hasState(?Var_Shipment, “Init_Shipment”), OrderArti-fact(?Var_Order),hasState(?Var_Order, “Paid_Order”), hasOrderID(?Var_Order,?Var_OrderID1), StudentArtifact(?Var_Student), hasAddress(?Var_Student, ?Var_address1), hasCustID(?Var_Student, ?Var_CustID1)
->
hasState(?Var_Shipment, “Created_Shipment”), hasShipmentID(?Var_Shipment, “String_shipmentID”), hasOrderID(?Var_Shipment, ?Var_OrderID2), Equal(?Var_OrderID1, ?Var_OrderID2), hasShipAd-dress(?Var_Shipment, ?Var_address2), Equal(?Var_address1, ?Var_address2), hasCustID(?Var_Shipment, ?Var_CustID2), Equal(?Var_CustID1, ?Var_CustID2)
Rule 2:
Event(?Var_Event_ConfirmShipment), hasEventID(?Var_Event_ ConfirmShipment, “AddItem”), hasEvent-Value(?Var_Event_ ConfirmShipment,true),
ShipmentArtifact(?Var_Shipment), hasState(?Var_Shipment, “Created_Shipment”),
->
hasState(?Var_Shipment, “ready_to_dispatch_Shipment”)
Rule 3:
Event(?Var_Event_StartShip), hasEventID(?Var_Event_StartShip, “AddItem”), hasEventValue(?Var_Event_StartShip,true),
ShipmentArtifact(?Var_Shipment), hasState(?Var_Shipment, “ready_to_dispatch_Shipment”),
->
hasState(?Var_Shipment, “in_ship_Shipment”), hasShipDate(?Var_Shipment, “Data_shipdate”)
Rule 4:
Event(?Var_Event_SignforReceiving), hasEventID(?Var_Event_ SignforReceiving, “AddItem”), hasEvent-Value(?Var_Event_ SignforReceiving,true),
ShipmentArtifact(?Var_Shipment), hasState(?Var_Shipment, “in_ship_Shipment”)
->

hasState(?Var_Shipment, “completed_Shipment”), hasReceiveDate(?Var_Shipment, “Date_ReceiveDate”)

B.6 ReturnItem Artifact

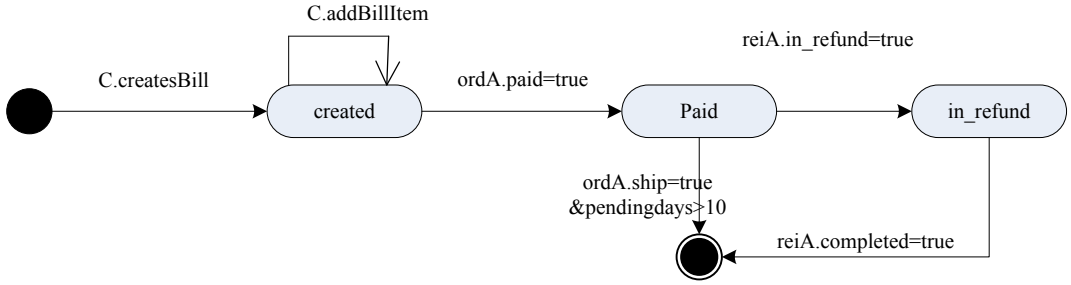
Attributes: OrderID, ProdName, ReturnedQuantity, Amount
State: Init, ceated, in_return, in_refund, completed
Lifecycle:



Business rules:
Rule 1:
Event(?Var_Event_ArgueProduct), hasEventID(?Var_Event_ ArgueProduct, “AddItem”), hasEvent-Value(?Var_Event_ ArgueProduct,true),
ReturnedItemArtifact(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “Init_ReturnedItem”), Orde-rItem(?Var_OrderItem), hasState(?Var_OrderItem, ?var_OrderID”), hasProd-Name(?Var_OrderItem,?var_ProdName)
hasPendingDays(?Var_OrderItem,?var_pendingDays), lessThan(?var_pendingDays, 10)
->
hasState(?Var_ReturnedItem, “Created_ReturnedItem”)
Rule 2:
Event(?Var_Event_AcceptReturnedItem), hasEventID(?Var_Event_ AcceptReturnedItem, “AcceptReturned-Item”), hasEventValue(?Var_Event_ AcceptReturnedItem,true), ReturnedItemArtifact(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “Created_ReturnedItem”),hasReturnedQuantity(?Var_ReturnedItem, ?Var_qty),hasPrice(?Var_ReturnedItem, ?var_price)
->
hasState(?Var_ReturnedItem, “in_return_ReturnedItem”), hasReturnAmount(?Var_ReturnedItem, ?var_ReturnedAmount), multiply(?Var_ReturnedAmount, ?Var_qty, ?var_price)
Rule 3:
Event(?Var_Event_ConfirmReceiveReturnedItem), hasEventID(?Var_Event_ ConfirmReceiveReturnedItem, “ConfirmReceiveReturnedItem”), hasEventValue(?Var_Event_ConfirmReceiveReturnedItem,true), Returned-ItemArtifact(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “in_return_ReturnedItem”)
->
hasState(?Var_ReturnedItem, “in_refund_ReturnedItem”)
Rule 4:
Event(?Var_Event_ConfirmReceiveRefund), hasEventID(?Var_Event_ ConfirmReceiveRefund, “ConfirmRe-ceiveRefund”), hasEventValue(?Var_Event_ConfirmReceiveRefund,true),
ReturnedItemArtifact(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “in_refund_ReturnedItem”)
->
hasState(?Var_ReturnedItem, “completed_ReturnedItem”)

B.7 Bill Artifact

Attributes:CustomerID, OrderID, BillID,Amount, refund
States:created, paid, in_refund, completed},
Lifecycle:



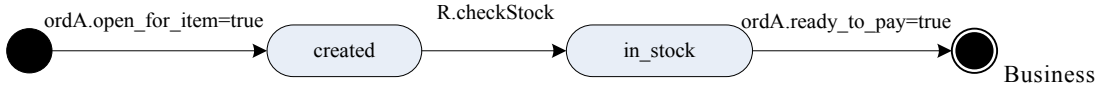
Business

Rules:

Rule 1: createBill
BillArtifact(?Var_Bill), hasState(?Var_Bill, “Init_Bill”), OrderArtifact(?Var_Order), hasState(?Var_Order, “Open_for_item”),hasOrderID(?Var_Order, “Order_ID”)
->
hasState(?Var_Bill, “Created_Bill”),hasBillID(?Var_Bill, “Bill_ID”), hasOrderID(?Var_Bill, “Order_ID”), hasAmount(?Var_Bill, 0),
Rule 2:
BillArtifact(?Var_Bill), hasState(?Var_Bill, “Created_Bill”),
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “newly_added_OrderItem”), hasAmount(?Var_OrderItem,?OrderItem_amount)
->
hasAmount(?Var_Bill, ?Bill_Amount), add(?Bill_Amount, Bill_Amount,?OrderItem_amount)
Rule 3: BillArtifact(?Var_Bill), hasState(?Var_Bill, “Created_Bill”),
OrderArtifact(?Var_Order), hasState(?Var_Order, “Paid_Order”)
->
hasState(?Var_Order, “paid_Bill”)
Rule 4:
BillArtifact(?Var_Bill), hasState(?Var_Bill, “paid_Bill”),
OrderArtifact(?Var_Order), hasState(?Var_Order, “shipped”),
OrderItemArtifact(?Var_OrderItem), hasPendingDays(?Var_OrderItem, ?pendingdays), greater-Than(?pendingdays, 10)
->
hasState(?Var_Bill, “completed_Bill”)
Rule 5:
BillArtifact(?Var_Bill), hasState(?Var_Bill, “paid_Bill”),
ReturnedItemArtifact(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “in_refund_ReturedItem”), hasAmount(?Var_ReturnedItem, ?amount”), hasRefund(?Var_Bill, ?refund)
->
hasState(?Var_Bill, “in_refund_Bill”), Equal(?amount, ?refund)
Rule 6:
BillArtifact(?Var_Bill), hasState(?Var_Bill, “in_refund_Bill”),
ReturnedItemArtifact(?Var_ReturnedItem), hasState(?Var_ReturnedItem, “completed_ReturnedItem”)
->
hasState(?Var_Bill, “completed_Bill”)

B.8 Stock Artifact

Attributes: ProdName,ProdType,Quantity
States: init, created, in_stock, completed
Lifecycle:



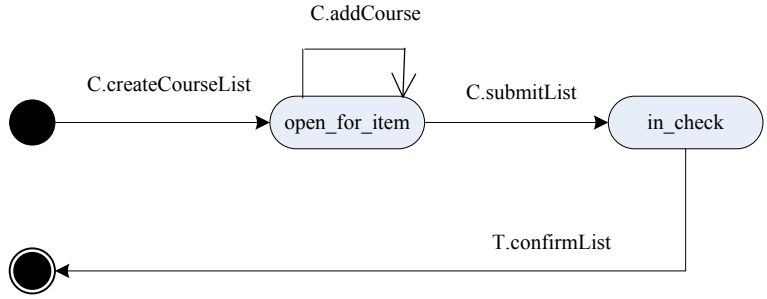
Business

rules :
Rule 1:
StockArtifact(?Var_Stock), hasState(?Var_Stock, “init_Stock”),
OrderArtifact(?Var_Order), hasState(?Var_Order, “open_for_item_Order”)

->
hasState(?Var_stock, “created_Stock”)
Rule 2:
OrderItemArtifact(?Var_OrderItem), hasState(?Var_OrderItem, “newly_added_OrderItem”), StockArtifact(?Var_StockArtifact), hasState(?Var_StockArtifact, “created_Stock”), hasProdName(?Var_OrderItem, ?Var_ProdName1), hasProdName(?Var_StockArtifact, ?Var_ProdName2), Equal(?Var_ProdName1, ?Var_ProdName2),hasQuantity(?Var_OrderItem, ?item_qty), hasStockQuantity(?Var_StockArtifact, ?stock_qty),
lessThanOrEqual(?item_qty,?stock_qty)
->
hasState(?Var_Stock, “in_stock_Stock”)
Rule 3:
StockArtifact(?Var_Stock), hasState(?Var_Stock, “in_stock_OrderItem”)
OrderArtifact(?Var_Order), hasState(?Var_order, “ready_to_pay_Order”)
->
hasState(?Var_Stock, “completed_Stock”)

B.9 CourseList Artifact

Attributes: CouseListID, StudentID, CourseNum
States: init, open_for_item, in_check, completed
Lifecycle:

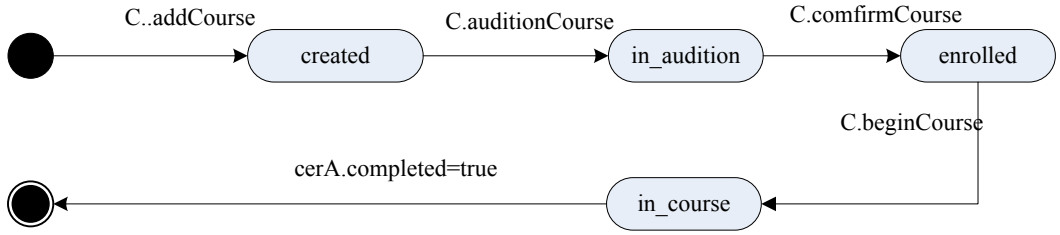


Business rules:
Rule 1:
Event(?Var_Event_CreateCourseList), hasEventID(?Var_Event_ CreateCourseList, “CreateCourseList”), hasEventValue(?Var_Event_ CreateCourseList,true), CourseListArtifact(?Var_CourseList), hasState(?Var_CourseList, “init_CourseList”)
StudentArtifact(?Var_Student), hasCustID(?Var_Student, ?CustID)
->
hasState(?Var_CourseList, “open_for_item_CourseList”), hasCoureListID(?Var_CourseList, ?CoureListID), hasStudentID(?Var_CourseList, ?CustID), hasCourseNum(?Var_CourseList, 0)
Rule 2:
Event(?Var_Event_AddCourse), hasEventID(?Var_Event_AddCourse, “AddCourse”), hasEventValue(?Var_Event_ AddCourse,true),
CourseListArtifact(?Var_CourseList), hasState(?Var_CourseList, “open_for_item_CourseList”),hasCourseNum(?Var_CourseList, ?courseNum)
->
Add(?courseNum, ?courseNum, 1)
Rule 3:
Event(?Var_Event_SubmitCourseList), hasEventID(?Var_Event_ SubmitCourseList, “SubmitCourseList”), hasEventValue(?Var_Event_ SubmitCourseList,true),
CourseListArtifact(?Var_CourseList), hasState(?Var_CourseList, “open_for_item_CourseList”)
->
hasState((?Var_CourseList, “in_check_CourseList”)
Rule 4:
Event(?Var_Event_ConfirmCourseList), hasEventID(?Var_Event_ ConfirmCourseList, “ConfirmCourseList”), hasEventValue(?Var_Event_ ConfirmCourseList,true),
CourseListArtifact(?Var_CourseList), hasState(?Var_CourseList, “in_check_CourseList”)
->

hasState((?Var_CourseList, “completed_CourseList”),

B.10 Course Artifact

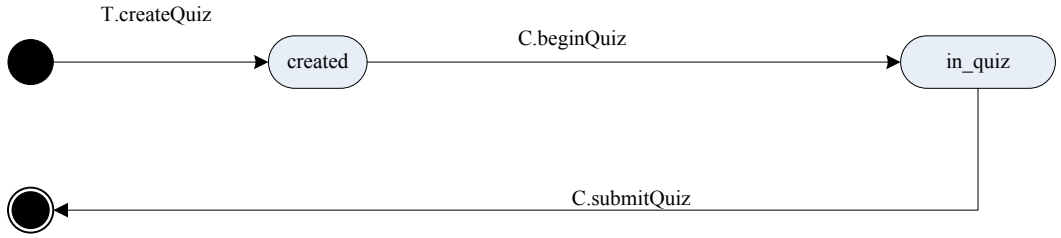
Attributes: CoureListID, courseID, teacherID, hours, startDate, finishDate
States: init, created, in_audition, enrolled,in_course, completed
Lifecycle:



Business rules:
Rule 1:
Event(?Var_Event_AddCourse), hasEventID(?Var_Event_AddCourse, “AddCourse”), hasEvent-Value(?Var_Event_ AddCourse,true),
CourseListArtifact(?Var_CourseList), hasState(?Var_Course, “open_for_item_CourseList”),
CourseArtifact(?Var_Course), hasState(?Var_Course, “Init_Course”), hasCourseListID(?Var_CourseList, ?CourseListID),
->
hasState(?Var_Course, “Created_Course”), hasCourseListID(?Var_Course, ?CourseListID), has-CourseID(?Var_Course, “CourseID”), hasTeacherID(?Var_Course, “TeacherID”)
Rule 2:
Event(?Var_Event_AuditionCourse), hasEventID(?Var_Event_AuditionCourse, “AuditionCourse”), hasEvent-Value(?Var_Event_ AuditionCourse,true),
CourseArtifact(?Var_Course), hasState(?Var_Course, “Created_Course”)
->
hasState(?Var_Course, “in_enrolled _Course”)
Rule 3:
Event(?Var_Event_ConfirmCourse), hasEventID(?Var_Event_ConfirmCourse, “ConfirmCourse”), hasEvent-Value(?Var_Event_ ConfirmCourse,true),
CourseArtifact(?Var_Course), hasState(?Var_Course, “in_audition_Course”)
->
hasState(?Var_Course, “enrolled_Course”)
Rule 4:
Event(?Var_Event_BeginCourse), hasEventID(?Var_Event_BeginCourse, “BeginCourse”), hasEvent-Value(?Var_Event_ BeginCourse,true),
CourseArtifact(?Var_Course), hasState(?Var_Course, “enrolled_Course”)
->
hasState(?Var_Course, “in_course_Course”)
Rule 5:
CourseArtifact(?Var_Course), hasState(?Var_Course, “in_audition_Course”)
CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “completed_Certificate”)
->
hasState(?Var_Course, “completed_Course”)

B.11 Quiz Artifact

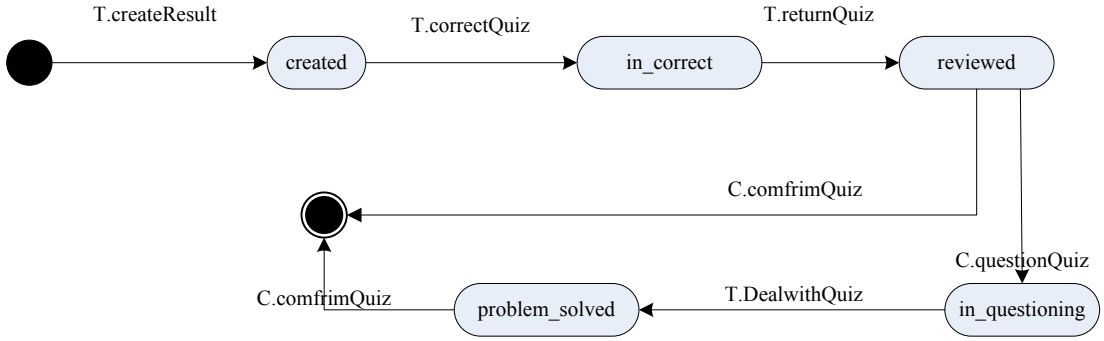
Attributes: QuizID, studentID, date, spendTime
States: init,created, in_quiz, completed
Lifecycle:



Business rules:
Rule 1:
Event(?Var_Event_CreateQuiz), hasEventID(?Var_Event_CreateQuiz, “CreateQuiz”), hasEvent-Value(?Var_Event_ CreateQuiz,true),
CourseArtifact(?Var_Course), hasState(?Var_Course, “in_course_Course”)
QuizArtifact(?Var_Quiz), hasState(?Var_Quiz, “init_Quiz”), hasCourseID(?Var_Course, ?Course_ID), hasTeacher(?Var_Course, ?Teacher_ID)
->
hasState(?Var_Quiz, “created_Quiz”), hasCourseID(?Var_Quize, ?Course_ID), hasTeacherID(?Var_Quiz, ?Teacher,ID)
Rule 2:
Event(?Var_Event_BeginQuiz), hasEventID(?Var_Event_BeginQuiz, “BeginQuiz”), hasEvent-Value(?Var_Event_ BeginQuiz,true),
QuizArtifact(?Var_Quiz), hasState(?Var_Quiz, “ created_Quiz ”),
->
hasState(?Var_Quiz, “in_quiz_Quiz”)
Rule 3:
Event(?Var_Event_SubmitQuiz), hasEventID(?Var_Event_SubmitQuiz, “SubmitQuiz”), hasEvent-Value(?Var_Event_ SubmitQuiz,true),
QuizArtifact(?Var_Quiz), hasState(?Var_Quiz, “ in_quiz_Quiz ”),
->
hasState(?Var_Quiz, “completed_Quiz”)

B.12 Exam Artifact

Attributes: ExamID, StudentID, CourseID, TeacherID, examScore
States: init, created, in_correct, reviewed, in_questioning, problem_solved, completed
Lifecycle:

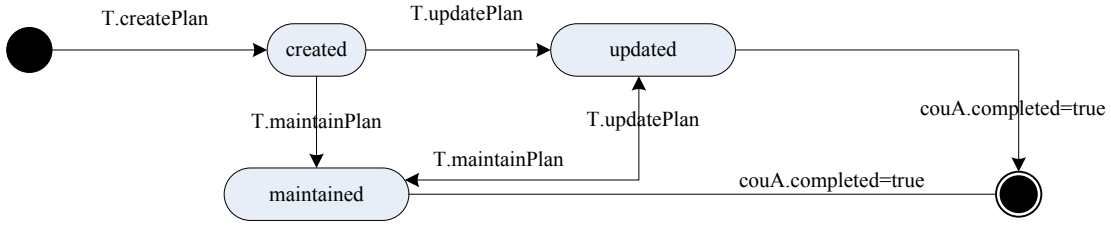


Business rules:
Rule 1:
QuizArtifact(?Var_Quiz), hasState(?Var_Quiz, “in_quiz_Quiz ”),
QuizResult(?Var_QuizResult), hasState(?Var_QuizResult, “init_QuizResult”)
->
hasState(?Var_QuizResult, “created_QuizResult”)
Rule2:
QuizArtifact(?Var_Quiz), hasState(?Var_Quiz, “completed_Quiz ”),
QuizResult(?Var_QuizResult), hasState(?Var_QuizResult, “init_QuizResult”)
->

hasState(?Var_QuizResult, “in_correct_QuizResult”)
Rule 3:
Event(?Var_Event_ReturnQuizResult), hasEventID(?Var_Event_ ReturnQuizResult, “ReturnQuizResult”), hasEventValue(?Var_Event_ ReturnQuizResult,true), QuizResult(?Var_QuizResult), hasState(?Var_QuizResult, “in_correct_QuizResult”)
->
hasState(?Var_QuizResult, “reviewed_QuizResult”)
Rule 4:
Event(?Var_Event_ConfirmQuizResult), hasEventID(?Var_Event_ ConfirmQuizResult, “ConfirmQuizResult”), hasEventValue(?Var_Event_ ConfirmQuizResult,true), QuizResult(?Var_QuizResult), hasState(?Var_QuizResult, “reviewed_QuizResult”)
->
hasState(?Var_QuizResult, “completed_QuizResult”)
Rule 5:
Event(?Var_Event_QuestionQuizResult), hasEventID(?Var_Event_ QuestionmQuizResult, “QuestionQuizResult”), hasEventValue(?Var_Event_ QuestionQuizResult,true), QuizResult(?Var_QuizResult), hasState(?Var_QuizResult, “reviewed_QuizResult”)
->
hasState(?Var_QuizResult, “in_questioning_QuizResult”)
Rule 6:
Event(?Var_Event_ResponseQuizResult), hasEventID(?Var_Event_ ResponseQuizResult, “ResponseResult”), hasEventValue(?Var_Event_ ResponseQuizResult,true), QuizResult(?Var_QuizResult), hasState(?Var_QuizResult, “in_questioning_QuizResult”)
->
hasState(?Var_QuizResult, “problem_solved_QuizResult”)
Rule 7:
Event(?Var_Event_ConfirmQuizResult), hasEventID(?Var_Event_ ConfirmQuizResult, “ConfirmResult”), hasEventValue(?Var_Event_ ConfirmQuizResult,true), QuizResult(?Var_QuizResult), hasState(?Var_QuizResult, “problem_solved_QuizResult”)
->
has(?Var_QuizResult, “completed_QuizResult”)

B.13 TeachingPlan Artifact

Attributes: TeachingPlanID, CourseID, customerID, recentQuizID, RecentQuizResult, maxQuizResultPerMonth
States: created, updated, maintained, completed
Lifecycle:



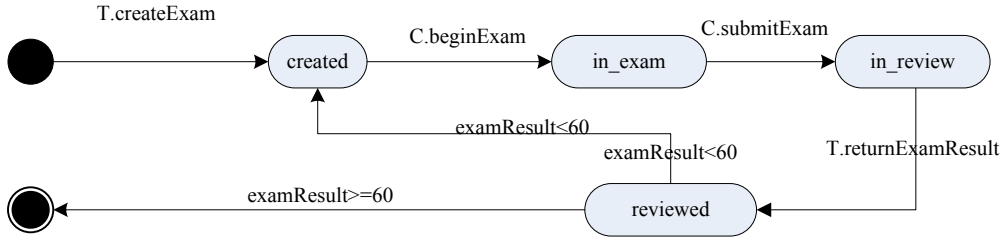
Business rules:
Rule 1:
CourseArtifact(?Var_Course), hasState(?Var_Course, “enrolled_Course”), TeachingPlanArtifact(?Var_TeachingPlan), hasState(?Var_TeachingPlan, “init_TeachingPlan), hasCourseID(?Var_Course, ?CourseID)
->
hasState(?Var_TeachingPlan, “created_TeachingPlan), hasTeachingPlanID(?Var_TeachingPlan, “TeachingPlan_ID”), hasCourseID(?Var_TeachingPlan, ?CourseID)
Rule2:
TeachingPlanArtifact(?Var_TeachingPlan), hasState(?Var_TeachingPlan, “created_TeachingPlan”), hasRecentQuizResult(?Var_TeachingPlan, ?RecentQuizResult), hasLastQuizResultButOne(?Var_TeachingPlan,

?LastButOne), greaterThan(?RecentQuizResult, ?LastButOne), hasMaxQuizResultPer-Month(?Var_TeachingPlan, ?MaxQuizResultPerMonth), greaterThan(?RecentQuizResult, ?MaxQuizResult-PerMonth)
->
hasState(?Var_TeachingPlan, “maintained_TeachingPlan),
Rule 3:
TeachingPlanArtifact(?Var_TeachingPlan), hasState(?Var_TeachingPlan, “created_TeachingPlan”), hasRe-centQuizResult(?Var_TeachingPlan, ?RecentQuizResult), hasLastQuizResultButOne(?Var_TeachingPlan, ?LastButOne), LessThanOrEqual(?RecentQuizResult, ?LastButOne), hasMaxQuizResultPer-Month(?Var_TeachingPlan, ?MaxQuizResultPerMonth), LessThanOrEqual(?RecentQuizResult, ?MaxQuizRe-sultPerMonth)
->
hasState(?Var_TeachingPlan, “updated_TeachingPlan)
Rule 4:
TeachingPlanArtifact(?Var_TeachingPlan), hasState(?Var_TeachingPlan, “updated_TeachingPlan), hasRe-centQuizResult(?Var_TeachingPlan, ?RecentQuizResult), hasLastQuizResultButOne(?Var_TeachingPlan, ?LastButOne), greaterThan(?RecentQuizResult, ?LastButOne), hasMaxQuizResultPer-Month(?Var_TeachingPlan, ?MaxQuizResultPerMonth), greaterThan(?RecentQuizResult, ?MaxQuizResult-PerMonth)
->
hasState(?Var_TeachingPlan, “maintained_TeachingPlan)
Rule 5:
TeachingPlanArtifact(?Var_TeachingPlan), hasState(?Var_TeachingPlan, “maintained_TeachingPlan), hasRe-centQuizResult(?Var_TeachingPlan, ?RecentQuizResult), hasLastQuizResultButOne(?Var_TeachingPlan, ?LastButOne), lessThanOrEqual(?RecentQuizResult, ?LastButOne), hasMaxQuizResultPer-Month(?Var_TeachingPlan, ?MaxQuizResultPerMonth), lessThanOrEqual (?RecentQuizResult, ?MaxQuizRe-sultPerMonth)
->
hasState(?Var_TeachingPlan, “updated_TeachingPlan)

Rule 6:
TeachingPlanArtifact(?Var_TeachingPlan), hasState(?Var_TeachingPlan, “updated_TeachingPlan), CourseArtifact(?Var_Course), hasState(?Var_Course, “completed_Course”)
->
hasState(?Var_TeachingPlan, “completed_TeachingPlan”)
Rule 7:
TeachingPlanArtifact(?Var_TeachingPlan), hasState(?Var_TeachingPlan, “maintaind_TeachingPlan), CourseArtifact(?Var_Course), hasState(?Var_Course, “completed_Course”)
->
hasState(?Var_TeachingPlan, “completed_TeachingPlan”)

B.14 FinalExam Artifact

Attributes: StudentID, CourseID, ExamID, ExamResult, date
States: created, in_exam, in_review, reviewed, completed
Lifecyle:

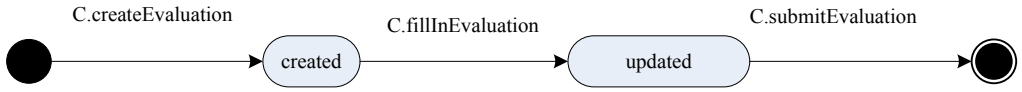


Business rules:
Rule 1:
CourseArtifact(?Var_Course), hasState(?Var_Course, “completed_Course”),

FinalExamArtifact(?Var_FinalExam), hasState(?Var_FinalExam, “init_FinalExam”)
->
hasState(?Var_FinalExam, “created_FinalExam”)
Rule 2:
Event(?Var_Event_BeginFinalExam), hasEventID(?Var_Event_ BeginFinalExam, “BeginFinalExam”), hasEventValue(?Var_Event_ BeginFinalExam,true), FinalExamArtifact(?Var_FinalExam),
hasState(?Var_FinalExam, “created_FinalExam”),
->
hasState(?Var_FinalExam, “in_exam_FinalExam”)
Rule 3:
Event(?Var_Event_SubmitFinalExam), hasEventID(?Var_Event_ SubmitFinalExam, “SubmitFinalExam”),
hasEventValue(?Var_Event_ SubmitFinalExam,true), FinalExamArtifact(?Var_FinalExam),
hasState(?Var_FinalExam, “in_exam_FinalExam”),
->
hasState(?Var_FinalExam, “in_review_FinalExam”)
Rule 4:
Event(?Var_Event_ReturnFinalExam), hasEventID(?Var_Event_ ReturnFinalExam, “ReturnFinalExam”),
hasEventValue(?Var_Event_ ReturnFinalExam,true), FinalExamArtifact(?Var_FinalExam),
hasState(?Var_FinalExam, “in_review_FinalExam”)
->
hasState(?Var_FinalExam, “reviewed_FinalExam”)
Rule 5
FinalExamArtifact(?Var_FinalExam), hasState(?Var_FinalExam, “reviewed_FinalExam”). hasExamResult(?Var_FinalExam, ?examResult),
greaterThanOrEqual(?examResult, 60)
->
hasState(?Var_FinalExam, “completed_FinalExam”)
Rule 6
FinalExamArtifact(?Var_FinalExam), hasState(?Var_FinalExam, “reviewed_FinalExam”). hasExamResult(?Var_FinalExam, ?examResult),
lessThan(?examResult, 60)
->
hasState(?Var_FinalExam, “created_FinalExam”)

B.15 Evaluation Artifact

Attributes: courseID, TeacherID, EvaluationID
States: init, created, updated, completed,
Lifecycle:



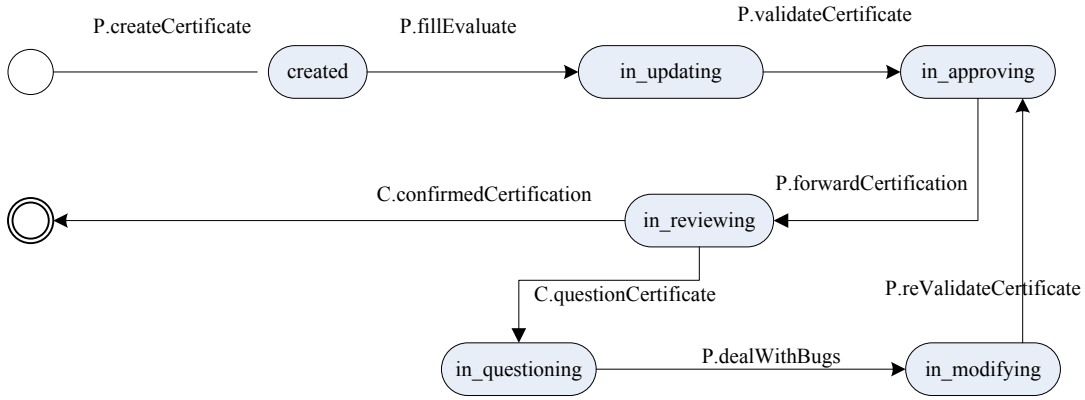
Business rule:
Rule 1:
CourseArtifact(?Var_Course), hasState(?Var_Course, “completed_Course”),
hasCourseID(?Var_Course, ?CourseID), hasTeacherID(?Var_Course, ?TeacherID)
EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “init_Evaluation”)
->
hasState(?Var_Evaluation, “created_Evaluation”), hasCourseID(?Var_Evaluation, ?CourseID), hasTeacherID(?Var_Evaluation, ?TeacherID)
Rule 2:
Event(?Var_Event_FillInEvaluation), hasEventID(?Var_Event_ FillInEvaluation, “FillInEvaluation”), hasEventValue(?Var_Event_ FillInEvaluation,true),
EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “created_Evaluation”)
->
hasState(?Var_Evaluation, “updated_Evaluation”)

Rule 3:

Event(?Var_Event_SubmitEvaluation), hasEventID(?Var_Event_ SubmitEvaluation, “SubmitEvaluation”), hasEventValue(?Var_Event_ SubmitEvaluation,true), EvaluationArtifact(?Var_Evaluation), hasState(?Var_Evaluation, “updated_Evaluation”)
->
hasState(?Var_Evaluation, “completed_Evaluation”)

B.16 Certification Artifact

Attributes: CertificationID, StudentID, CourseID, Hours, ExamResult, date, States: init, created, in_updating, in_approving, in_reviewing, in_questioning, in_modifying, completed
Lifecycle:



Business rules:
Rule 1:
FinalExamArtifact(?Var_FinalExam), hasState(?Var_FinalExam, “completed_FinalExam”), CertificationArtifact(?Var_Certificate), hasState(?Var_Certificate, “init_Certificate”)
CourseArtifact(?Var_Course),hasCourseID(?Var_Course, ?CourseID)
StudentArtifact(?Var_Student), hasCustID(?Var_Student, ?CustID)
->
hasState(?Var_Certificate, “created_Certificate”), hasCourseID(?Var_Certificate, ?CourseID), has-CustID(?Var_Certificate, ?CustID)
Rule 2:
Event(?Var_Event_FillInCertificate), hasEventID(?Var_Event_ FillInCertificate, “FillInCertificate”), hasEv-entValue(?Var_Event_ FillInCertificate,true),
CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “created_Certificate”)
->
hasState(?Var_Certificate, “in_updating_Certificate”)
Rule 3:
Event(?Var_Event_ApproveCertificate), hasEventID(?Var_Event_ ApproveCertificate, “ApproveCertificate”), hasEventValue(?Var_Event_ ApproveCertificate,true), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “in_updating_Certificate”)
->
hasState(?Var_Certificate, “in_approving_Certificate”)
Rule 4:
Event(?Var_Event_SubmitCertificate), hasEventID(?Var_Event_ SubmitCertificate, “SubmitCertificate”), hasEventValue(?Var_Event_ SubmitCertificate,true), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “in_approving_Certificate”)
->
hasState(?Var_Certificate, “in_reviewing_Certificate”)
Rule5:
Event(?Var_Event_ConfirmCertificate), hasEventID(?Var_Event_ ConfirmCertificate, “ConfirmCertificate”), hasEventValue(?Var_Event_ ConfirmCertificate,true), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “in_reviewing_Certificate”)
->
hasState(?Var_Certificate, “completed_Certificate”)

Rule6:
Event(?Var_Event_QuestionCertificate), hasEventID(?Var_Event_ QuestionCertificate, “QuestionCertificate”), hasEventValue(?Var_Event_ QuestionCertificate,true), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “in_reviewing_Certificate”)
->
hasState(?Var_Certificate, “in_questioning_Certificate”)
Rule7:
Event(?Var_Event_ModifyCertificate), hasEventID(?Var_Event_ ModifyCertificate, “ModifyCertificate”), hasEventValue(?Var_Event_ ModifyCertificate,true), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “in_questioning_Certificate”)
->
hasState(?Var_Certificate, “in_modifying_Certificate”)
Rule8:
Event(?Var_Event_ReValidateCertificate), hasEventID(?Var_Event_ ReValidateCertificate, “ReValidateCertificate”), hasEventValue(?Var_Event_ ReValidateCertificate,true), CertificateArtifact(?Var_Certificate), hasState(?Var_Certificate, “in_modifying_Certificate”)
->
hasState(?Var_Certificate, “in_Approving_Certificate”)

Appendix C: Sample Source Code

C.1 Class SBVRtoOWL

```
public class SBVRtoOWL {
                SBVRtoOWL(){

        }
                /**
                 * @param args
                 *
                 *
                 */

        public static String modifiedString (String url, String input, String para, String value)
        {
                StringTokenizer st = new StringTokenizer(input);
                //String temp=null;
                int count=st.countTokens();
                //System.out.println(st.countTokens());
           //StringTokenizer st = new StringTokenizer("this is a test");
                String temp="";
                String str = "";
                String result="";
                String s;
            while (st.hasMoreTokens()) {
                 temp=st.nextToken();
                 //System.out.println(temp);

               temp+="+";
               str+=temp;
            }
            str = str.substring(0,str.length() - 1);

            String last=str.substring(str.length()-1,str.length());
                 if(last.equals(".")==true)
                                result=str;
                 else
                                result=str+".";

            result="?text="+ result +"&"+para +"=" +value;
            result=url+result;
            return result;

        }

                public static void requestHTTP(String strURL, String filepath) throws ClientProtocolEx-
ception, IOException
                {

                        HttpGet get = new HttpGet(strURL);
                        HttpClient httpclient = new DefaultHttpClient();
                  HttpResponse response = httpclient.execute(get);

                  InputStream ins = response.getEntity().getContent();

                  String str2=stream2String(ins);
```

```
                saveOWL(filepath,str2);


            }
            public static void save2File()
            {


            }
            public static String stream2String(InputStream ins)
            {
                        BufferedReader reader=new BufferedReader(new InputStream-
Reader(ins),16*1024);
                        StringBuilder sb = new StringBuilder();
                        String line=null;
                        try{
                                    while((line=reader.readLine())!=null){
                                                sb.append(line+"\n");
                                    }
                        }
                        catch(IOException e){
                                    e.printStackTrace();
                        }finally{
                                    try{
                                                ins.close();
                                    }catch(IOException e){
                                                e.printStackTrace();
                                    }
                        }
                        return sb.toString();
            }
            public static String streamToString(InputStream ins) throws UnsupportedEncodingExcep-
tion, IOException
            {
                          byte[] b = new byte[1024];
                          BufferedReader reader=new BufferedReader(new InputStream-
Reader(ins),16*1024);
                         StringBuilder sb = new StringBuilder();
                          while (ins.read(b) != -1) {
                          sb.append(new String(b, "UTF-8"));

                          }
                          return sb.toString();

            }
            public static void saveOWL(String despath, InputStream ins) throws IOException
            {
                        File file=new File(despath);
                        byte buffer[]=new byte[1024];
                        FileOutputStream fos=new FileOutputStream(file);
                        int ret=-1;
                        while((ret = ins.read(buffer, 0, 1024)) != -1) {
      fos.write(buffer, 0, 1024);
   }
                        fos.close();
            }
            public static void saveOWL(String despath,String content) throws IOException
            {
                        File file=new File(despath);
```

```
                        byte buffer[]=new byte[1024];
                        buffer=content.getBytes();
                        int b=content.length();
                        FileOutputStream fos=new FileOutputStream(file);
                        fos.write(buffer,0,b);
                        //fos.flush();
                        fos.close();
                }
}
```

C.2 The Function of 'developArtifactOntology()'

```
public void developArtifactOntology()
        {
                OWLClass class_Artifact = factory.getOWLClass(IRI.create(ontologyIRI + "#Artifact"));
                OWLClass class_Event = factory.getOWLClass(IRI.create(ontologyIRI + "#Events"));
                OWLClass class_VisitorArt = factory.getOWLClass(IRI.create(ontologyIRI + "#Visito-
rArtifact"));
                OWLClass class_StudentArt = factory.getOWLClass(IRI.create(ontologyIRI + "#Studen-
tArtifact"));

                ///////event
                OWLDataProperty hasEventID = factory.getOWLDataProperty(IRI.create(ontologyIRI +
"#hasEventID"));
                OWLDataProperty hasEventValue = fac-
tory.getOWLDataProperty(IRI.create(ontologyIRI + "#hasEventValue"));
                //Set<OWLAxiom> domainsAndRanges =new HashSet<OWLAxiom>();

                OWLDataProperty hasVisitorID = factory.getOWLDataProperty(IRI.create(ontologyIRI
+ "#hasVisitorID"));
                OWLDataProperty hasAccessTime = fac-
tory.getOWLDataProperty(IRI.create(ontologyIRI + "#hasAccessTime"));
                OWLDataProperty hasQuitTime = factory.getOWLDataProperty(IRI.create(ontologyIRI
+ "#hasQuitTime"));
                OWLDataProperty hasStates= factory.getOWLDataProperty(IRI.create(ontologyIRI +
"#hasStates"));
                OWLDataProperty hasCustomerID = fac-
tory.getOWLDataProperty(IRI.create(ontologyIRI + "#hasCustomerID"));
    /////Rule 1
    SWRLVariable var_Vis1 = factory.getSWRLVariable(IRI.create(ontologyIRI + "#Var_Visitor1"));
    SWRLVariable var_Event1 = factory.getSWRLVariable(IRI.create(ontologyIRI + "#Var_Event1"));
    Set<SWRLAtom> body_vis1 = new HashSet<SWRLAtom>();
    body_vis1.add(factory.getSWRLClassAtom(class_VisitorArt, var_Vis1));
    body_vis1.add(factory.getSWRLClassAtom(class_Event, var_Event1));
            String rand="Visitor_"+this.randomString(6);
    System.out.println("the random string is: "+rand);

    OWLLiteral lit_bool = factory.getOWLTypedLiteral(true);
    SWRLLiteralArgument litArg_bool = factory.getSWRLLiteralArgument(lit_bool);

    OWLLiteral lit_string_ID = factory.getOWLTypedLiteral(rand);
    SWRLLiteralArgument litArg_string_ID = factory.getSWRLLiteralArgument(lit_string_ID);

    OWLLiteral lit_string_InitVisitor = factory.getOWLTypedLiteral("Init_Visitor");
    SWRLLiteralArgument litArg_string_InitVisitor = fac-
tory.getSWRLLiteralArgument(lit_string_InitVisitor);


    OWLLiteral lit_string_InitCustomer = factory.getOWLTypedLiteral("Init_Customer");
```

```
        SWRLLiteralArgument litArg_string_InitCustomer = fac-
tory.getSWRLLiteralArgument(lit_string_InitCustomer);

        OWLLiteral lit_string_CreatedVisitor = factory.getOWLTypedLiteral("Created_Visitor");
        SWRLLiteralArgument litArg_string_CreatedVisitor = fac-
tory.getSWRLLiteralArgument(lit_string_CreatedVisitor);
       // body.add(df.getSWRLDataPropertyAtom(getOWLDataProperty("d"), varY, litArg1));
        ////

        OWLLiteral lit_string_CreatedCustomer = factory.getOWLTypedLiteral("Created_Customer");
        SWRLLiteralArgument litArg_string_CreatedCustomer = fac-
tory.getSWRLLiteralArgument(lit_string_CreatedCustomer);

        body_vis1.add(factory.getSWRLDataPropertyAtom(hasStates, var_Vis1, litArg_string_InitVisitor));

        //event
        OWLLiteral lit_string_EventID = factory.getOWLTypedLiteral("isVisited");
        SWRLLiteralArgument litArg_string_EventID = factory.getSWRLLiteralArgument(lit_string_EventID);

        OWLLiteral lit_bool_EventValue = factory.getOWLTypedLiteral(true);
        SWRLLiteralArgument litArg_bool_EventValue = fac-
tory.getSWRLLiteralArgument(lit_bool_EventValue);
        ////////

        body_vis1.add(factory.getSWRLDataPropertyAtom(hasEventValue, var_Event1, li-
tArg_bool_EventValue));
        body_vis1.add(factory.getSWRLDataPropertyAtom(hasEventID, var_Event1, litArg_string_EventID));

        Set<SWRLAtom> head_vis1 = new HashSet<SWRLAtom>();
        head_vis1.add(factory.getSWRLDataPropertyAtom(hasStates, var_Vis1, litArg_string_CreatedVisitor));
        head_vis1.add(factory.getSWRLDataPropertyAtom(hasVisitorID, var_Vis1, litArg_string_ID));
        SWRLRule rule_vis1 = factory.getSWRLRule(body_vis1,
              head_vis1);
        manager.applyChange(new AddAxiom(ontology, rule_vis1));

        ///////Rule 2: becomeCustomer is invoked
        // var_Vis1 = factory.getSWRLVariable(IRI.create(ontologyIRI + "#Var_Visitor1"));

                          //manager.addAxiom(ontology, beCus2);
                          SWRLVariable var_Vis_BeCus = factory.getSWRLVariable(IRI.create(ontologyIRI +
"#Var_VisitorBeCus"));
        Set<SWRLAtom> body_vis2 = new HashSet<SWRLAtom>();
        body_vis2.add(factory.getSWRLClassAtom(class_Event, var_Vis_BeCus));
        body_vis2.add(factory.getSWRLClassAtom(class_VisitorArt, var_Vis1));
        SWRLVariable var_Cus = factory.getSWRLVariable(IRI.create(ontologyIRI + "#Var_Customer"));
        body_vis2.add(factory.getSWRLClassAtom(class_StudentArt, var_Cus));

        OWLLiteral lit_string_EventID2 = factory.getOWLTypedLiteral("becomeCustomer");
        SWRLLiteralArgument litArg_string_EventID2 = fac-
tory.getSWRLLiteralArgument(lit_string_EventID2);

        OWLLiteral lit_string_QuitedVisitor = factory.getOWLTypedLiteral("QuitedVisitor");
        SWRLLiteralArgument litArg_string_QuitedVisitor = fac-
tory.getSWRLLiteralArgument(lit_string_QuitedVisitor);

        rand="Customer_"+this.randomString(6);
        OWLLiteral lit_string_CusID1 = factory.getOWLTypedLiteral(rand);
        SWRLLiteralArgument litArg_string_CusID1 = factory.getSWRLLiteralArgument(lit_string_CusID1);

            body_vis2.add(factory.getSWRLDataPropertyAtom(hasStates, var_Cus, litArg_string_InitCustomer));
```

```
	body_vis2.add(factory.getSWRLDataPropertyAtom(hasStates, var_Vis1, litArg_string_CreatedVisitor));
	body_vis2.add(factory.getSWRLDataPropertyAtom(hasEventValue, var_Vis_BeCus, li-
tArg_bool_EventValue));
	body_vis2.add(factory.getSWRLDataPropertyAtom(hasEventID, var_Vis_BeCus, li-
tArg_string_EventID2));

		Set<SWRLAtom> head_vis2 = new HashSet<SWRLAtom>();
	head_vis2.add(factory.getSWRLDataPropertyAtom(hasStates, var_Vis1, litArg_string_QuitedVisitor));
	head_vis2.add(factory.getSWRLDataPropertyAtom(hasCustomerID, var_Cus, litArg_string_CusID1));
	head_vis2.add(factory.getSWRLDataPropertyAtom(hasStates, var_Cus, litArg_string_CreatedCustomer));
	//head_vis2.add(factory.getSWRLSameIndividualAtom(var_Vis1, var_Cus));
	SWRLRule rule_vis2 = factory.getSWRLRule(body_vis2,
		head_vis2);
	manager.applyChange(new AddAxiom(ontology, rule_vis2));


	///////Rule 3: completeVisitor is invoked

		SWRLVariable var_Vis_Quit = factory.getSWRLVariable(IRI.create(ontologyIRI +
"#Var_Visitor_Quit"));
	SWRLVariable var_Event_Quit = factory.getSWRLVariable(IRI.create(ontologyIRI +
"#Var_Event_Quit"));

	Set<SWRLAtom> body_vis3 = new HashSet<SWRLAtom>();
	body_vis3.add(factory.getSWRLClassAtom(class_VisitorArt, var_Vis_Quit));
	body_vis3.add(factory.getSWRLClassAtom(class_Event, var_Event_Quit));
	body_vis3.add(factory.getSWRLDataPropertyAtom(hasStates, var_Vis_Quit, li-
tArg_string_CreatedVisitor));

	//event hasID completeVisitor, hasValue true
	OWLLiteral lit_string_EventID_Quit = factory.getOWLTypedLiteral("completeVisitor");
	SWRLLiteralArgument litArg_string_EventID_Quit = fac-
tory.getSWRLLiteralArgument(lit_string_EventID_Quit);

	body_vis3.add(factory.getSWRLDataPropertyAtom(hasEventValue, var_Event_Quit, li-
tArg_bool_EventValue));
	body_vis3.add(factory.getSWRLDataPropertyAtom(hasEventID, var_Event_Quit, li-
tArg_string_EventID_Quit));

	//////////////////////////////////////////////////////
	Set<SWRLAtom> head_vis3 = new HashSet<SWRLAtom>();
	head_vis3.add(factory.getSWRLDataPropertyAtom(hasStates, var_Vis_Quit, li-
tArg_string_QuitedVisitor));
	SWRLRule rule_vis3 = factory.getSWRLRule(body_vis3,
		head_vis3);
	manager.applyChange(new AddAxiom(ontology, rule_vis3));


   //  List<OWLOntologyChange>  changes = new ArrayList<OWLOntologyChange>();
    // changes.add(new RemoveAxiom(ont, oldAxiom));
    //manager.applyChanges(changes);

		}
```

C.3 Invoke the 'Becomecustomer' Event

```
final Button becomecustomerButton = new Button(eventInvokeAreaGroup, SWT.RADIO);
		becomecustomerButton.addSelectionListener(new SelectionAdapter() {
			public void widgetSelected(final SelectionEvent e) {
```

```
                                        ////////////////////////////
                                        try {
                                                message_text.setText("");
                                                String eventID="becomeCustomer";
                                                boolean  eventValue=true;
                                                String visitorID="";
                                                String temp="";
                                                OWL2APIProcessor owl2 = new OWL2APIProcessor();
                                                OWLXMLOntologyFormat format = new OWLXMLOn-
tologyFormat();
                                                String filename="c:/tmp/MergedOntology1.owl";
                                                File file3=new File(filename);

                                                String file_art="c:/tmp/artifactOntology1.owl";
                                                File file_artifact=new File(file_art);

                                                //owl2.developOntology();

                                                owl2.developArtifactOntology();
                                                owl2.saveOntologys(file_artifact,format);

                                                File filem=new File("c:/tmp/isVisitedEvent.owl");


                                                temp="Save ontology to file:"+ filename+"\n";


                                                owl2.ontology=owl2.mergeOntology(file_artifact,
filem);
                                                owl2.saveOntologys(file3,format);
                                                message_text.setText(temp);
                                                System.out.println();
                                                  System.out.println("Begin to reason before becomeCus-
tomer event...");
                                                  temp=temp+"begin to reason before becomeCustomer
event..."+"\n";
                                                PelletReasoner reasonerPellet1=owl2.reasoner();
                                                System.out.println("Finish reasoning before becomeCus-
tomer event!");
                                                temp=temp+"finish reasoning before becomeCustomer
event!"+"\n\n";
                                                System.out.println();

                                                 OWLDataProperty hasStates =
owl2.factory.getOWLDataProperty(IRI.create(owl2.ontologyIRI + "#hasStates"));
                                                OWLDataProperty hasCustomerID =
owl2.factory.getOWLDataProperty(IRI.create(owl2.ontologyIRI + "#hasCustomerID"));
                                                OWLClass class_VisitorArt =
owl2.factory.getOWLClass(IRI.create(owl2.ontologyIRI+"#VisitorArtifact"));
                                                OWLClass class_StudentArt =
owl2.factory.getOWLClass(IRI.create(owl2.ontologyIRI+"#StudentArtifact"));
                                                OWLDataProperty hasVisitorID =
owl2.factory.getOWLDataProperty(IRI.create(owl2.ontologyIRI + "#hasVisitorID"));

                                                ////////////////////////////////////////////////////////////////////////
                                                int ind_count=0;
                                                String temp2="";
                                                NodeSet<OWLNamedIndividual> individuals_before = reason-
erPellet1.getInstances(class_VisitorArt, false);
                                  for(Node<OWLNamedIndividual> sameInd : individuals_before)
```

```
                                        {
                                                ind_count++;
                                        }
                                        System.out.println("ind_count is: "+ind_count);
                                        String []origin_state=new String[ind_count];
                                        ind_count=0;

                                                        for(Node<OWLNamedIndividual> sameInd :
individuals_before) {
                                                                // sameInd contains information
about the individual (and all other individuals that were inferred to be the same)
                                                                temp2="";

                                                                OWLNamedIndividual ind =
sameInd.getRepresentativeElement();
                                                                System.out.println(sameInd);

        temp2=temp2+sameInd.toString()+"\n";

                                                        // get the info about this specific individual
                                                                Set<OWLLiteral> states = reason-
erPellet1.getDataPropertyValues(ind, hasStates );
                                                                //reasoner.

                                                                String statesValue;
                                                                if(states.isEmpty())
                                                                {
                                                                        Sys-
tem.out.println("states is unknown");
                                                                        temp2=temp2+"Unknow
states for this individual"+"\n\n";

                                                                }
                                                                else
                                                                {


                                                                statesValue =
states.iterator().next().getLiteral();
                                                                ori-
gin_state[ind_count]=statesValue;
                                                                System.out.println("this invidual
has original States: "+statesValue);
                                                                temp2=temp2+"this invidual origi-
nal hasStates:"+statesValue+"\n\n";


                                                                }
                                                                ind_count++;
                                                                temp=temp+temp2;


                                                        System.out.println();
                                                        System.out.println();
                                                }

        ////////////////////////////////////////////////////////////////////////////
                                                        temp2="";
                                                        ind_count=0;
                                                NodeSet<OWLNamedIndividual> individu-
als_before_cus = reasonerPellet1.getInstances(class_StudentArt, false);
```

```
                                                    for(Node<OWLNamedIndividual> sameInd : indi-
viduals_before_cus)
                                                    {
                                                            ind_count++;
                                                    }
                                                    System.out.println("ind_count is: "+ind_count);
                                                    String []origin_state_cus=new String[ind_count];
                                                    ind_count=0;

                                                                    for(Node<OWLNamedIndividual>
sameInd : individuals_before_cus) {
                                                                            // sameInd contains in-
formation about the individual (and all other individuals that were inferred to be the same)
                                                                            temp2="";

                                                                            OWLNamedIndividual
ind =  sameInd.getRepresentativeElement();
                                                                            Sys-
tem.out.println(sameInd);

            temp2=temp2+sameInd.toString()+"\n";

                                                                    // get the info about this specific
individual
                                                                            Set<OWLLiteral> states
= reasonerPellet1.getDataPropertyValues(ind, hasStates );
                                                                            //reasoner.

                                                                            String statesValue;
                                                                            if(states.isEmpty())
                                                                            {
                                                                                    Sys-
tem.out.println("states is unknown");

            temp2=temp2+"Unknow states for this individual"+"\n\n";

                                                                            }
                                                                            else
                                                                            {


                                                                            statesValue =
states.iterator().next().getLiteral();
                                                                            ori-
gin_state_cus[ind_count]=statesValue;
                                                                            System.out.println("this
invidual has current States: "+statesValue);
                                                                            temp2=temp2+"this in-
vidual has current States:"+statesValue+"\n\n";

                                                                            }
                                                                            ind_count++;
                                                                            temp=temp+temp2;


                                                                    System.out.println();
                                                                    System.out.println();
                                                            }
```

```
                                        ////////////////////////////////////////////////////////////////////////////
                                        ////////////////////////////////////////////////////////////////////////////

                                                OWLClass class_Event=
owl2.factory.getOWLClass(IRI.create(owl2.ontologyIRI+"#Event"));
                                                //////set value to invoke event
                                        OWLNamedIndividual in_becomeCustomer1 =
owl2.factory.getOWLNamedIndividual(IRI.create(owl2.ontologyIRI + "#in_BecomeCustomer1"));
                                        OWLClassAssertionAxiom beCus1 =
owl2.factory.getOWLClassAssertionAxiom(class_Event, in_becomeCustomer1);
                                        owl2.manager.addAxiom(owl2.ontology, beCus1);


                                        SWRLVariable var_Event1 =
owl2.factory.getSWRLVariable(IRI.create(owl2.ontologyIRI + "#Var_Event1"));
                                                OWLDataProperty hasEventID =
owl2.factory.getOWLDataProperty(IRI.create(owl2.ontologyIRI + "#hasEventID"));
                                                OWLDataProperty hasEventValue =
owl2.factory.getOWLDataProperty(IRI.create(owl2.ontologyIRI + "#hasEventValue"));


                                                OWLLiteral lit_string_EventID =
owl2.factory.getOWLTypedLiteral(eventID);
                                        SWRLLiteralArgument litArg_string_EventID =
owl2.factory.getSWRLLiteralArgument(lit_string_EventID);

                                        OWLLiteral lit_bool_EventValue =
owl2.factory.getOWLTypedLiteral(eventValue);
                                        SWRLLiteralArgument litArg_bool_EventValue =
owl2.factory.getSWRLLiteralArgument(lit_bool_EventValue);


                                        Set<SWRLAtom> body_vis1 = new HashSet<SWRLAtom>();


                                        body_vis1.add(owl2.factory.getSWRLDataPropertyAtom(hasEventValue,
var_Event1, litArg_bool_EventValue));
                                        body_vis1.add(owl2.factory.getSWRLDataPropertyAtom(hasEventID,
var_Event1, litArg_string_EventID));


                                        Set<OWLAxiom> axioms_vis = new HashSet<OWLAxiom>();
                                        axi-
oms_vis.add(owl2.factory.getOWLDataPropertyAssertionAxiom(hasEventID, in_becomeCustomer1, eventID));
                                        axi-
oms_vis.add(owl2.factory.getOWLDataPropertyAssertionAxiom(hasEventValue, in_becomeCustomer1, event-
Value));
                                        owl2.manager.addAxioms(owl2.ontology, axioms_vis);


                                                File tmp_m=new File("c:/tmp/becomeCustomer.owl");
                                                try {
                                                        owl2.manager.saveOntology(owl2.ontology,
format, IRI.create(tmp_m.toURI()));
                                                } catch (OWLOntologyStorageException e1) {
                                                        // TODO Auto-generated catch block
                                                        e1.printStackTrace();
                                                }
                                                 System.out.println("begin to reason...");
```

```
                                        temp=temp+"begin to reason after the event..."+"\n";
                                        //PelletReasoner reasonerPellet=owl2.reasoner();
                                        ////
                                        PelletReasoner reasonerPellet = PelletReasonerFac-
tory.getInstance().createReasoner(owl2.ontology );
                                                System.out.println("done.");

                                                reasonerPellet.getKB().realize();

                                        ////
                                                System.out.println("finish reasoning.");
                                                  temp=temp+"finish to reason after the
event..."+"\n\n";
                                                System.out.println();
                                        //////now query the reasoner
                                        ////////////////////////////////////////////////////////////////////////////
                                        ////////////////////////////////////////////////////////////////////////////

                                        String filename2="c:/tmp/becomeCustomerEvent.owl";
                                        File file2=new File(filename2);

                                        //save inferred ontology

                                        OWLOntology exportedOntology =
owl2.manager.createOntology(IRI.create(file2.toURI()) );
                                        InferredOntologyGenerator generator = new InferredOn-
tologyGenerator(reasonerPellet);
                                        generator.fillOntology( owl2.manager, exportedOntology
);

                                        // replace old value with new value
                                        List<OWLOntologyChange> changes2 = new Array-
List<OWLOntologyChange>();
                                        Set<OWLAxiom> axioms2 = exportedOntol-
ogy.getAxioms();
                                        for (OWLAxiom ax : axioms2) {

                                                if(ax.toString().contains("Created_Visitor") ||
ax.toString().contains("Init_Customer"))
                                                {
                                                        changes2.add(new RemoveAx-
iom(exportedOntology, ax));
                                                }
                                        }

                                owl2.manager.applyChanges(changes2);
                                        ////////////////////

                                        try {

                    owl2.manager.saveOntology(exportedOntology, new OWLXMLOntologyFor-
mat(),IRI.create(file2.toURI()));
                                                System.out.println("the file is save
to"+filename2);
                                        } catch (OWLOntologyStorageException e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }

                                        ///begin to reason with the event becomecustomer
```

```
                                                    NodeSet<OWLNamedIndividual> individu-
als_vis = reasonerPellet.getInstances(class_VisitorArt, false);
                                                temp2="";
                                                ind_count=0;
                                                for(Node<OWLNamedIndividual> sameInd : individu-
als_vis) {
                                                    // sameInd contains information about the in-
dividual (and all other individuals that were inferred to be the same)
                                                    temp2="";

                                                    OWLNamedIndividual ind =
sameInd.getRepresentativeElement();
                                                    System.out.println(sameInd);
                                                    temp2=temp2+sameInd.toString()+"\n";

                                                // get the info about this specific individual
                                                    //Set<OWLLiteral> states = reasonerPel-
let.getDataPropertyValues(ind, hasStates );
                                                    //reasoner.
                                                    Set<OWLLiteral> states=
        ind.getDataPropertyValues(hasStates, exportedOntology);
                                    System.out.println("Visitor states:"+states.toString());




                                                    Set <OWLLiteral> visID = reasonerPel-
let.getDataPropertyValues(ind, hasVisitorID);
                                                    //getDataPropertyValues(in_Visitor1,
hasStates );

                                                    String visIDValue;
                                                    if(visID.isEmpty())
                                                    {
                                                        System.out.println("visID is un-
known");
                                                        temp2=temp2+"Unknow visitorID
for this individual"+"\n\n";

                                                    }
                                                    else
                                                    {


                                                    visIDValue =
visID.iterator().next().getLiteral();
                                                    System.out.println("this invidual hasID:
"+visIDValue);
                                                    temp2=temp2+"this invidual ha-
sID:"+visIDValue+"\n";

                                                    /////



                                                    String state =
states.iterator().next().getLiteral();
                                                    System.out.println("this invidual hasStates:
"+state);
                                                    temp2=temp2+"this individual has
states:"+state+"\n";
```

```
                                                    temp2=temp2+"\n"+"Invoke BecomeCustom-
erArtifact Service:"
                                                            +"\n"+
                                                            "The last state of this individual is:
"
                                                            +
                                                            origin_state[ind_count]
                                                            +
                                                            ", and the assigned visitorID is: "
                                                            +
                                                            visIDValue
                                                            +
                                                            "."+
                                                            "\n"
                                                            +
                                                            "The visitor artifact's current state
is:"
                                                            +
                                                            state
                                                            +
                                                            "."+"\n\n";

                                                    }
                                                    ind_count++;
                                                    temp=temp+temp2;


                                                System.out.println();
                                                System.out.println();
                                            }

                                            // get all instances of Customer individual
                                            NodeSet<OWLNamedIndividual> individuals = reason-
erPellet.getInstances(class_StudentArt, false);
                                            temp2="";
                                            ind_count=0;
                                            for(Node<OWLNamedIndividual> sameInd : individu-
als) {
                                                    // sameInd contains information about the in-
dividual (and all other individuals that were inferred to be the same)
                                                    temp2="";
                                                    OWLNamedIndividual ind =
sameInd.getRepresentativeElement();
                                                    System.out.println(sameInd);
                                                    temp2=temp2+sameInd.toString()+"\n";

                                                // get the info about this specific individual
                                                    //Set<OWLLiteral> states = reasonerPel-
let.getDataPropertyValues(ind, hasStates );
                                                    //reasoner.
                                                    Set<OWLLiteral> states=
        ind.getDataPropertyValues(hasStates, exportedOntology);
                                        System.out.println("customer statas:" +states.toString());



                                                    Set <OWLLiteral> cusID = reasonerPel-
let.getDataPropertyValues(ind, hasCustomerID);
                                                    //getDataPropertyValues(in_Visitor1,
hasStates );
```

```
                                                String cusIDValue;
                                                if(cusID.isEmpty())
                                                {
                                                        System.out.println("cusID is un-
known");
                                                        temp2=temp2+"Unknow Cus-
tomerID for this individual"+"\n\n";

                                                }
                                                else
                                                {

                                                cusIDValue =
cusID.iterator().next().getLiteral();
                                                System.out.println("Customer hasID:
"+cusIDValue);
                                                temp2=temp2+"this invidual ha-
sID:"+cusIDValue+"\n";

                                                /////

                                                String state =
states.iterator().next().getLiteral();
                                                System.out.println("this individual hasStates:
"+state);
                                                temp2=temp2+"this individual has
states:"+state+"\n";
                                                temp2=temp2+"\n"+"Invoke CreateCustom-
erArtifact Service:"
                                                        +"\n\n"+
                                                        "The last state of this individual is:
"
                                                        +
                                                        origin_state_cus[ind_count]
                                                        +
                                                        ", and the assigned CustomerID is:
"
                                                        +
                                                        cusIDValue
                                                        +
                                                        "."+
                                                        "\n"
                                                        +
                                                        "The visitor artifact's current state
is:"
                                                        +
                                                        state
                                                        +
                                                        ".";

                                                }
                                                temp=temp+temp2;
                                                ind_count++;

                                        System.out.println();
                                        System.out.println();
                                }

                                message_text.setText(temp+
```

```
                                                                    "\nnow you are customer and you
can select courses.\n" +
                                                                    "If you want to be a quit, please se-
lect the button CompleteVisitor to" +
                                                                    " invoke the completeVisitor Ser-
vice;"+"\n"+
                                                                    "Else you can select the button
Quit to quit the session!");

                                        } catch (OWLOntologyCreationException e1) {
                                                // TODO Auto-generated catch block
                                                e1.printStackTrace();
                                        }
                                        ////////////////////////////

                                }
                        });
```

Appendix D: OWL Ontology

D.1 Generic Ontology

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://ontology/test/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://ontology/test/">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <SubClassOf>
      <Class IRI="#AdvicerelatedServiceSystem"/>
      <Class IRI="#SystemType"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#AtomicServiceSystem"/>
      <Class IRI="#ServiceSystem"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#Competency"/>
      <Class IRI="#Input"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#Competency"/>
      <Class IRI="#Output"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#CompetencyCharacteristics"/>
      <Class IRI="#Characteristics"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ComplexServiceSystem"/>
      <Class IRI="#ServiceSystem"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#Customer"/>
      <Class IRI="#Role"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#CustomerInput"/>
      <Class IRI="#Input"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#CustomerInputCharacteristics"/>
      <Class IRI="#Characteristics"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#DigitalActor"/>
      <Class IRI="#ServiceActor"/>
   </SubClassOf>
   <SubClassOf>
```

```
    <Class IRI="#EnvDemandDrivenType"/>
    <Class IRI="#EnvironmentType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#EnvDomanClusterType"/>
    <Class IRI="#EnvironmentType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#EnvLooseCoupleType"/>
    <Class IRI="#EnvironmentType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#EnvOpenType"/>
    <Class IRI="#EnvironmentType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#EnvSelfOrganizeType"/>
    <Class IRI="#EnvironmentType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#EnvWebBasedType"/>
    <Class IRI="#EnvironmentType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#FunctionalResultrelatedServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#LeaserelatedServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Manufacturer"/>
    <Class IRI="#Role"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Organization"/>
    <Class IRI="#ServiceActor"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#OutsourcerelatedServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Partner"/>
    <Class IRI="#Role"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#PayperservicerelatedServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#People"/>
    <Class IRI="#ServiceActor"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#PolicyMaker"/>
    <Class IRI="#Role"/>
  </SubClassOf>
  <SubClassOf>
```

```
    <Class IRI="#ProductPoolingrelatedServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ProductrelatedServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Provider"/>
    <Class IRI="#Role"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ReceiveChannel"/>
    <Class IRI="#DeliveryChannel"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#RentrelatedServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Resource"/>
    <Class IRI="#Input"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Resource"/>
    <Class IRI="#Output"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ResourceCharacteristics"/>
    <Class IRI="#Characteristics"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#SendChannel"/>
    <Class IRI="#DeliveryChannel"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Service"/>
    <Class IRI="#Input"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Service"/>
    <Class IRI="#Output"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ServiceActorCharacteristics"/>
    <Class IRI="#Characteristics"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ServiceCharacteristics"/>
    <Class IRI="#Characteristics"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#SuperServiceSystem"/>
    <Class IRI="#SystemType"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#Technology"/>
    <Class IRI="#Input"/>
  </SubClassOf>
  <SubClassOf>
```

```
    <Class IRI="#Technology"/>
    <Class IRI="#Output"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#TechnologyCharacteristics"/>
    <Class IRI="#Characteristics"/>
  </SubClassOf>
  <SubObjectPropertyOf>
    <ObjectProperty IRI="#competencyRepresentedBy"/>
    <ObjectProperty IRI="#representedBy"/>
  </SubObjectPropertyOf>
  <SubObjectPropertyOf>
    <ObjectProperty IRI="#customerInputRepresentedBy"/>
    <ObjectProperty IRI="#representedBy"/>
  </SubObjectPropertyOf>
  <SubObjectPropertyOf>
    <ObjectProperty IRI="#deliveryChannelRepresentedBy"/>
    <ObjectProperty IRI="#representedBy"/>
  </SubObjectPropertyOf>
  <SubObjectPropertyOf>
    <ObjectProperty IRI="#hasInteraction"/>
    <ObjectProperty IRI="#hasCollaboration"/>
  </SubObjectPropertyOf>
  <SubObjectPropertyOf>
    <ObjectProperty IRI="#resourceRepresentedBy"/>
    <ObjectProperty IRI="#representedBy"/>
  </SubObjectPropertyOf>
  <SubObjectPropertyOf>
    <ObjectProperty IRI="#serviceRepresentedBy"/>
    <ObjectProperty IRI="#representedBy"/>
  </SubObjectPropertyOf>
  <SubObjectPropertyOf>
    <ObjectProperty IRI="#technologyRepresentedBy"/>
    <ObjectProperty IRI="#representedBy"/>
  </SubObjectPropertyOf>
  <InverseObjectProperties>
    <ObjectProperty IRI="#competencyRepresentedBy"/>
    <ObjectProperty IRI="#competencyCharacteristicsOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#hasCompetency"/>
    <ObjectProperty IRI="#competencyOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#hasContext"/>
    <ObjectProperty IRI="#contextOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#customerInputRepresentedBy"/>
    <ObjectProperty IRI="#customerInputCharacteristicsOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#hasCustomerInput"/>
    <ObjectProperty IRI="#customerInputOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#deliveryChannelRepresentedBy"/>
    <ObjectProperty IRI="#deliveryChannelCharacteristicsOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
```

```
    <ObjectProperty IRI="#hasResource"/>
    <ObjectProperty IRI="#resourceOf"/>
</InverseObjectProperties>
<InverseObjectProperties>
    <ObjectProperty IRI="#hasTechnology"/>
    <ObjectProperty IRI="#technologyOf"/>
</InverseObjectProperties>
<InverseObjectProperties>
    <ObjectProperty IRI="#resourceRepresentedBy"/>
    <ObjectProperty IRI="#resourceCharacteristicsOf"/>
</InverseObjectProperties>
<InverseObjectProperties>
    <ObjectProperty IRI="#serviceRepresentedBy"/>
    <ObjectProperty IRI="#serviceCharacteristicsOf"/>
</InverseObjectProperties>
<InverseObjectProperties>
    <ObjectProperty IRI="#technologyRepresentedBy"/>
    <ObjectProperty IRI="#technologyCharacteristicsOf"/>
</InverseObjectProperties>
<IrreflexiveObjectProperty>
    <ObjectProperty IRI="#belongsTo"/>
</IrreflexiveObjectProperty>
<IrreflexiveObjectProperty>
    <ObjectProperty IRI="#collaboratesWith"/>
</IrreflexiveObjectProperty>
<IrreflexiveObjectProperty>
    <ObjectProperty IRI="#competesWith"/>
</IrreflexiveObjectProperty>
<IrreflexiveObjectProperty>
    <ObjectProperty IRI="#exchangesArtifact"/>
</IrreflexiveObjectProperty>
<IrreflexiveObjectProperty>
    <ObjectProperty IRI="#inheritsFrom"/>
</IrreflexiveObjectProperty>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#addsValue"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#belongsTo"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#collaboratesWith"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#competencyCharacteristicsOf"/>
    <Class IRI="#CompetencyCharacteristics"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#competencyOf"/>
    <Class IRI="#Competency"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#competencyRepresentedBy"/>
    <Class IRI="#Competency"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#competesWith"/>
```

```
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#consumes"/>
    <Class IRI="#Customer"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#consumes"/>
    <Class IRI="#Partner"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#contextOf"/>
    <Class IRI="#Context"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#customerInputCharacteristicsOf"/>
    <Class IRI="#CustomerInputCharacteristics"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#customerInputOf"/>
    <Class IRI="#CustomerInput"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#customerInputRepresentedBy"/>
    <Class IRI="#CustomerInput"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#deliveryChannelCharacteristicsOf"/>
    <Class IRI="#DeliveryChannelCharacteristics"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#deliveryChannelOf"/>
    <Class IRI="#DeliveryChannel"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#deliveryChannelRepresentedBy"/>
    <Class IRI="#DeliveryChannel"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#exchangesArtifact"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#facilitates"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasCollaboration"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasCompetency"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasContentsOf"/>
    <Class IRI="#Interaction"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasContext"/>
```

```
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasCustomerInput"/>
    <Class IRI="#Customer"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasDeliveryChannel"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasEnvType"/>
    <Class IRI="#Environment"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasExtentLevel"/>
    <Class IRI="#Characteristics"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasFlow"/>
    <Class IRI="#Interaction"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasInput"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasInteraction"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasNeeds"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasOutput"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasPartner"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasResource"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasRole"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasServiceActor"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasSystemType"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#hasTechnology"/>
```

```
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#inheritsFrom"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#isCollaComposedOf"/>
    <Class IRI="#Collaboration"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#isComposedOf"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#isFrom"/>
    <Class IRI="#Flow"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#isTo"/>
    <Class IRI="#Flow"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#locatesIn"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#performs"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#provides"/>
    <Class IRI="#Partner"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#provides"/>
    <Class IRI="#Provider"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#representedBy"/>
    <Class IRI="#Competency"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#representedBy"/>
    <Class IRI="#CustomerInput"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#representedBy"/>
    <Class IRI="#DeliveryChannel"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#representedBy"/>
    <Class IRI="#Resource"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#representedBy"/>
    <Class IRI="#Service"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#representedBy"/>
```

```
    <Class IRI="#Technology"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#resourceCharacteristicsOf"/>
    <Class IRI="#ResourceCharacteristics"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#resourceOf"/>
    <Class IRI="#Resource"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#resourceRepresentedBy"/>
    <Class IRI="#Resource"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#serviceCharacteristicsOf"/>
    <Class IRI="#ServiceCharacteristics"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#serviceRepresentedBy"/>
    <Class IRI="#Service"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#specifiesChannel"/>
    <Class IRI="#Flow"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#supervises"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#technologyCharacteristicsOf"/>
    <Class IRI="#TechnologyCharacteristics"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#technologyOf"/>
    <Class IRI="#Technology"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#technologyRepresentedBy"/>
    <Class IRI="#Technology"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty IRI="#addsValue"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#belongsTo"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#collaboratesWith"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#competencyCharacteristicsOf"/>
    <Class IRI="#Competency"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#competencyOf"/>
```

```
    <Class IRI="#ServiceActor"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#competencyRepresentedBy"/>
    <Class IRI="#CompetencyCharacteristics"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#competesWith"/>
    <Class IRI="#ServiceSystem"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#consumes"/>
    <Class IRI="#Service"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#contextOf"/>
    <Class IRI="#ServiceSystem"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#customerInputCharacteristicsOf"/>
    <Class IRI="#CustomerInput"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#customerInputOf"/>
    <Class IRI="#Customer"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#customerInputRepresentedBy"/>
    <Class IRI="#CustomerInputCharacteristics"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#deliveryChannelCharacteristicsOf"/>
    <Class IRI="#DeliveryChannel"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#deliveryChannelOf"/>
    <Class IRI="#ServiceActor"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#deliveryChannelRepresentedBy"/>
    <Class IRI="#DeliveryChannelCharacteristics"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#exchangesArtifact"/>
    <Class IRI="#ServiceActor"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#facilitates"/>
    <Class IRI="#ServiceSystem"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasCollaboration"/>
    <Class IRI="#Collaboration"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasCompetency"/>
    <Class IRI="#Competency"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasContentsOf"/>
```

```
    <Class IRI="#Artifact"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasContext"/>
    <Class IRI="#Context"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasCustomerInput"/>
    <Class IRI="#CustomerInput"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasDeliveryChannel"/>
    <Class IRI="#DeliveryChannel"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasEnvType"/>
    <Class IRI="#EnvironmentType"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasExtentLevel"/>
    <Class IRI="#ExtentLevel"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasFlow"/>
    <Class IRI="#Flow"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasInput"/>
    <Class IRI="#Input"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasInteraction"/>
    <Class IRI="#Interaction"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasNeeds"/>
    <Class IRI="#Need"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasOutput"/>
    <Class IRI="#Output"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasPartner"/>
    <Class IRI="#ServiceActor"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasResource"/>
    <Class IRI="#Resource"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasRole"/>
    <Class IRI="#Role"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasServiceActor"/>
    <Class IRI="#ServiceActor"/>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#hasSystemType"/>
```

```
    <Class IRI="#SystemType"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#hasTechnology"/>
    <Class IRI="#Technology"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#inheritsFrom"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#isCollaComposedOf"/>
    <Class IRI="#Interaction"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#isComposedOf"/>
    <Class IRI="#ComplexServiceSystem"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#isFrom"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#isTo"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#locatesIn"/>
    <Class IRI="#Environment"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#performs"/>
    <Class IRI="#Service"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#provides"/>
    <Class IRI="#Service"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#representedBy"/>
    <Class IRI="#DeliveryChannelCharacteristics"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#resourceCharacteristicsOf"/>
    <Class IRI="#Resource"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#resourceOf"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#resourceRepresentedBy"/>
    <Class IRI="#ResourceCharacteristics"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#serviceCharacteristicsOf"/>
    <Class IRI="#Service"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#serviceRepresentedBy"/>
```

```
    <Class IRI="#ServiceCharacteristics"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#specifiesChannel"/>
    <Class IRI="#DeliveryChannel"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#supervises"/>
    <Class IRI="#ServiceSystem"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#technologyCharacteristicsOf"/>
    <Class IRI="#Technology"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#technologyOf"/>
    <Class IRI="#ServiceActor"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#technologyRepresentedBy"/>
    <Class IRI="#TechnologyCharacteristics"/>
</ObjectPropertyRange>
<DataPropertyDomain>
    <DataProperty IRI="#hasBoolValue"/>
    <Class IRI="#Characteristics"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#hasDateValue"/>
    <Class IRI="#Characteristics"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#hasFloatValue"/>
    <Class IRI="#Characteristics"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#hasIntValue"/>
    <Class IRI="#Characteristics"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#hasStates"/>
    <Class IRI="#Artifact"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#hasStringValue"/>
    <Class IRI="#Characteristics"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#isComparable"/>
    <Class IRI="#Resource"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#isTangible"/>
    <Class IRI="#Characteristics"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty IRI="#hasBoolValue"/>
    <Datatype abbreviatedIRI="xsd:boolean"/>
</DataPropertyRange>
<DataPropertyRange>
    <DataProperty IRI="#hasDateValue"/>
```

```
    <Datatype abbreviatedIRI="xsd:dateTime"/>
  </DataPropertyRange>
  <DataPropertyRange>
    <DataProperty IRI="#hasFloatValue"/>
    <Datatype abbreviatedIRI="xsd:float"/>
  </DataPropertyRange>
  <DataPropertyRange>
    <DataProperty IRI="#hasIntValue"/>
    <Datatype abbreviatedIRI="xsd:integer"/>
  </DataPropertyRange>
  <DataPropertyRange>
    <DataProperty IRI="#hasStates"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataPropertyRange>
  <DataPropertyRange>
    <DataProperty IRI="#hasStringValue"/>
    <Datatype abbreviatedIRI="xsd:string"/>
  </DataPropertyRange>
  <DataPropertyRange>
    <DataProperty IRI="#isComparable"/>
    <Datatype abbreviatedIRI="xsd:boolean"/>
  </DataPropertyRange>
  <DataPropertyRange>
    <DataProperty IRI="#isTangible"/>
    <Datatype abbreviatedIRI="xsd:boolean"/>
  </DataPropertyRange>
  <DLSafeRule>
    <Body>
      <ClassAtom>
        <Class IRI="#Partner"/>
        <Variable IRI="#z"/>
      </ClassAtom>
      <ClassAtom>
        <Class IRI="#Role"/>
        <Variable IRI="#z"/>
      </ClassAtom>
      <ClassAtom>
        <Class IRI="#ServiceActor"/>
        <Variable IRI="#y"/>
      </ClassAtom>
      <ClassAtom>
        <Class IRI="#ServiceSystem"/>
        <Variable IRI="#x"/>
      </ClassAtom>
      <ObjectPropertyAtom>
        <ObjectProperty IRI="#hasRole"/>
        <Variable IRI="#y"/>
        <Variable IRI="#z"/>
      </ObjectPropertyAtom>
      <ObjectPropertyAtom>
        <ObjectProperty IRI="#hasServiceActor"/>
        <Variable IRI="#x"/>
        <Variable IRI="#y"/>
      </ObjectPropertyAtom>
    </Body>
    <Head>
      <ClassAtom>
        <Class IRI="#ComplexServiceSystem"/>
        <Variable IRI="#x"/>
      </ClassAtom>
```

```
      </Head>
    </DLSafeRule>
    <DLSafeRule>
      <Body>
        <ObjectPropertyAtom>
          <ObjectProperty IRI="#hasExtentLevel"/>
          <Variable IRI="#x"/>
          <Variable IRI="#z"/>
        </ObjectPropertyAtom>
      </Body>
      <Head>
        <DataPropertyAtom>
          <DataProperty IRI="#isTangible"/>
          <Variable IRI="#x"/>
          <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#boolean">true</Literal>
        </DataPropertyAtom>
      </Head>
    </DLSafeRule>
</Ontology>
<!-- Generated by the OWL API (version 3.2.4.1806) http://owlapi.sourceforge.net -->
```

D.2 Example of Requirement Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<Ontology
    xml:base="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.w3.org/2002/07/owl#"
    ontologyIRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test">
  <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#failsIn"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Yong"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#TofelTest"/>
  </ObjectPropertyAssertion>
  <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#freshman"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Yong"/>
  </ClassAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#looksFor"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#TofelTest"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind632590779913759612"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#improve"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind632590779913759612"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#English"/>
  </ObjectPropertyAssertion>
  <ClassAssertion>
    <Class
```

```
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Course"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind632590779913759612"/>
</ClassAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isTo"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind182802749450795657"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isFrom"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind182802749450795657"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">15</Literal>
</DataPropertyAssertion>
<ClassAssertion>
  <Class
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AvailableTime"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind182802749450795657"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isOn"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind182802749450795657"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Monday"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isTo"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind330262509177784423"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isFrom"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind330262509177784423"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">15</Literal>
</DataPropertyAssertion>
<ClassAssertion>
  <Class
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AvailableTime"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind330262509177784423"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isOn"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind330262509177784423"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Tuesday"/>
</ObjectPropertyAssertion>
```

```
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isTo"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind695229118293173590"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isFrom"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind695229118293173590"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">15</Literal>
</DataPropertyAssertion>
<ClassAssertion>
  <Class
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AvailableTime"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind695229118293173590"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isOn"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind695229118293173590"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Wednesday"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isTo"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind3512272939621714 88"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isFrom"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind3512272939621714 88"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">15</Literal>
</DataPropertyAssertion>
<ClassAssertion>
  <Class
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AvailableTime"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind3512272939621714 88"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isOn"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind3512272939621714 88"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Thursday"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isTo"/>
  <NamedIndividual
```

```
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind243688342255743560"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isFrom"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind243688342255743560"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">15</Literal>
</DataPropertyAssertion>
<ClassAssertion>
  <Class
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AvailableTime"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind243688342255743560"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isOn"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind243688342255743560"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Friday"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isTo"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind61361969176850395"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isFrom"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind61361969176850395"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">15</Literal>
</DataPropertyAssertion>
<ClassAssertion>
  <Class
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AvailableTime"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind61361969176850395"/>
</ClassAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isOn"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind61361969176850395"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Saturday"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isTo"/>
  <NamedIndividual
        IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind386919377791799265"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
```

```
    <DataProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isFrom"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind386919377791799265"/>
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">15</Literal>
   </DataPropertyAssertion>
   <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AvailableTime"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind386919377791799265"/>
   </ClassAssertion>
   <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#isOn"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind386919377791799265"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Sunday"/>
   </ObjectPropertyAssertion>
   <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#have"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Yong"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Freely_Access_Online_Education_Resource
s"/>
   </ObjectPropertyAssertion>
   <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#have"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Freely_Access_Online_Education_Resource
s"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind935427655481810153"/>
   </ObjectPropertyAssertion>
   <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#updateAuthroty"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind935427655481810153"/>
   </ClassAssertion>
   <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#have"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Freely_Access_Online_Education_Resource
s"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind668181356860434024"/>
   </ObjectPropertyAssertion>
   <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#readAuthroty"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind668181356860434024"/>
   </ClassAssertion>
   <ObjectPropertyAssertion>
```

```
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#have"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Freely_Access_Online_Education_Resource
s"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind141206776005882582"/>
  </ObjectPropertyAssertion>
  <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#deliveryChannel"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind141206776005882582"/>
  </ClassAssertion>
  <ClassAssertion>
    <ObjectMinCardinality cardinality="5">
      <ObjectProperty
              IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#have"/>
      <Class
              IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#feature"/>
    </ObjectMinCardinality>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Freely_Access_Online_Education_Resource
s"/>
  </ClassAssertion>
  <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#TechnologyCharacteristic"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Freely_Access_Online_Education_Resource
s"/>
  </ClassAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#achieve"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Access_Resources"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Freely_Access_Online_Education_Resource
s"/>
  </ObjectPropertyAssertion>
  <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#InternalTechnology"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Access_Resources"/>
  </ClassAssertion>
  <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#feature"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AccessTime"/>
  </ClassAssertion>
  <ClassAssertion>
    <Class
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#feature"/>
    <NamedIndividual
            IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Availability"/>
  </ClassAssertion>
  <ClassAssertion>
```

```
  <Class
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#feature"/>
  <NamedIndividual
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#AccessLocation"/>
 </ClassAssertion>
 <ClassAssertion>
  <ObjectMinCardinality cardinality="24">
   <ObjectProperty
           IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#accessTime"/>
   <Class
           IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#hour"/>
  </ObjectMinCardinality>
  <NamedIndividual
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Access_Resource"/>
 </ClassAssertion>
 <ObjectPropertyAssertion>
  <ObjectProperty
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#studiesVia"/>
  <NamedIndividual
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Yong"/>
  <NamedIndividual
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Web"/>
 </ObjectPropertyAssertion>
 <ClassAssertion>
  <ObjectSomeValuesFrom>
   <ObjectProperty
           IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#studiesVia"/>
   <Class
           IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#MobileNetwork"/>
  </ObjectSomeValuesFrom>
  <NamedIndividual
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Yong"/>
 </ClassAssertion>
 <ClassAssertion>
  <Class
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#unlimitedLocation"/>
  <NamedIndividual
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind494558263500937500"/>
 </ClassAssertion>
 <ClassAssertion>
  <Class
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#accessLocation"/>
  <NamedIndividual
          IRI="http://attempto.ifi.uzh.ch/ontologies/owlswrl/test#Ind494558263500937500"/>
 </ClassAssertion>
</Ontology>
```

N° d'ordre 2012ISAL0024 Année 2012

Résumé de Thèse

Modélisation et Conception de "IT-enabled Service Systems" dirigés par les exigences et la collaboration

Présentée devant
L'institut national des sciences appliquées de Lyon

Pour obtenir
Le grade de docteur

Formation doctorale : Informatique

École doctorale : École Doctorale Informatique et Mathématiques

Par
Yong Peng
(Ingénieur en Informatique)

Soutenue le 22 mars 2012 devant la Commission d'examen

Jury

TATA Samir	Professeur à TELECOM SudParis	Président
VALLESPIR Bruno	Professeur à l'Université Bordeaux 1	Rapporteur
FRONT Agnès	Maître de Conférences HDR à l'Université Pierre Mendès France de Grenoble	Rapporteur
BENSLIMANE Djamal	Professeur à l'Université Claude Bernard	Examinateur
BENALI Khalid	Maître de Conférences HDR à l'Université de Lorrain	Examinateur
BADR Youakim	Maître de Conférences à l'INSA de Lyon	Directeur de Thèse
BIENNIER Frédérique	Professeur à l'INSA de Lyon	Co-directrice de Thèse

Laboratoire de recherche: Laboratoire d'Informatique en Image et Systèmes d'information (LIRIS)

TABLE DES MATIERES

TABLE DES FIGURES

1 Introduction

Les services sont caractérisés par des propriétés telles que l'intangibilité, l'hétérogénéité, l'inséparabilité et la périssabilité (Rathmell 1966, Zeithaml et al. 1985, Wolak et al. 1998). L'hétérogénéité reflète la grande variabilité dans les offres de services. L'inséparabilité indique que la production et de la consommation de services sont simultanées tandis que la périssabilité indique que les services ne peuvent pas être stockés ou transportés pour une utilisation différée. Du fait de ces caractéristiques, il n'est pas évident que les acteurs économiques puissent comprendre les services, les concevoir et les produire de la même façon qu'ils produisent des biens tangibles. Le secteur des services occupe aujourd'hui une position dominante dans les économies des pays développés. Malheureusement, la productivité est très basse dans le secteur de services. Par comparaison à la productivité dans le secteur manufacturier, nous nous sommes focalisés dans nos activités de recherche sur les services « facilités » par les technologies de l'information et la communication désignés par la suite avec l'acronyme ITeS à partir du sigle en anglais, IT-enabled Services. Les ITeSs sont caractérisés par leurs aspects économiques, sociétales et techniques. Ils nécessitent souvent la collaboration entre différents acteurs pour co-créer de la valeur ajoutée.

En générale, les ITeSs sont considérés comme des services traditionnels (par exemple, la formation, le commerce, l'ingénierie, ...) qui sont supportés par les TICs. De ce fait, les TICs sont utilisées comme un élément d'innovation plutôt que comme un outil pour d'automatisation des processus. Comparés aux services traditionnels du secteur tertiaire, les services facilités par les technologies de l'information et des communications suscitent un intérêt croissant des clients et fournisseurs d'une part du fait de l'automatisation des processus et d'autre part grâce aux nouveaux canaux de communication (Internet, réseaux mobiles, ...) que ces services supportent. De ce fait, les ITeS co-créent de la valeur ajoutée grâce à la collaboration entre les clients et les fournisseurs lors de la conception et de l'approvisionnement des services. Cet enrichissement des services traditionnels conduit à une remise à plat des méthodes actuelles de conception de biens et de services. En effet, ces méthodes ne permettent pas de répondre aux exigences imposées par ce contexte de collaboration multidisciplinaire qui intègre les entreprises, les technologies de l'information et de la communication et les acteurs sociaux. Les caractéristiques intrinsèques des services (à savoir

l'intangibilité, l'inséparabilité, la périssabilité et la simultanéité) et leur nature sociotechnique requiert à la fois une méthodologie de conception globale dirigée par les exigences des clients (en vue de leur satisfaction) et une approche systémique prenant en compte la dimension collaborative, le cycle de vie des services et les changements organisationnels, métiers et technologiques.

Pour innover dans les ITeS et créer de nouveaux services en tenant compte la collaboration inter-acteurs, la pluridisciplinarité et l'intégration de divers aspects métiers, technologique et ceux liés aux ressources humaines, nous avons identifié les questions de recherche suivantes:

1- Comment partager une compréhension commune de la structure de l'information dans le système de services pour faire face aux ambiguïtés dues en contexte multi-métiers et faciliter la collaboration des acteurs de service?

2- Comment développer une méthodologie de conception collaborative pour les ITeSs en considérant les quatre caractéristiques des services, les exigences des clients (et leur satisfaction) et la collaboration entre les acteurs de l'écosystème de services pour assurer la co-production de ces services ?

3- Comment exprimer concrètement les besoins du client qui sont souvent imprécis et non structurés pour réduire les différences entre le service perçu et le service fourni d'une part, et comment réduire les différences entre les divers acteurs qui utilisent des modèles et / ou des langages différents pour collaborer?

4- Comment représenter les connaissances explicites échangées durant la collaboration entre les acteurs et comment modéliser cette collaboration?

Pour faire face à ces enjeux, nous proposons une méthodologie descendante pour modéliser et concevoir un système de services dirigé par les exigences des clients et supportant la collaboration entre tous les acteurs afin de permettre la co-création du système de services. Notre méthodologie de conception repose sur une approche pluridisciplinaire et offre un ensemble de modèles inter-connectés (modèle de référence de service, modèle d'exigence et modèle de collaboration) ce qui permet d'une part, de donner de la flexibilité au système et de le rendre adaptable en cas de changements des exigences, et d'autre part de supporter la collaboration entre tous les acteurs. Le modèle de référence offre une description des différentes dimensions du système de services (ontologique, caractéristiques et systémique) et

explicite ainsi les connaissances liées aux différents domaines. En se basant sur le modèle d'exigences, les besoins du client sont spécifiés dans un langage partagé et compréhensible par tous les acteurs. Ceci permet la propagation de ces exigences dans tout le cycle de vie des services et leur diffusion à tous les acteurs. Le modèle de collaboration préconise une approche guidée par les données c-à-d une approche opposée aux processus métiers collaboratifs traditionnels - ce qui favorise l'interopérabilité technique et sémantique et augmente la stabilité du système face aux changements. Notre approche de collaboration s'appuie sur les canaux de communication qui engendrent des flux d'objets métiers (Business Artefact) selon lesquels des règles d'affaires sont générées afin d'invoquer les composants logiciels sous-jacents.

2 État d'Art

Avant de détailler notre contribution, nous présentons un état de l'art sur les méthodologies et les modèles pour concevoir des systèmes de services, définir les exigences et modéliser la collaboration.

Nous distinguons deux approches pour concevoir les systèmes de services respectivement l'approche d'affaire et l'approche systémique. L'approche d'affaire s'appuie sur le domaine disciplinaire du Marketing et propose des méthodes pour concevoir des services ou produits traditionnels en prenant en compte la participation des clients. Cette approche s'avère inutile pour les ITeSs parce qu'elle se concentre sur le rôle des décideurs du système de services plutôt que sur la co-création de valeur par collaboration. L'approche systémique en revanche intègre la notion de multi-vues et considère un service comme un système socio-technique. Elle offre aussi une description de tous les composants du système et de leurs dépendances, ce qui peut faciliter l'innovation. Pour prendre en compte les contraintes d'interopérabilité entre les acteurs, nous avons aussi étudié les ontologies qui sont utilisées pour formaliser les connaissances et faire face aux ambiguïtés conceptuelles et raisonner sur les systèmes de logiciels traditionnels (Smith, 1998, Jurisica et al. 1999, Touzi 2007). Plus particulièrement, Baida (2006) a présenté un modèle d'ontologie pour les services afin de formaliser les connaissances des domaines d'affaires et techniques et automatiser la composition de services. Cependant, ce modèle d'ontologie formalise les connaissances sur les services plutôt que les systèmes de service. De ce fait, il ne décrit pas les composants du système de services et leurs interdépendances dans une perspective de systématisation. Pour pallier les problèmes liés aux méthodologies de conception de services, nous avons étudié les méthodologies traditionnelles développées dans le cadre de systèmes d'information. Nous avons constaté que les approches top-down et

bottom-up sont largement adoptées dans le cadre des architectures d'entreprise comme les cadres « Information System Architecture » (Sowa et Zachman (1992), « The Open Group Architecture Framework » (TOGAF 2009), et « Federal Enterprise Architecture » (FEAF 1999). Les approches top-down conviennent souvent à la bonne compréhension des stratégies d'affaires. Ces approches mettent également en évidence le rôle dominant des prestataires de services lors de la conception du système d'information d'entreprise. Bien que les clients prennent partiellement part aux activités des concepteurs (par exemple dans la phase d'analyse des besoins et l'évaluation des services), ils ne participent pas à la co-création de la valeur.

Les approches top-down sont destinées à un contexte particulier d'utilisation comme indiqué par (Gartner Group 2009) ; "*worked well when applied to complex, fixed functions — that is, human artefacts, such as aircraft, ships, buildings, computers and even EA software. However, it works poorly when applied to an equally wide variety of domains because they do not behave in a predictable way*". Or, les systèmes pour les ITeS sont « dirigés » par la collaboration puisque les acteurs du système de services collaborent entre eux pour co-créer solution. Ces collaborations sont très agiles et dépendent des contextes de différents acteurs aux moments des interactions. Les approches top-down ne sont pas assez souples pour supporter l'agilité dans les processus des entreprises et ne permettent de mobiliser les ressources internes et externes dynamiquement afin de s'adapter aux changements techniques, organisationnels, métiers, sociétaux.

A l'inverse, les approches bottom-up (Sabatier 1986, Kautz et al. 1994) sont initiées aux niveaux opérationnels et techniques et puis étendent progressivement vers le niveau métier puis niveau stratégique. Ces approches visent généralement à construire les systèmes d'information à partir de sous-systèmes de différents métiers (ventes, achats, production, services après vente, …). En raison de l'absence de stratégie globale de haut niveau et d'une compréhension commune par les acteurs de différents métiers, ces approches sont difficiles à adapter aux services qui nécessitent les collaborations à tous les niveaux afin de co-créer la valeur-ajoutée.

En outre, en raison de la nature économique sociétale, technique et systématique des systèmes de service, les acteurs provenant de différentes disciplines utilisent des modèles et des langages différents et ont des vues différentes sur les systèmes de services quand ils veulent collaborer pour atteindre un objectif commun (la co-création de la valeur). Les intérêts de ces acteurs sont très divers et parfois conflictuels. Leurs comportements sont dynamiques et

imprévisibles comme le constate Gall et al. (2010). Pour cette raison, le recours à une approche dite « middle-out » permet et facilite la collaboration entre les acteurs du système de services (les clients, les experts métiers, les spécialistes IT et les développeurs). Cette approche pourrait être considérée comme une stratégie de médiation favorisant les échanges, l'innovation et l'adaptation aux besoins des clients Parag et Janda (2010) également estiment que "les approches middle-out ne sont pas nécessairement des approches top-down et bottom-up, mais plutôt une démarche supplémentaire qui pourrait être plus efficace face aux changements dans les exigences et les contextes d'exécution".

En conclusion, parce que l'approche middle-out n'est ni tout à fait stratégique, ni technique, les acteurs dans les phases de conception et de production de services peuvent se référer à ce niveau de granularité intermédiaire pour gérer leurs affaires et les adapter aux changements. En effet, les acteurs au niveau technique peuvent se référer au niveau intermédiaire de l'approche middle-out pour implémenter des logiciels et automatiser les processus métiers. Dans le domaine des ITeSs, l'approche middle-out requiert un modèle de référence pour 1) illustrer la méthodologie de conception des systèmes de services ayant des points de vue statique et dynamique, 2) fournir une compréhension commune afin de réduire les ambiguïtés conceptuelles entre les acteurs du système de services au niveau stratégique et au niveau stratégique, 3) formaliser les connaissances des domaines métiers et techniques, afin d'automatiser la co-création de la valeur ajoutée. Ce modèle de référence doit aussi permettre de relier les acteurs dans une chaîne de valeur selon une approche systémique et les aider à se concentrer sur leurs activités principales et les modes d'interaction via les canaux de communication.

Concernant les modèles de spécification des exigences des clients, nous constatons que ces clients jouent un rôle significatif dans les systèmes de service car ils participent à toutes les phases du cycle de vie de service, ce n'est pas le cas dans les systèmes logiciels ou des systèmes d'information où les clients se participent partiellement à la phase de spécification. C'est pour cela que l'identification des exigences dans un contexte de collaboration devient un objectif critique pour atteindre la satisfaction de clients dans la phase de consommation de services. Les systèmes ITeS sont souvent dirigés par les exigences et requièrent donc une approche d'ingénierie pour identifier et préciser ces exigences dans un contexte de collaboration. Malheureusement, les modèles d'exigences traditionnels (Rolland 1993, Schmitt 1993, Pohl 1993, Lee 1996, Toffolon et Dakhli 1999, Donzelli 2004, Baida 2006)

sont principalement développés pour le domaine du génie logiciel et ils soulèvent plusieurs problèmes quand ils sont appliqués aux ITeSSs:

1) Les clients identifient difficilement leurs exigences de manière appropriée pour les services attendus à cause de leurs lacunes dans des domaines variés tels que les NTIC.

2) Des écarts sont constatés entre les services perçus par les fournisseurs et les services souhaités par les clients. En ce sens, ces acteurs doivent collaborer en respectant un modèle d'exigences suffisamment expressif et rigoureux pour spécifier et valider les besoins et les propriétés fonctionnelles et non fonctionnelles de services.

3) Les spécialistes IT jouent un rôle important dans l'innovation et l'intégration des sous-systèmes selon les modèles d'exigences conçus par les différents acteurs. Or, les modèles d'exigences traditionnels ne prennent pas en considération les aspects sociétaux, techniques et métiers.

En parallèle, nous avons étudié les approches de modélisation de collaboration dynamique qui se caractérise par le fait qu'elle ne peut pas être définie ou déterminée à l'avance par des processus métiers. Pour les prestataires de service, la satisfaction du client reste l'objectif ultime à atteindre. Cependant, la stratégie à adopter dans un contexte de collaboration n'est pas claire au début : c'est au fur et mesure que les prestataires interagissent avec les clients qu'ils déterminent progressivement les décisions à prendre. Les modalités de collaboration sont souples et dynamiques et non prédéfinies et déterministes. En outre, dans les ITeSs, des acteurs du systèmes de service échangent essentiellement des objets métiers (exprimés en tant que connaissances explicites) ceci nécessite une structuration de ces connaissances afin d'automatiser leur traitement par les infrastructures logicielles. Les documents, les dossiers d'affaires, les notes et les courriels se sont que quelques exemples pour enregistrer les connaissances explicites échangées. Par conséquent, afin d'automatiser ou de semi-automatiser les processus de collaboration, il est nécessaire de les formaliser, de les structurer et de concevoir des mécanismes dirigés par le traitement de ces connaissances. Cette approche est différente des approches traditionnelles qu'on trouve dans le domaine de la conception desEn effet, les approches traditionnelles sont souvent basées sur les activités ou les processus métiers dans lesquels chaque activité est parfois implémentée comme un service Web pour résoudre les problèmes d'interopérabilité (Tidwell 2000) et augmenter l'agilité du système d'information. De ce fait, ces approches s'appuient sur une phase d'analyse pour définir à priori un ensemble d'activités et spécifier leur d'ordre d'exécution. Des approches alternatives (par exemple, Kumaran et al. 2008, Muller et al. 2008, et Narendra et al. 2009) utilisent les données métiers

pour modéliser simultanément l'aspect métier et la collaboration. De même l'approche selon des « Business Artefact » proposé par (Nigam et Caswell 2003) est une approche centrée sur les données. Elle modélise les objets métier sous forme d'une liste d'attributs, d'un ensemble d'états et d'un cycle de vie décrivant les transitions entre ces états. Si cette approche se révèle un moyen utile pour aborder les problèmes d'interopérabilité (métier et technique), elle n'est jamais utilisée dans la modélisation de collaboration dynamique (adhoc).

Cet état de l'art a montré le besoin d'une méthodologie de conception des systèmes de services et particulièrement adaptées aux services facilités par les technologies de l'information et des communications (ITeS). Une vue systématique paraît indispensable pour identifier les composants d'un tel système de services et leurs interdépendances où l'aspect pluridisciplinaire reste un obstacle dans la conception, la collaboration et la spécification des exigences dans les systèmes ITeSs.

3 Contributions

Pour répondre à ces enjeux, nous proposons une méthodologie descendante pour modéliser et concevoir un système de services dirigé par les exigences des clients et supportant la collaboration entre tous les acteurs afin de permettre la co-création de ce système. Notre méthodologie repose sur une approche pluridisciplinaire et offre un ensemble de modèles interconnectés (modèle de référence de service, modèle d'exigences et modèle de collaboration). Ceci permet d'une part de donner de la flexibilité au système et de le rendre adaptable en cas de changements des exigences et d'autre part de supporter la collaboration entre tous les acteurs. Dans les paragraphes suivant la présentation de notre méthodologie, nous détaillons ces différents modèles.

3.1 Méthodologie de Conception Middle-out

Notre méthodologie middle-out consiste de six phases (Figure 1).

1- Phase de préparation: il s'agit de formaliser les connaissances dans les ITeSSs et offrir une compréhension générale afin de raisonner sur les composants du système de services. Cette phase contient deux sous phases : 1) le développement des ontologies génériques abstraites et le développement des ontologies de domaine.

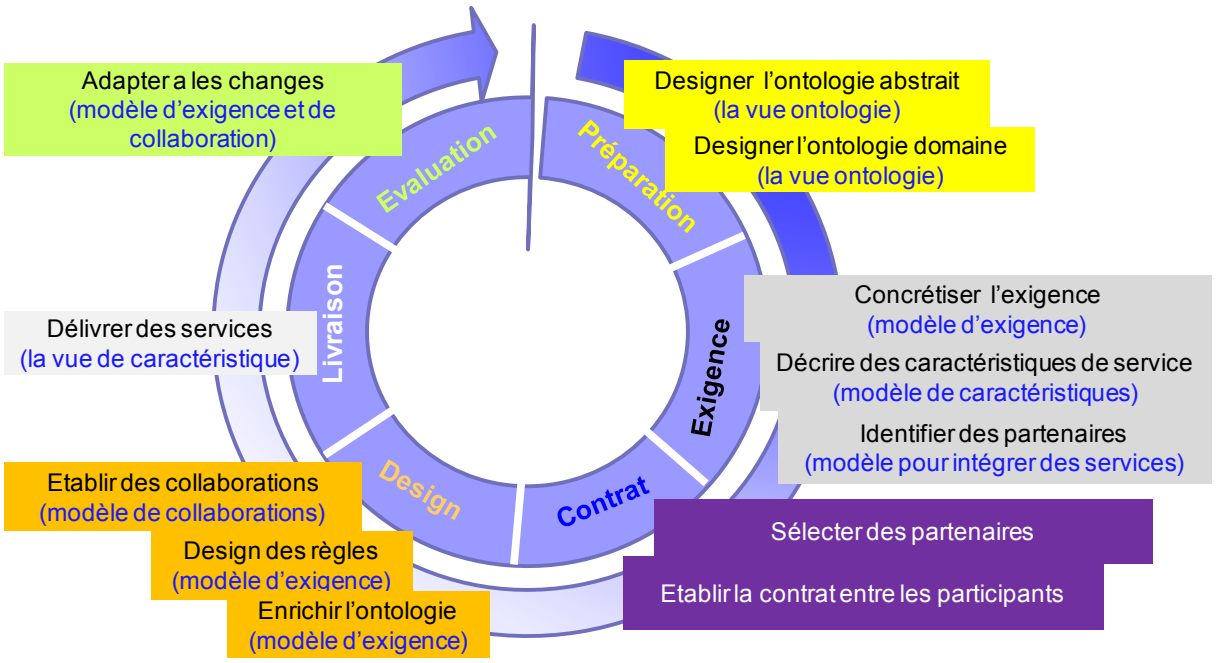


Figure 1: La méthodologie middle-out

2- cation des exigences: cette phase se concerne sur la concrétisation des exigences et leur représentation sous forme de caractéristiques pour le(s) service(s) attendu(s).

3- Phase de contractualisation : dans cette phase, le fournisseur de services identifie les partenaires et établit un contrat avec eux pour co-créer les service demandés.

4- Phase de la conception: cette phase concerne l'enrichissement des ontologies et la création des règles métiers selon les exigences. Cette phase comprend également la spécification des exigences et la conception des flux de collaboration.

5- Phase d'approvisionnement: cette phase consiste à mettre en place des canaux de communication pour faciliter les échanges entre le fournisseur de services et les clients. Si les besoins des clients changent, le système doit s'adapter aux changements.

6- Phase d'évaluation: cette phase concerne la qualité de service et les mesures à mettre en place pour évaluer la satisfaction des clients et adapter le service aux changements.

3.2 Le Modèle de Référence

Les systèmes ITeS sont très complexes à concevoir à cause des caractéristiques des services (l'intangibilité, l'hétérogénéité, l'inséparabilité et la périssabilité), de la collaboration pluridisciplinaire et leur nature sociétale et technique. Afin de réduire cette complexité, nous

proposons d'analyser le système de services selon plusieurs vues interconnectées pour révéler les différentes facettes d'un service. Pour cela, nous introduisons un modèle de référence pour offrir une compréhension partagée du système de service et raisonner sur ses composants. Le modèle de référence comprend trois vues (Figure 2) : la vue systématique, la vue ontologique et la vue à base de caractéristiques de service. Ces vues constituent un niveau intermédiaire, ni technique ni stratégique, a pour objectif de faciliter la collaboration de tous les acteurs de service concernés. Le modèle de référence est conçu pour être extensible. Par exemple, de nouveaux comme la vue de valeur-ajoutée ou une vue d'agrégation de plusieurs systèmes de service pour construire un nouveau système de services.

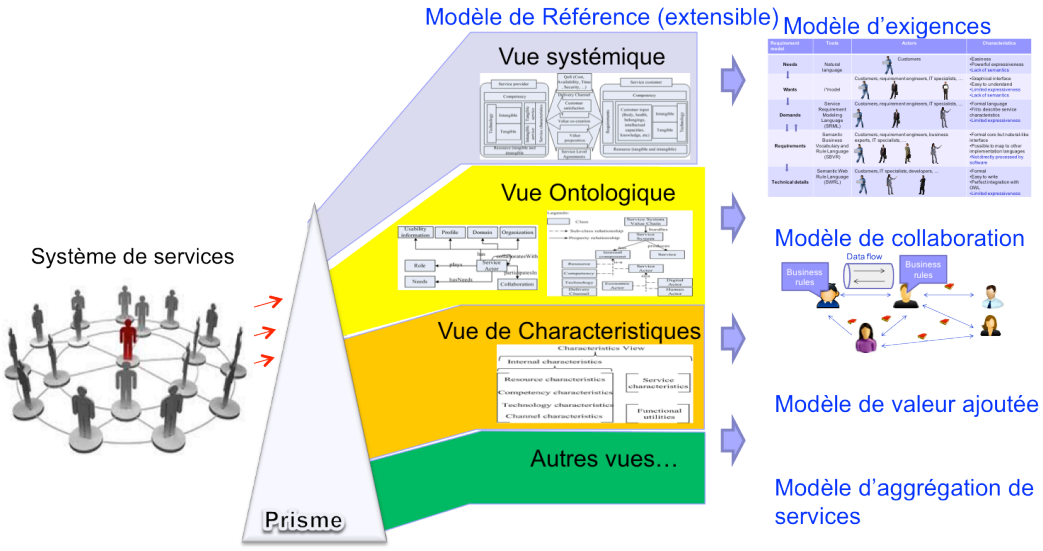


Figure 2: Modèle de référence

La vue systémique fournit une représentation systématique d'un ITeS en identifiant les composants fondamentaux du système (les acteurs, les ressources tangibles et intangibles, les compétences, les technologies, les canaux et les processus) et décrivant les interactions entre fournisseurs et clients d'une part et entre les différents composants d'autre part. Comme la vue systématique est descriptive (énumération des composants), la vue ontologique la complète par un ensemble de concepts, une logique descriptive et un vocabulaire commun partagé par tous les acteurs de services de différents domaines. Cette vue est en particulier composée de concepts génériques, de leurs instances et de l'ensemble de contraintes qui imposent des restrictions sur leurs relations d'interdépendance (Noy et McGuinness, 2001). La vue ontologique a pour objectif;

1) partager une compréhension commune de la structure du système de service.

2) permettre la réutilisation des connaissances du domaine d'application.

3) Séparer les connaissances génériques sur les services (ontologies de haut niveau) des connaissances opérationnelles propre à un domaine d'application (ontologies de domaine).

Nous avons groupé les ontologies en deux catégories (Figure 3) :

- Les ontologies abstraites de haut niveau décrivent les concepts génériques et les terminologies d'un système de service. Elles comprennent les ontologies de système, les ontologies des acteurs, les ontologies d'exigences, et les ontologies de collaboration.

- Les ontologies de domaine décrivent un domaine d'application particulier (par exemple les services financiers, les services d'ingénierie, ..).

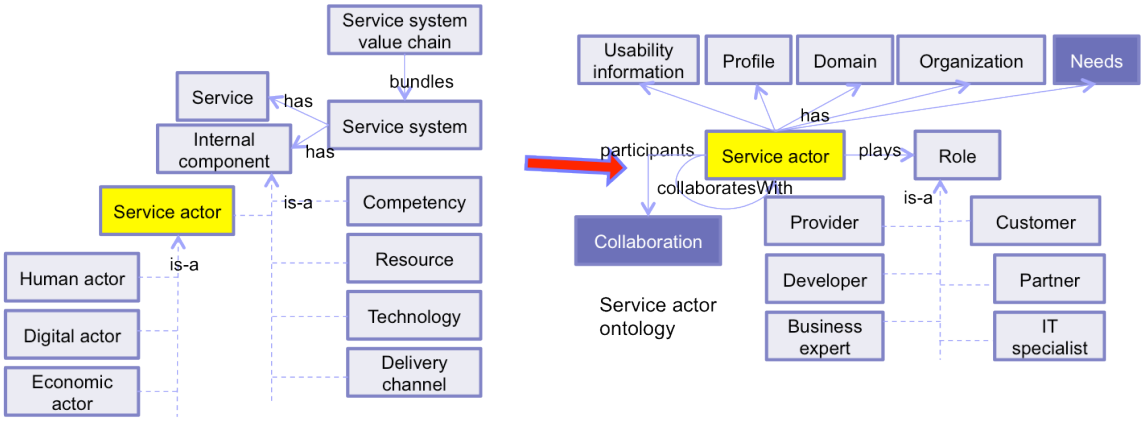


Figure 3: La vue ontologique

La vue à base de caractéristiques de service décrit le service avec des propriétés qualitatives. Il s'agit d'une représentation matricielle où chaque élément de la matrice est une propriété non-fonctionnelle (qualitative). La notion de caractéristique est initialement développée dans le domaine manufacturier (Lancaster, 1966) pour décrire les caractéristiques qu'un produit incarne. Saviotti et Metcalfe (1984) ont constaté que les constructeurs obtiennent des avantages concurrentiels en offrant aux utilisateurs des combinaisons particulières de caractéristiques décrivant les aspects techniques du produit (caractéristiques techniques), l'utilité du produit rendue par les méthodes de production (caractéristiques de processus).

Dans notre contribution, nous avons utilisé la vue à base de caractéristiques de service pour développer le modèle d'exigences et nous l'avons étendu aux différents composants internes du système de services en distinguant les caractéristiques de ressources (tangible et intangible), les caractéristiques de compétences, les caractéristique technologiques et les caractéristiques des canaux de communication. Un système de service expose ainsi ses caractéristiques finales aux clients (ou aux autres systèmes de services). Ces derniers n'ont pas à s'intéresser réellement à la structure complexe de ses composants internes ni à la façon dont les composants internes

sont intégrés pour fournir les caractéristiques finales. De ce fait, cette vue permet aux différents acteurs de se concentrer sur leur métier et sur les interactions avec d'autres acteurs plutôt que de se confronter à la complexité des composants internes du système de services.

3.3 Le Modèle d'Exigences

Dans les systèmes de service, les acteurs (les clients, les partenaires, les experts, les spécialistes, les développeurs, …) utilisent des modèles et langages différents pour spécifier les exigences selon lesquelles ils collaborent pour concevoir et ensuite consommer les services. Dans la pratique, ceci crée un écart entre le service perçu par les acteurs et le service (à rendre) attendu par le client. Notre méthodologie de conception collaborative met en avant les exigences en permettant l'intégration des modèles d'exigences de tous les acteurs et à différents niveaux stratégiques, métiers, opérationnels et techniques.

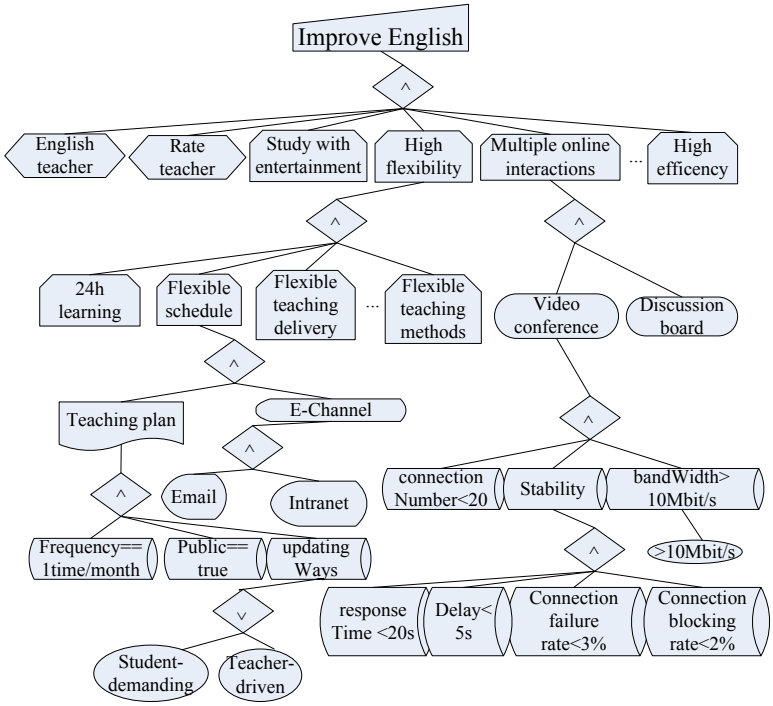


Figure 4: Le modèle i* étendu

Notre méthodologie a pour objectif de réduire les différences entre les acteurs qui doivent collaborer et conserver la diversité des modèles ou langages qu'ils utilisent et qui conviennent à leur savoir-faire. Pour cela, nous introduisons un modèle d'exigences selon une approche descendante (top-down) pour concrétiser des besoins peu clairs, imprécis et non-structurés exprimés par un client en un ensemble d'exigences claires, précises et structurées manipulables par les développeurs et/ou les responsables métier. Notre modèle d'exigences favorise également une collaboration horizontale interdisciplinaire puisqu'il est basé sur un modèle de

référence partagé par tous les acteurs permettant ainsi l'intégration de toutes les ressources, les compétences, les technologies et les moyens de communication.

Notre modèle d'exigences s'appuie sur un ensemble de standards et de spécifications que nous avons développé ou étendu à savoir le modèle *i**(Figure 4), le langage de notations des exigences de services (SRML) (Figure 5), les règles et les vocabulaires métiers (SBVR), les règles sémantiques (SWRL), les ontologies génériques et les ontologies de domaine qui sont fournies par la vue ontologique.

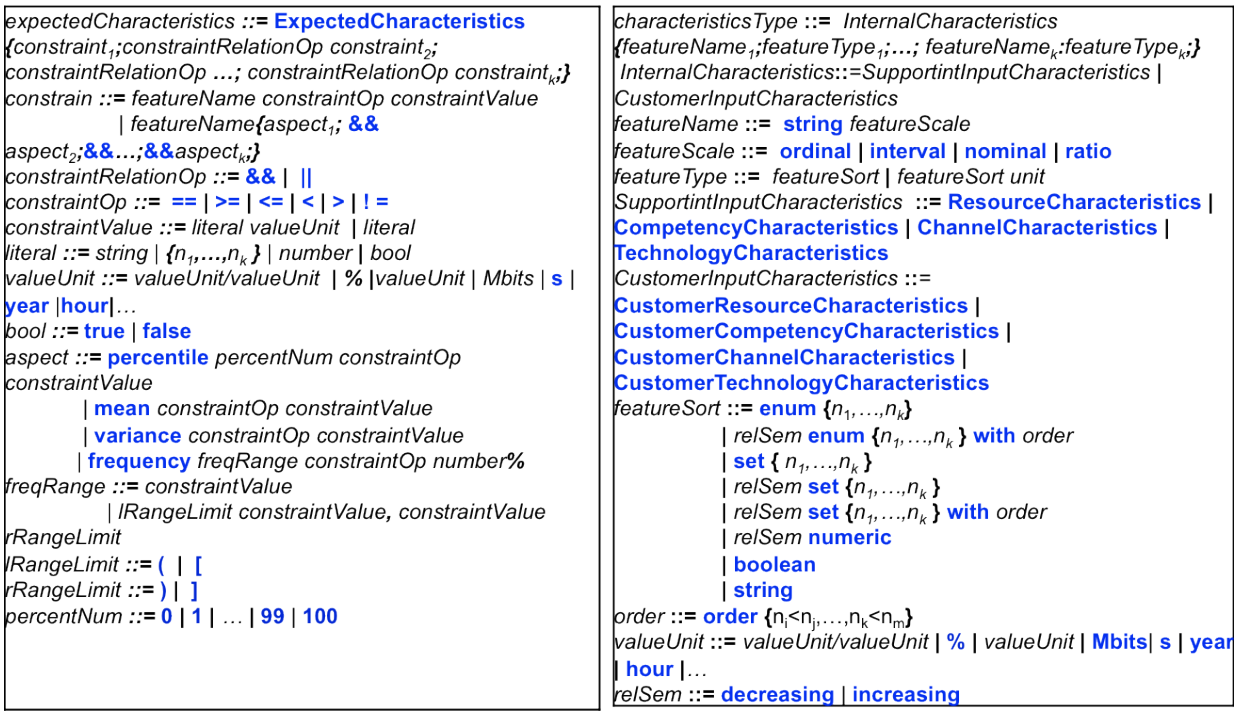


Figure 5: Le modèle SRML

Le modèle i* a été initialement introduit par Yu (1995) dans le domaine du génie logiciel comme un modèle dirigé par les « buts » permettant de décomposer un problème complexe en un sous-ensemble de problèmes moins complexes. Nous avons étendu ce modèle pour l'appliquer aux systèmes des ITeS en ajoutant un ensemble de nouveaux concepts liés aux composants des services et en tenant compte des caractéristiques des services, des exigences et des contraintes. Bien que ce modèle offre une représentation graphique qui facilite la compréhension d'un système de services par des clients novices, il devient parfois une source d'ambiguïté sémantique puisqu'il permet aux acteurs des services de déduire des interprétations différentes sur les exigences. En ce sens, nous avons complété ce modèle avec le langage formel, nommé SRML (Service Requirement Management Language), en représentant les exigences d'une façon claire et précise. SRML est basé sur le langage de modélisation de qualité de services (QML) développé dans le domaine du génie logiciel (Frolund et

Koistien,1998). Dans notre contribution, nous avons utilisé SRML pour concrétiser les besoins du client en un ensemble de caractéristiques quantitatives, qualitatives et fournir des directives aux concepteurs au moment de la collaboration, et mesurer la satisfaction du client au moment de l'approvisionnement des ITeSs.

Toutefois, SRML est assez technique, ce qui le rend difficile à comprendre par les acteurs non-techniciens (notamment les clients). À cette fin, nous avons proposé l'utilisation de SBVR. Ceci suppose donc de transformer les expressions en SRML en expressions SBVR (SBVR 2008). SBVR est un langage structuré proche du langage naturel, ce qui peut faciliter son interprétation par tous les acteurs des services. En s'appuyant sur les expressions SBVR, les spécialistes IT peuvent collaborer avec des experts métiers et générer un ensemble de nouveaux concepts et d'axiomes pour enrichir les ontologies de système de service. De plus, SBVR permet d'identifier les acteurs de services et de générer les règles métiers exprimées dans notre contribution par le Web sémantique (SWRL) (Horrocks et al. 2004). Ces règles métiers invoquent les ressources (par exemple les services web), mettent à jour les « artefact métiers » et mobilisent les composants du système de services.

3.4 Le Modèle de Collaboration

Dans un contexte de système de services, les collaborations entre les acteurs sont souvent dynamiques et imprévisibles. Les approches dirigées par les activités permettent de modéliser les processus métiers qui peuvent être définis dans la phase de conception. Malheureusement, ces approches ne sont que peu adaptées aux collaborations dynamiques. En revanche, les approches dirigées par données sont de plus en plus utilisées dans la modélisation de collaborations centrées sur les échanges entre les acteurs. Un « artefact métier » est tout simplement un document métier autonome comprenant une liste d'attributs, un ensemble d'états et un le cycle de vie qui indique comment les attributs sont modifiés en changeant les états. Le concept d'artefact métier permet de décrire comment les données échangées dans les collaborations sont mises à jour suite aux changements.

En effet, les acteurs coopèrent en échangeant des connaissances métiers (explicites) que nous avons modélisées avec les artefacts métiers. Nous avons défini également des modèles de collaboration dynamiques à base d'artefacts métiers lors de la conception et de l'approvisionnement de services. En effet, une collaboration dynamique est modélisée comme un flux d'artefact métier où les acteurs participent et partagent des règles métier qui gouvernent le traitement des artefacts (Figure 6). Pour ce qui concerne l'implémentation,

chaque artefact est associé à un service Web générique permettant son émission et sa réception. Pendant l'exécution, quand un acteur de service reçoit un artefact, il s'appuie sur les ontologies développées et les règles métier pour raisonner sur l'invocation ou la mobilisation des composants du système de service et mettre à jour les artefact métiers et leurs états selon les transition décrites dans leur cycle de vie.

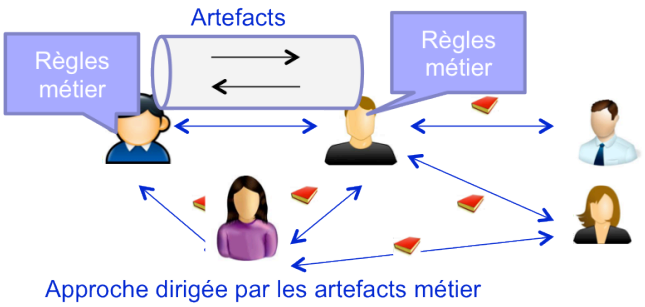


Figure 6: Collaboration basée sur les échanges des artefacts métier

Nous avons constaté que la collaboration dans le cadre d'un service comprend plusieurs tâches répétitives. Pour cela, nous avons adopté des patrons pour capitaliser les contextes d'utilisation afin de les réutiliser et améliorer l'efficacité de notre méthodologie de conception de système de services. Chaque patron de collaboration est composé d'une structure générique comprenant: un contexte, un objectif décrivant le problème et une solution générique (y compris les artefact métiers, les ontologies concernées, les règles métiers contrôlant leurs modifications et les règles de transitions d'états, …).

Comparativement aux approches traditionnelles dirigées par les données, l'intérêt principal de notre approche unifiée d'exigences et de collaboration se traduit par :

1- Une compréhension commune de la collaboration selon une vision multi-desciplinaire inter-acteurs facilitant ainsi leur raisonnement sur les composants de systèmes de services.

2- L'intégration des artefacts métier dans les différentes étapes de la méthodologie pour représenter les connaissances métiers échangées. En effet, l'utilisation des artefacts aux niveaux métier, opérationnel et technologique permet d'interconnecter ces niveaux et d'établir des interfaces d'échanges entre les composants du système de service sans préciser comment ces artefacts seront traités, le traitement des artefact étant délégué à chaque acteur qui devrait respecter les états et les règles de transition

4 Implémentation

Nous avons présenté dans notre contribution une architecture technique pour la mise en œuvre d'un système d'ITeS. Nous avons également réalisé un prototype pour simuler plusieurs scenarios de collaboration entre des acteurs de service échangeant des artefact métier. L'architecture technique est composée de cinq modules:

1- Module de développement d'ontologies: ce module est utilisé pour développer des ontologies génériques, ontologies de domaine et des ontologies reliées aux différents composants de systèmes de service.

2- Module de gestion des exigences: ce module est utilisé pour concrétiser les besoins imprécis des clients et spécifier les exigences définies par les fournisseurs de service.

3- Module de génération d'artefacts: ce module est utilisé pour générer les artefacts métier à partir des exigences et enrichir les ontologies de système par les concepts générés par les artefacts métiers afin de supporter les processus de collaboration.

4- Module de collaboration: ce module est utilisé pour raisonner sur les concepts ontologiques (classes, instances, attributs, contraintes, ..) et appliquer les règles métier pour mobiliser les ressources tangibles et intangibles.

5- Module de caractéristiques de service: ce module est utilisé pour enregistrer des caractéristiques de services et sélectionner les partenaires.

Nous avons également développé des algorithmes pour transformer les expressions SBVR en OWL et générer les artefacts métier. Les interfaces du prototype sont respectivement présentées dans les Figure 7 et 8.

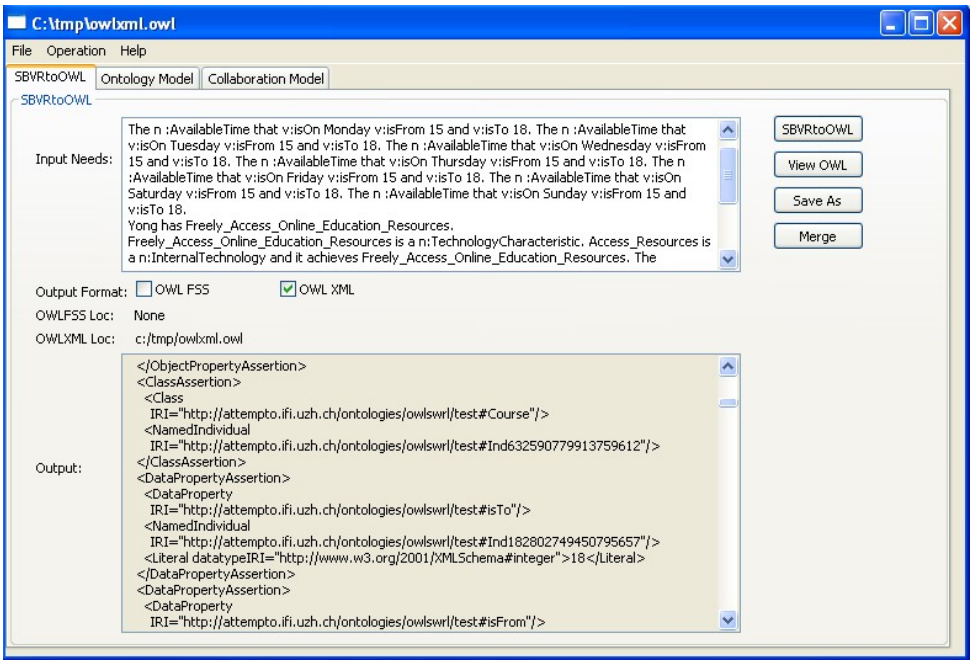


Figure 7: La transformation de SBVR en OWL

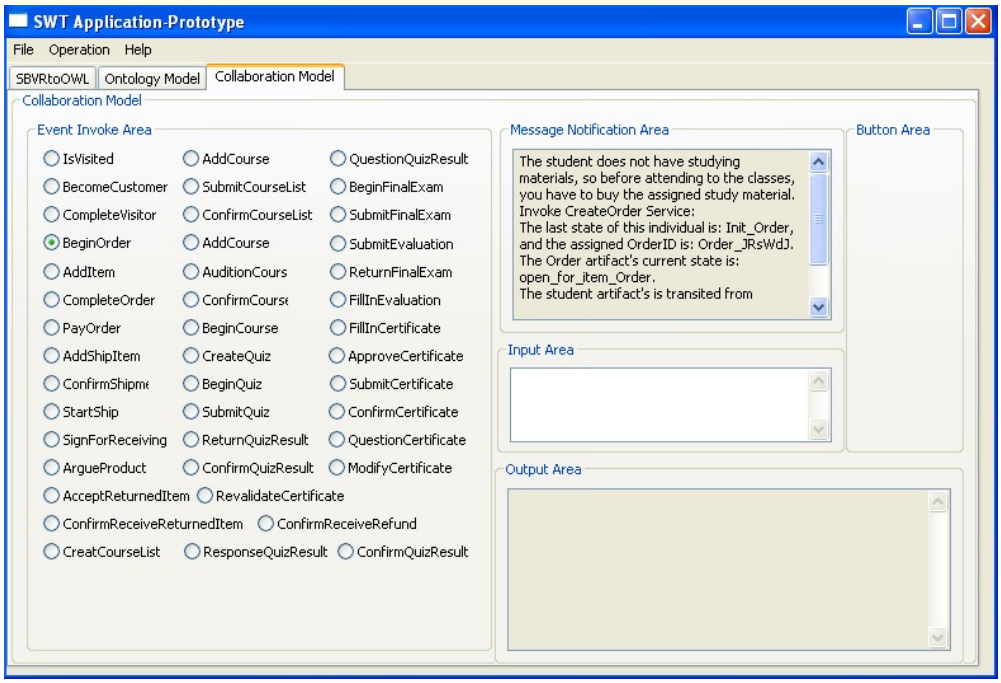


Figure 8: Collaborations basées sur les échanges des artefact métiers

5 Conclusion

La complexité croissante liée aux interactions des acteurs de différentes disciplines et aux caractéristiques de services rendent difficiles les tâches de conception et d'approvisionnement de systèmes d'ITeSs. Dans un tel contexte, les acteurs devraient se concentrer sur les connaissances métier échangées plutôt que sur les activités qui doivent mettre en place pour

échanger et traiter ces connaissances ainsi que la gestion de structures internes de leurs systèmes de service. Dans notre contribution, nous avons proposé « une vue à base de caractéristiques » pour assurer une abstraction des structures internes des services et se concentrer sur les objectifs des interactions via les modèles d'exigences tel que le modèle i*.

En outre, les systèmes ITeS sont évolutifs, collaboratifs et dirigés par la satisfaction des exigences des clients et leurs satisfactions. Notre contribution suit la récente veille technologique du groupe Gartner (2010) qui favorise une approche hybride permettant la conception de systèmes d'information émergents adaptés aux changements. Pour répondre aux transformations des entreprises et améliorer les processus d'innovation et la stratégie. Notre vision à long terme considère un système de services comme un écosystème (système de systèmes) où les variations sont nombreuses et parfois imprévisibles. Ceci qui rend la conception est difficile en utilisant les architectures d'entreprise traditionnelles souvent considérées comme des architectures monolithiques. Un écosystème de services requiert une approche holistique pour co-créer de nouveaux services en intégrant des services existants et favoriser l'interopérabilité aux niveaux métiers, organisationnels et opérationnels. De ce fait, plusieurs axes d'amélioration et de recherche sont envisageables à savoir :

1 – L'étude d'un mécanisme d'agrégation de systèmes de service à base de leurs de caractéristiques. Ce mécanisme pourrait traiter les caractéristiques en s'appuyant sur des mesures de comparaison entre les caractéristiques exposés par plusieurs systèmes d'ITeSs et le système de service qu'on souhaite fournir. Ceci permet de chercher des partenaires de service en faisant correspondre leurs caractéristiques de service et développer l'approche d'écosystème de services.

2- L'enrichissement des modèles d'exigences proposés ou la proposition d'un modèle unique d'exigences pour spécifier les besoins et déduire automatiquement les artefacts métier tout en enrichissant les ontologies et les processus de collaborations. La granularité des artefacts métier pourrait être contrôlée dans ce nouveau contexte de spécification des besoins.

3 – L'extension du modèle de référence avec un ensemble de vues qui devraient s'articuler avec les vues que nous avons proposés. Ceci comprend une vue de valeur-ajoutée permettant de mesurer les bénéfices pour les fournisseurs ou les consommateurs de services, une vue d'assemblage de blocs de composants internes du système de service pour faciliter la réutilisation et l'assemblage dans des nouveaux domaines d'application etc...

4- La gestion des artefact métiers et les patrons de collaboration reste un axe de recherche à améliorer en tenant compte des correspondances entre les artefacts publiques et privées et l'enrichissement de patrons par des règles métiers réutilisables dans des contextes similaires de conception et de production de systèmes de service.

6 Bibliographie

Rathmell J. M. (1966) What Is Meant by Services? Journal of Marketing, 30(4), pp. 32-36.

Zeithaml V.A., Parasuraman A. and Berry L.L. (1985) Problems and Strategies in Services Marketing. Journal of Marketing, 49 (2), pp. 33-46.

Wolak R., Kalaftis S. and Harris P. (1998) an Investigation into Four Characteristics of Services. Journal of Empirical Generalisations in Marketing Science, 3(2), pp. 22-41.

Spohrer J., Maglio P.P, Bailey J. et al. (2007) Steps toward a Science of Service Systems. IEEE. Computer, 40 (1), pp. 71-77.

Smith B. (1998) Basic Concepts of Formal Ontology. In: Guarino N (Ed.), Proceedings of Formal Ontology in Information Systems (FOIS'98), 6-8 June Trento, Italy. Amsterdam: IOS Press, pp. 19-28.

Jurisica I., Mylopoulos J. and Yu E. (1999) Using Ontologies for Knowledge Management: An Information Systems Perspective, In: Proceedings of the 62nd Annual Meeting of the American Society for Information Science(ASIS '99), Oct. 31 – Nov 4, Washington, D. C. Medford : American Society for Information Science by Information Today, pp. 482-496;

Touzi J., Bénaben F. and Pingaud H. (2007) Interoperability Through Model Based Generation: the Case of the Collaborative IS. In Doumeingts G., Müller J., Morel G., and Vallespir B. (Eds.), Enterprise Interoperability: New Challenges and Approaches. Springer, pp. 407-416.

Baida Z. (2006) Software-aided Service Bundling Intelligent Methods & Tools for Graphical Service Modeling. Ph.D thesis. Amsterdam, Netherlands : Vrije Universiteit, 301 p. ISBN-10: 90-810622-1-2.

Sowa J.F. and Zachman J.A. (1992) Extending and Formalizing the Framework for Information Systems Architecture. IBM Systems Journal, 31(3), pp.590-616.

TOGAF (2009) Welcome to TOGAF Version 9 -- The Open Group Architecture Framework. "http://pubs.opengroZup.org/architecture/togaf9-doc/arch/". lasted accessed in December 2011.

FEAF (1999) Federal Enterprise Architecture Framework Version 1.1. "http://www.cio.gov/documents/fedarch1.pdf". lasted accessed in December 2011.

Sabatier P. (1986) Top-Down and Bottom-Up Approaches to Implementation Research: a Critical Analysis and Suggested Synthesis. Journal of Public Policy, 6(1), pp. 21-48.

Gartner Group (2009) Gartner Identifies New Approach for Enterprise Architecture. "http://www.gartner.com/it/page.jsp?id=1124112". lasted accessed in December 2011.

Kautz H., Selman B. and Coen M. (1994) Bottom-Up Design of Software Agents. Communication of the ACM, 37(7), pp. 143-147.

Gall N., Newman D., Allega P. et al. (2010) Introducing Hybrid Thinking for Transformation, Innovation and Strategy. White paper, ID Number: G00172065. Gartner Group. "http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1352013 ". lasted accessed in December 2011.

Parag Y. and Janda K. (2010) Midstream and Sideways: Considering a Middle-out Approach to Changing Energy Demand. In: Sussex energy group conference, Oxford, U.K., Oxford University, February 2010, Proceedings: Oxford, United Kingdom, Environmental Change Institute, 16 p.

Rolland C. (1993) Modeling the Requirements Engineering Process. In: 3rd European-Japanese Seminar on Information Modelling and Knowledge Bases, May 31-June 3, Budapest, Hungary. Amsterdam : IOS Press, pp. 86-97.

Schmitt J.R. (1993) Product Modeling for Requirements Engineering Process Modeling. In: IFIP WG 8.1 Conf. on Information Systems Development Process, September 1-3, Como, Italy. Amsterdam : North-Holland, pp. 231-245.

Pohl K. (1993) The Three Dimensions of Requirements Engineering. In Rolland C., Bodart F., Cauvet C. (Eds), Fifth International Conference on Advanced Information Systems Engineering (CAiSE'93), June 8-11, Paris, France. Paris: Springer-Verlag, pp.175-292.

Lee M.J. (1996) Foundations of the Win-Win Requirements Negotiation System. Ph.D. Thesis. California : University of South California, 192 p.

Toffolon C. and Dakhli S. (1999) Requirements Elicitation Process: An Iterative Approach based on the Spiral Model. In: XVIth International Symposium on Computer Information and Sciences (ISCIS'99), 1999 October 18-20, Izmir, Turquie. New York : Elsevier Science Publishers, 13 p.

Donzelli P. (2004) A Goal-driven and Agent-based Requirements Engineering Framework. Requirements Eng, 9(1), pp. 16–39.

Tidwell D. (2000) Web Services: The Web's Next Revolution. IBM Tutorial. "http://www.cn-java.com/download/book/wsbasics-a4.pdf". last accessed in October 2011.

Nigam A. and Caswell N. S. (2003) Business Artifacts: An Approach to Operational Specification. IBM Systems Journal, 42(3): 428- 445.

Kumaran S., Liu R., and Wu F.Y. (2008) On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'2008), June 18-20, Montpellier, France. Berlin: Springer-Verlag, pp. 32-47.

Muller D., Reichert M. and Herbst J. (2008) A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. In*:* Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'2008), Montpellier, France. Berlin: Springer-Verlag, pp. 48-63.

Narendra N.C., Badr Y., Thiran P. and Maamar Z. (2009) Towards a Unified Approach for Business Process Modeling Using Context-based Artifacts and Web Services. In: Proceedings of the IEEE International Conference on Services Computing (SCC'2009), September 21-25, Bangalore, India. Washington, DC : IEEE Computer Society, pp.332-339.

Noy N.F. and McGuinness D.L. (2001) Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report SMI-2001-0880. Toronto: University of Toronto, 25 p.

Lancaster K.J. (1966) A New Approach to Consumer Theory. Journal of Political Economy, 74(2), pp. 133-157.

Saviotti P.P. and Metcalfe J.S. (1984) A Theoretical Approach to the Construction of Technological Output Indicators. Research Policy, 13 (3), pp. 141-151.

SBVR (2008) SBVR 1.0 documentation. "http://www.omg.org/spec/SBVR/1.0/". lasted accessed in December 2011.

Yu E. (1995) Modelling Strategic Relationships for Process Reengineering Ph.D. Thesis. Dept. of Computer Science, University of Toronto. 1995

Frolund S. and Koistinen J (1998) QML: A Language for Quality of Service Specification. Technical Report HPL-98-10, Palo Alto. California : Hewlett-Packard, 63 p.

Horrocks I. et al. (2004) SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C. "http://www.w3.org/Submission/SWRL/". lasted accessed in December 2011.

Gartner Group (2010) Gartner Says Hybrid Thinking for Enterprise Architecture Can Help Organisations Embrace Transformation, Innovation and Strategy "http://www.gartner.com/it/page.jsp?id=1368613". last accessed in December 2011.