Formalisation des Processus de l'Ingénierie Système : Proposition d'une méthode d'adaptation des processus génériques à différents contextes d'application

Samuel Rochet

N° d'ordre : 901

THÈSE

présentée à

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE TOULOUSE

pour l'obtention du

DOCTORAT

de l'Université de Toulouse délivré par l'Institut National des Sciences Appliquées de Toulouse

Spécialité : Systèmes Automatiques

par

Samuel ROCHET

LABORATOIRE TOULOUSAIN DE TECHNOLOGIE ET D'INGÉNIERIE DES SYSTÈMES
École Doctorale Systèmes

Titre de la thèse :

- Formalisation des processus de l'Ingénierie Système : -

Proposition d'une méthode d'adaptation des processus génériques à différents contextes d'application

Soutenue le 26 Novembre 2007, devant le jury :

Rapporteurs :	Colette ROLLAND	Professeur à l'Université Paris I Panthéon Sorbonne
	Alain BERNARD	Professeur à l'École Centrale de Nantes
Examinateurs :	Claude BARON	Professeur à l'INSA de Toulouse - Directrice de thèse
	Daniel ESTEVE	Directeur de recherche émérite au LAAS-CNRS
	Christian PERCEBOIS	Professeur à l'Université Paul Sabatier Toulouse III
	Éric BONJOUR	Maître de Conférences à l'Université de Franche-Comté
Membres invités :	Michel ALDANONDO	Professeur à l'École des Mines d'Albi-Carmaux
	Lionel JAOUEN	Industriel, Airbus

Remerciements

Je tiens à remercier ma directrice de thèse Mme Claude Baron pour toute la confiance qu'elle a placé en moi depuis mon DEA. C'est elle qui m'a poussé et qui a permis l'aboutissement de ces travaux. Merci aussi aux membres du Laboratoire Toulousain de Technologie et d'Ingénierie des Systèmes (LATTIS) pour leur accueil et leur soutien scientifique et moral durant les années que j'ai partagées avec eux.

Je remercie Mme Colette Rolland et M. Alain Bernard les rapporteurs de cette thèse pour leurs commentaires et leurs questions qui sont venus enrichir ma réflexion ainsi que les autres membres de mon jury de thèse MM. Daniel Esteve, Michel Aldanondo, Christian Percebois, Éric Bonjour et Lionel Jaouen pour m'avoir fait l'honneur d'assister à ma soutenance.

Parmi eux, je tiens à remercier tout particulièrement M. Daniel Esteve pour son intuition et ses nombreux conseils qui m'ont évité bien des égarements.

Merci à Hugues Malgouyres qui a vécu avec moi ses années de thèse et dont les travaux ont trouvé une application dans la validation de mes modèles.

Ce travail a pu voir le jour grâce à l'appui de la société Airbus. Je remercie tous les membres du département SW, en particulier M. Lionel Jaouen qui m'a accueilli dans son équipe et autorisé à consulter les projets actuels pour formuler plus correctement la problématique système.

Merci à Isabelle, ma première lectrice, qui m'est devenue indispensable.

Merci enfin à toute ma famille et à mes amis qui m'ont tour à tour supporté, aidé ou simplement diverti un instant.

Table des matières

Introduction Générale

Le progrès technique se fonde sur une dynamique de compréhension toujours plus approfondie. Il se crée régulièrement de nouveaux procédés et de nouveaux systèmes toujours plus complexes. Ces systèmes « artefacts » sont aujourd'hui multifonctionnels et intègrent des services divers créant des secteurs industriels et socio–économiques à l'échelle mondiale : la santé, les transports routiers, aériens et maritimes, les télécommunications, la défense... Cela a conduit à la création d'un spectre de disciplines scientifiques nouvelles : les « sciences de l'ingénieur » où l'on va trouver les méthodologies, les méthodes et les outils pour concevoir, fabriquer et exploiter ces nouveaux systèmes et systèmes de systèmes. Alors que dans les grands systèmes naturels (astrophysique, météorologie, sciences de la terre, *etc.*) l'Homme est dans la prédiction et l'observation, cherchant à comprendre, l'Homme est, dans les systèmes « artefacts », concepteur puis client et enfin exploitant. Dans les deux cas, l'Homme est face à la complexité. Dans l'étude des phénomènes naturels, on mesure cette maîtrise par la qualité des prédictions des comportements. Dans la conception de systèmes, on mesure la maîtrise grâce à des **critères de « conformité » aux « exigences » d'un cahier des charges**, dans le respect de l'environnement (développement durable).

Dans cette réflexion et thèse, on suppose que les technologies devant être mises en œuvre sont disponibles et accessibles. La question ouverte est de savoir : **comment définir et mettre en œuvre toutes les activités nécessaires au développement du produit complexe que l'on projette ?**

Il s'agit donc d'abord de définir une *méthodologie*, une *démarche* en adéquation avec des outils existants (ou sur le point de le devenir) pour la mettre efficacement en œuvre : c'est un objectif central de l'ingénierie système. On comprendra mieux l'importance de la disponibilité de cette méthodologie en considérant le chemin parcouru ces dernières années en modèles de systèmes. Il était commun d'attribuer la création d'un système à un inventeur entouré d'une équipe. Le succès est aujourd'hui toujours le résultat d'une maîtrise technologique mais surtout le résultat d'une bonne organisation des *activités* (définition 1) !

Définition 1 : *Activité* [Wik07]

Ensemble distinct d'actions identifiées, organisé selon un processus logique, observable en tant que tel. Il peut désigner aussi une ou plusieurs tâches exécutées par un ou plusieurs employés à l'intérieur d'un processus.

Idéalement, une bonne ingénierie des systèmes va considérer toutes les « activités » utiles identifiables, les « planifier » selon des « étapes » parfaitement « spécifiées » et strictement « vérifiables », en utilisant au mieux tous les « moyens » disponibles pour développer un produit « conforme » dans le respect des « exigences » et des « contraintes » du projet.

Pour accéder à cet idéal, l'approche qui s'est imposée est assez classique :

1. inventorier tous les acteurs directs et indirects du développement (contexte),
2. inventorier les activités utiles (classification en métier),
3. dégager, pour chaque activité, les processus de gestion spécialisés les mieux adaptés,
4. faire un travail de surveillance et de supervision pour bien articuler les différents processus entre eux selon des critères de performance et de conformité aux spécifications.

La démarche qui s'est montrée la plus efficace est collective : **rassembler toutes les expériences possibles, standardiser, normaliser...** pour capitaliser les acquis, échanger, et disposer de critères commun d'évaluation des performances et de la qualité. Cette recommandation inclut naturellement les exigences et les contraintes liées à la totalité du cycle de vie du produit et doit intégrer toutes considérations interférant couramment avec la bonne gestion de l'environnement.

Selon notre avis, les points 1, 2 et 3 sont sinon complets et maîtrisés du moins suffisamment avancés pour permettre des conceptions et des réalisations très complexes. Par contre, le point 4 est encore laissé largement entre les mains de l'Homme. **Notre parti pris, pour franchir cette problématique d'interaction de processus multiples, est d'imaginer et de proposer une approche globale de définition d'un modèle multiprocessus formel. Pour cela, nous considérons qu'une solution générique à l'ingénierie d'un système est définie par les travaux antérieurs (points 1, 2 et 3). Notre proposition est de travailler sur une représentation formalisée de cette solution à partir de laquelle sont dérivées des solutions dédiées, adaptées à différents contextes d'application.**

Cela conduit à :

1. partir d'un scénario donné par les normes d'ingénierie système : dans notre cas, l'EIA-632,
2. en extraire un modèle global générique d'interaction multiprocessus,
3. l'enrichir de toutes les particularités propres à l'entreprise et spécifiques au projet,
4. afin d'aboutir à un modèle spécifique que l'on puisse exploiter opérationnellement pour :
   - l'analyse du bon déroulement des choses,
   - le diagnostic de certaines insuffisances,
   - l'organisation du suivi de conformité,
   - et, à terme, une planification globale.

Notre objectif et notre contribution est ici :

1. de rechercher une formalisation des processus pour laquelle nous recommandons :
   - de s'appuyer sur les acquis d'un standard tel que l'EIA-632,
   - de choisir un langage de modélisation tel que SPEM/UML qui satisfasse aux exigences génériques de l'interopérabilité des méthodes et des outils,
2. et de proposer une façon de les enrichir et de les spécialiser.

Au delà de notre contribution, nous pouvons d'ores et déjà envisager la nécessité de renforcer la modélisation dans deux directions : la temporisation et l'évaluation des coûts, jusqu'à pouvoir générer des solutions alternatives et les comparer [RB06].

Ce travail s'inscrit dans une dynamique régionale et mondiale qui vise à une plus grande maîtrise du développement des grands systèmes. Nous avons bénéficié, pour poser la problématique au plus juste, de l'appui de la société Airbus qui nous a accueilli dans ses locaux et a permis de formuler notre problématique sur les aspects multi–processus de l'ingénierie système. Nous nous sommes appuyés sur une réflexion commune entre notre laboratoire d'accueil, le LATTIS, et le LAAS–CNRS, notamment les groupes MIS [Gui07] et ISI [KS07, DMSS07, YS06, MJS05], ce qui nous a permis d'avoir une double vision des problèmes : celle de l'organisation des projets et celle de la conception technique des systèmes. Le travail de thèse de Citlalih Gutierrez [GE07], auquel nous avons participé dans le développement d'un outil GESOS d'optimisation parmi différentes alternatives de réalisation du processus lors de la planification, et des projets comme TOPCASED et ATLAS ont aussi aidé au développement de notre travail.

Dans le premier chapitre de cette thèse, nous présenterons les différentes disciplines qui fondent nos travaux. Nous reviendrons sur la discipline de « l'ingénierie système » et sur les grandes normes qui s'y réfèrent. Nous rentrerons dans le détail de l'EIA-632 que nous avons sélectionné comme standard de référence pour illustrer nos propositions.

Le second chapitre traitera de la première de nos contributions : Après un rapide rappel sur l'ingénierie des modèles, ce chapitre explorera comment formaliser les normes d'ingénierie systèmes et présentera un modèle des processus de l'EIA-632 en SPEM/UML.

A ce stade, les processus formalisés ne sont pas encore opérationnels. Le chapitre 3 traitera d'une première partie de notre seconde contribution : la spécialisation des processus. Il présentera une démarche de modélisation permettant l'intégration de spécificités liées au domaine d'activité et au projet via des transformations de modèles.

Enfin, le chapitre 4 viendra compléter la démarche du chapitre 3 selon plusieurs aspects : il présentera une méthode et un outil original de vérification de propriétés des modèles de processus utilisés en complément de la démarche de modélisation. Il donnera un exemple d'application de la démarche sur un projet universitaire. Il conclura par des réflexions sur les améliorations et les extensions qui peuvent venir enrichir la méthode que nous avons développée dans la perspective d'une ingénierie système guidée vers les modèles enfin opérationnelle.

Chapitre 1

Cadre général des travaux et problématique

« *Ces deux petits morceaux sont écrits il y a longtemps, et, tout médiocres qu'ils sont, je ne serai pas en ce moment en état de les faire.* »

Jean le Rond d'Alembert, Lettre au roi de Prusse, 28 avril 1777

1.1 Introduction

On note une évolution naturelle des tâches de conception vers la conception de systèmes de plus en plus complexes : ils intègrent de plus en plus de fonctions, doivent interagir, répondre à des exigences de fiabilité et de sécurité de plus en plus fortes, être économiques et écologiques tout en restant simples d'utilisation, de prix réduit et rapidement disponibles. Cette évolution passe par une réorganisation des partenariats qui, elle aussi, se complexifie : réseaux de partenaires, co–développement, dispersion des contributions…

Ces problématiques ont progressivement conduit à des réflexions orientées non plus vers le seul « comment faire » mais aussi vers le « comment s'organiser pour le faire ». Si, séparément, chaque contributeur au développement d'un système est hautement spécialisé dans son domaine, aucun en revanche n'a la visibilité suffisante pour la prise en compte de la problématique globale. L'ingénierie système, au service du maître d'œuvre, est apparue comme une

discipline émergeante qui se propose de résoudre les problèmes posés par le développement des systèmes complexes.

Nos travaux reposent en grande partie sur cette nouvelle discipline. Pour développer notre démarche, nous employons nombre de ses concepts, et, pour valider notre démarche, nous avons choisi de l'appliquer à l'un des standards de référence de l'ingénierie système : l'EIA-632.

On verra par la suite, dans les chapitres 2 et 3, comment nous pouvons formaliser les processus recommandés par une norme d'ingénierie système puis les spécialiser à un domaine d'activité et à un projet particulier.

Dans ce premier chapitre, nous présenterons d'abord les principes fondateurs et les avancées de cette nouvelle science de l'ingénieur qu'est l'ingénierie système. Dans la première section, nous la présenterons en tant que discipline puis, dans la seconde section, les outils, modèles et méthodologies qui la constituent seront détaillés. Les deux sections suivantes seront plus spécifiques, elles présenteront respectivement les normes d'ingénierie système qui sont à la base de nos réflexions, et plus spécialement la norme EIA-632 choisie comme le scénario à partir duquel dériver des processus spécifiques. Enfin, dans la dernière section, nous repositionnerons la problématique de la thèse dans ce contexte de l'ingénierie système.

1.2 L'ingénierie système

1.2.1 Définition et objectifs de l'ingénierie système

L'ingénierie Système (IS) est, selon la définition 2, une approche interdisciplinaire dont l'objectif est la conception et la réalisation de systèmes complexes définis selon la définition 3. Elle se compose de deux activités principales [Def01] : la connaissance technique des domaines mis en œuvre dans le système et la gestion de leur mise en œuvre.

Définition 2 : *Ingénierie Système* [Tho93]

L'ingénierie système est une approche et des moyens interdisciplinaires permettant la réalisation et le déploiement de systèmes réussis. Elle peut être vue comme l'application de techniques d'ingénierie à l'ingénierie des systèmes, aussi bien que comme l'application d'une approche systématisée aux efforts d'ingénierie.

Définition 3 : *Système* [SC95]

Un système est un ensemble de composants interdépendants qui interagissent entre eux de façon organisée vers un but commun. Les composants d'un système peuvent être divers et se composer de personnes, d'organismes, de procédures, de logiciels, d'équipements ou d'installations.

Ce qui démarque l'ingénierie système des autres disciplines d'ingénierie est qu'elle ne s'attache pas seulement à la réalisation d'un produit physique. Elle cherche aussi à guider et à coordonner au mieux l'effort de toutes les disciplines concernées par la conception et la réalisation du produit.

La conception et le développement d'un système demandent la contribution de techniques nécessairement diverses. Chacune de ces techniques se concentre sur un aspect particulier du système, sans visibilité sur les autres aspects du système : les concepteurs ont une vision structuro–fonctionnelle du système ; l'équipe d'industrialisation le considère comme un élément à intégrer dans la chaîne d'assemblage ; les achats comme une liste de fournitures… L'ingénierie système doit apporter un point de vue global sur le système pour permettre la mise en cohérence des contributions spécifiques sur le système.

L'ingénierie système considère tout le cycle de vie du système, de la définition des exigences client au retrait de service en passant par la conception, la réalisation et la mise sur le marché. Elle définit des méthodes et des moyens permettant de satisfaire au mieux les besoins techniques et organisationnels qui se heurtent en pratique à des contraintes de coûts, de délais et de productivité.

1.2.2 État des connaissances et des pratiques

Les premiers travaux importants sur les systèmes sont apparus au cours de la seconde guerre mondiale. Ils coïncident avec l'objectif de développement de systèmes militaires de plus en plus complexes. Dans les années 60, les programmes militaires spatiaux américains ont ainsi proposés de nouvelles approches industrielles plus rationnelles. En 1990, le National COuncil on System Engineering (NCOSE) fusionne des corporations et organisations américaines en une société dont l'objectif est de développer les pratiques et les formations en ingénierie système. En 1995, le NCOSE s'ouvre aux participations internationales et devient l'INCOSE (INternational COuncil on Systems Engineering). En France, il faut attendre 1999 pour que l'Association Française d'Ingénierie Système (AFIS) soit créée. Elle compte aujour-

d'hui 24 membres et plus de 500 adhérents individuels[AFI07].

Initialement, les travaux en ingénierie système se sont plutôt centrés autour d'une démarche de conceptualisation. De nombreux auteurs se sont d'abord attachés à identifier les concepts de l'IS [WAHH99] et à décrire le processus de référence autour duquel vont s'articuler les projets. L'essentiel des travaux s'orientait vers la définition d'un processus générique d'ingénierie système [Boa95b, Boa95a, FM99] avec des sous–composantes techniques (gestion du risque et de la complexité, traitement des exigences, *etc.*). Ils ont été à la base de la rédaction de premiers grands recueils sur l'ingénierie système rédigés par des organismes comme la NASA [SC95] ou le DOD [Def01].

Les premiers concepts de l'IS ont été complétés par de nombreux travaux au cours de la dernière décennie [Rom96, Bar97, Pen97, Gir99, Cha99, Lar03, Lon03] et sont maintenant établis et reconnus au niveau international via des normes. On constate maintenant une évolution des travaux de recherche. Ces travaux se tournent vers la mise en application des concepts développés plutôt que sur le développement de nouveaux concepts. Cette mise en application passe principalement par des réflexions autour de l'identification des liens entre produit et processus [Lab04, GE07], de la formation à l'IS [BLWvH05, JD07] et enfin l'outillage de la démarche d'IS [Ped06, TMHS06, SC07]. Bien entendu, un travail de fond demeure et des analyses critiques de l'existant [BY02, BY03], des propositions d'amélioration [BB01, WW03] ou d'évaluation [Hon04, Hon06] paraissent régulièrement dans la littérature spécialisée.

1.2.3 Principes et moyens de l'ingénierie système

L'objectif de l'ingénierie système est de permettre la réalisation et le développement de systèmes complexes. Elle consiste en :

- la recherche et le développement des techniques de l'ingénierie pour la création de systèmes complexes,
- la recherche d'approches construites et cohérentes réalisant la coordination de ces efforts d'ingénierie.

Dans ce contexte, la complexité n'est pas seulement limitée aux techniques et technologies de l'ingénierie de systèmes mais inclut aussi les organisations. Un système peut devenir très complexe par l'augmentation en amont des données, des variables, mais encore par le nombre de domaines simultanément impliqués dans son fonctionnement. Certains systèmes de taille particulièrement grande sont approchés par une voie nouvelle : les systèmes de systèmes [BYAB+04, Jam05b, Kea05].

L'ingénierie système est là pour favoriser le développement de méthodes et d'outils permettant de comprendre et de gérer la complexité des systèmes. Ces outils sont amenés à manipuler un large spectre de domaines scientifiques et techniques qui inclut [Ped06] :

- l'analyse fonctionnelle et l'analyse d'architecture,
- l'analyse des exigences et leur vérification,
- l'analyse du risque et des autres données du projet,
- la simulation et la modélisation du système.

Cette approche interdisciplinaire de l'ingénierie système est intrinsèquement complexe puisque les interactions avec l'environnement et même les interactions entre toutes les composantes du système ne sont pas toujours bien définies ou bien comprises (du moins *a priori*). Définir et caractériser de tels systèmes et sous–systèmes ainsi que leurs interactions sont un des objectifs de l'ingénierie système. En faisant cela, le pont entre exigences informelles des utilisateurs et les spécifications techniques qu'un ingénieur peut implémenter pourra être franchi. L'IS cherche à gérer cette problématique par une approche holistique et interdisciplinaire :

Vue holistique : L'ingénierie système se fonde sur la définition des besoins du client et l'identification des fonctionnalités du système au plus tôt dans le cycle de développement (traitement et documentation des exigences). Elle réalise alors la conception et la validation du système en considérant le problème sur l'ensemble de son cycle de vie. Le modèle des processus d'ingénierie est au cœur de l'ingénierie système. Ce processus d'ingénierie a pour but d'organiser l'effort technique et organisationnel dans tout le cycle de vie. Il couvre la définition du concept et la production et va, dans certains cas, jusqu'à la mise à disposition et au retrait de service du système.

Couverture interdisciplinaire : Le développement de systèmes demande souvent les contributions de diverses disciplines techniques. Pour obtenir une telle expertise, un ingénieur système est souvent un ingénieur traditionnel possédant une expertise dans un domaine et une connaissance des autres domaines incluant la gestion et les processus d'affaire, ce qui aide à l'intégration des sous–systèmes et à la validation des exigences. En donnant une vue système (holistique) à l'effort de développement, l'IS aide à intégrer toutes les contributions techniques dans un effort commun coordonné.

L'industrie commence à accepter l'idée que l'ingénierie d'un système, qu'il soit grand ou petit, peut être non–déterministe : éléments non prévus dans l'environnement du système ou caractéristiques du système non prévues (propriétés émergentes). Les décisions prises en début

de projet ont un impact énorme sur la poursuite de celui–ci et peuvent avoir des conséquences des années, voire des décennies, après avoir été prises.

La comparaison faite par Alfred Spector [SG86] illustre parfaitement l'impact de propriétés non–déterministes dans les projets : Spector compare les développements respectifs d'un pont et d'un logiciel. La construction du pont se fait dans les temps et dans le budget initial alors que le projet de développement logiciel dépasse le budget et les délais initiaux. Il montre que la principale différence, qui rend difficile la réussite du projet logiciel, vient de l'environnement du projet : dans le cas du pont, l'environnement est figé et peu de flexibilité est laissée pour la modification des spécifications alors que dans le cas du logiciel qui correspond à un mode de développement plus récent, l'environnement est soumis à des changements brutaux qui demandent des adaptations rapides...

D'autres études ont cherché à déterminer les causes d'échec des projets dont le fameux rapport CHAOS et ses successeurs [Sta94, Sta99, Sta01]. Même si elles ont été critiquées par la suite [Gla06], elles ont permis de mettre en évidence l'importance de la gestion de projet et notamment du traitement des exigences [JMS02].

De nombreux processus ont été proposés pour organiser l'effort technique sur le cycle de vie du projet. Ces processus peuvent être génériques (cycles en cascade, en V, en spirale), liés à un domaine ou à un outil comme UP ou définis spécifiquement à une entreprise (NASA, Boeing).

Terry Bahill [BG98] a répertorié les similarités entre ces processus et a identifié, à partir de celles–ci, un processus générique regroupant leurs « meilleures » propriétés : le processus SIMILAR illustré figure 1.1. Ce processus se compose de 7 tâches interagissant de manière parallèle et itérative : définir le problème, étudier les alternatives, modéliser le système, intégrer, lancer le système, évaluer la performance et ré–évaluer. Il est important de noter que l'exécution de ces tâches n'est pas séquentielle et qu'aucun ordre n'est établi *a priori*. L'idée générale de ce processus est de réduire les risques pris en systématisant les mécanismes de spécification–validation.

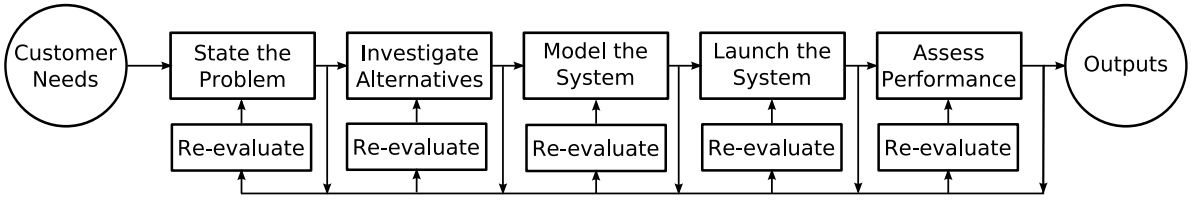


FIG. 1.1: *Le processus SIMILAR selon [BG98].*

L'ingénierie système repose sur cette idée qu'il existe un processus interdisciplinaire qui permet de s'assurer que les besoins du client et des parties intéressées seront satisfaites en termes de qualité, de confiance, de coût et de délai sur l'ensemble du cycle de vie du système. Les processus les plus récents décrits dans des normes comme l'IEEE 1220 et l'EIA-632 sont d'ailleurs issus de ce principe et l'on peut faire les correspondances avec le processus SIMILAR (tableaux 6 et 7 en annexes). Nous reviendrons sur le détail de ces normes dans la section 1.4.

1.3 Outils, modèles et méthodologies en ingénierie système

La démarche d'IS coordonne des actions interdisciplinaires utiles sur l'ensemble du cycle de vie du système. Cette coordination s'appuie sur la définition et l'utilisation du concept de processus et la mise en place de méthodes et d'outils tels que définis par [Est07] dans les définitions 4, 5, 6 et 7 ci–dessous.

Définition 4 : *Processus* [Est07]

Un processus est une séquence logique de tâches réalisées pour atteindre un objectif particulier. Un processus définit ce qui est à faire sans préciser le « comment faire ». La structure d'un processus peut se présenter sous différents niveaux d'agrégation pour permettre les analyses et répondre aux différents besoins d'aide à la décision.

Définition 5 : *Méthode* [Est07]

Une méthode est attachée aux techniques de réalisation d'une tâche. Elle définit le « comment faire » de chaque tâche (dans ce contexte méthode, technique, pratique et procédure sont souvent interchangeables). A tous niveaux, les tâches de processus sont réalisées en utilisant des méthodes. Cependant, chaque méthode est aussi, elle–même, un processus avec sa séquence de tâches à réaliser selon des méthodes particulières. **En d'autres mots, le « comment » d'un niveau d'abstraction devient le « quoi » du niveau inférieur.**

Définition 6 : *Outil* [Est07]

Un outil est attaché à une méthode particulière, pour augmenter l'efficacité de réalisation de la tâche. **Le rôle d'un outil est de faciliter le « comment ».**

Définition 7 : *Méthodologie* [Est07]

Une méthodologie est définie comme une collection de processus, de méthodes et d'outils reliés. Une méthodologie est essentiellement une « recette » et peut être comprise comme l'application de processus, méthodes et outils relatifs à une classe de problèmes qui ont quelque chose en commun.

Les outils, modèles et méthodologies de l'IS seront ici classés en fonction de leur couverture des domaines de l'IS. On distingue :

Les outils, modèles et méthodologies des activités d'IS spécifiques à une tâche donnée dans un processus d'IS,

Les modèles et outils méthodologiques orientés autour de la définition et la formalisation des processus,

Les modèles universels de système qui s'attachent à décrire un système sur l'ensemble de son cycle de vie dans un langage unique,

L'ingénierie système guidée par les modèles qui cherche à intégrer modèles de processus et modèles de systèmes dans une démarche unifiée.

Ces quatre points sont développés ci–après.

1.3.1 Outils, modèles et méthodologies des activités propres à l'ingénierie système

Si l'ingénierie système est une démarche méthodologique qui a pour objectif de maîtriser la conception de systèmes et de produits complexes, son application passe par la mise en œuvre d'une multitude de démarches techniques dédiées à la réalisation des activités d'ingénierie système. Grossièrement, on peut considérer les activités de l'IS comme les tâches élémentaires d'un processus ou d'une méthode.

Ces activités peuvent être :

1. liées à des activités spécifiques de l'IS telles que l'analyse des exigences ou la gestion des risques
2. ou liées à des activités d'ingénierie classique dans un contexte d'IS telles que la conception mécanique ou l'étude thermique de composants.

Quelle que soit son origine, chacune de ces activités est supportée par des outils, des modèles, des langages et/ou des méthodologies spécifiques à son domaine. Dans le cas de langages, on parlera alors de « langages dédiés » ou Domain-Specific Languages (DSL)[1].

L'utilisation d'outils, de modèles et de méthodologies spécifiques est rendue nécessaire par le besoin de disposer d'un moyen adapté et optimisé de traiter chaque problème particulier de la conception et du développement d'un système. Cependant, si chaque outil, modèle ou méthodologie est adapté à une problématique, aucun d'eux ne considère le problème de l'ingénierie du système dans son ensemble. Ils n'ont ni la vision holistique, ni la vision interdisciplinaire de l'IS dont l'enjeu est justement de parvenir à organiser l'utilisation de ces ressources.

1.3.2 Modèles et outils méthodologiques relatifs aux processus

A l'inverse, d'autres modèles et outils s'attachent au suivi et à l'organisation des activités interdisciplinaires sur le cycle de vie du système.

On distingue principalement deux types d'approches : les approches purement méthodologiques et les approches liées à un outil.

1.3.2.1 Approches purement méthodologiques

Ces approches se concentrent exclusivement sur la définition des processus d'IS. Elles font l'hypothèse que les praticiens trouveront eux–même les méthodes et les outils nécessaires à leur application.

Les premiers processus proposés sont le cycle en V puis le modèle en spirale. Ils ont rapidement évolué pour prendre la forme des normes proposées par les organismes d'IS [oEE99, Ele99, AFN03b], la NASA [SC95] ou le DOD [Def01].

Les processus décrits sont essentiellement textuels. On peut cependant noter quelques efforts récents pour les formaliser : citons les travaux de Caple [Cap01, Cap03] qui a cherché

[1] Le terme DSL est ici employé dans un sens très général plus vaste que son acceptation commune. Il inclut tout type de langage utilisé en ingénierie et n'est pas limité aux langages de programmation.

à modéliser des concepts de l'IS. Malheureusement, le langage de description qu'il utilise est très loin des références actuelles : il est exclusivement graphique, n'est pas accompagné d'outils et n'a pas été utilisé en dehors de ses travaux.

1.3.2.2 Approches méthodologiques liées à un outil

Parallèlement à ces approches purement méthodologiques, on voit les éditeurs d'environnements de développement proposer des méthodologies de conception associées à leurs outils. On peut citer les méthodes :

- Harmony-SE de Telelogic,
- OOSEM (Object-Oriented System Engineering Method) de l'INCOSE [LFM00],
- RUP SE (Rational Unified Process for System Engineering) d'IBM,
- MBSEM (Model-Based System Engineering Methodology) de Vitech.

Associer un processus de conception à un outil de conception est une démarche essentiellement commerciale pour les éditeurs. Si les processus proposés peuvent s'avérer intéressants, ils restent la plupart du temps informels, ne sont pas intégrés dans les outils de développement et ne sont pas non plus supportés par des outils associés qui auraient pu être dédiés à la gestion des processus[2]. Ils reviennent donc à utiliser, là encore, des processus décrits de manière textuelle. D'autre part, la représentation du système et son évolution au cours du processus de conception sont limitées par l'utilisation d'un langage unique tout au long de son ingénierie.

1.3.3 Modèles universels du système

Tout au long de son développement, la représentation du système passe par différents stades : expression des exigences, solution logique puis physique, *etc.*Un des enjeux de l'IS est de s'assurer de la cohérence et de l'interopérabilité de ces représentations au cours de l'avancement du projet. Dans cette optique, on a vu apparaître de nouveaux langages de description de systèmes comme SysML ou UDPM qui cherchent à unifier les représentations du système dans un formalisme unique tout au long de son cycle de vie.

Ces langages reposent sur une décomposition du modèle en « vues ». Chaque vue est dédiée à la représentation d'un aspect particulier du système : exigences, structures, comportements... et représentée par des diagrammes spécialisés dans la représentation de ces aspects. Les développeurs ne manipulent plus une multitude de modèles rédigés dans des formalismes

[2]Harmony-SE et OOSEM ne sont pas accompagnés d'outils. RUP SE et MBSEM possèdent des outils support liés aux activités demandées (outil d'analyse du risque, gestion des exigences, *etc.*) et non pas liés directement à la formalisation et la gestion du processus proposé.

disjoints mais un seul modèle dans lequel l'utilisation d'un langage unique permet d'établir des ponts entre les vues. A partir de là, ils peuvent assurer une cohérence entre les vues du modèle ou faire un suivi des éléments du modèle en associant, par exemple, exigences et fonctions.

1.3.3.1 Le langage SysML

SysML (System Modeling Language) est né de la collaboration entre l'INCOSE et l'OMG. Après plusieurs versions, plutôt orientées autour de la modélisation des concepts d'IS [Fri03, JB02, Oli03], la version finale de SysML a été adoptée en mai 2006 [OMG06b].

SysML est un langage de modélisation spécifique à l'ingénierie système. Il supporte la spécification, l'analyse, la conception et la validation et vérification d'une large gamme de systèmes.

Dérivé d'UML qui est centré sur la conception de logiciels, SysML est orienté vers les systèmes. Il propose :

- une sémantique plus flexible et plus expressive,
- un langage « réduit » plus facile à apprendre et à appliquer,
- des tables d'allocations propres aux concepts de l'IS (allocation d'exigences, de fonctions, *etc.*) et
- une représentation orientée modèles, vues et points de vue qui permet une décomposition selon les intérêts.

1.3.3.2 Le langage UPDM

Le DoDAF (Department of Defense Architecture Framework) [DOD07a, DOD07b, DOD07c] est un environnement de développement d'architectures de systèmes ou d'entreprises défini par le département de la défense des États-Unis (DOD). Comme ses dérivés tels que le MODAF [Moda, Modb] ou le NAT, son objectif est de s'assurer d'une approche consistante dans le développement d'une architecture d'entreprise, de système ou de système de systèmes. Il définit un ensemble de concepts clefs pour décrire le contexte opérationnel.

Le profil UML pour le DoDAF/MoDAF, UPDM, permet une représentation standardisée des vues du DoDAF. Ce profil permet d'utiliser un langage commun d'expression de l'architecture pour améliorer la qualité de l'architecture ou du modèle de système de systèmes, améliorer l'inter–opérabilité entre outils et les communications entre parties intéressées et réduire les besoins de re–formation sur différents outils ou projets.

1.3.4 Ingénierie système guidée par les modèles

1.3.4.1 Principes de l'ingénierie système guidée par les modèles

L'ingénierie système guidée par les modèles (définition 8) est un concept introduit par Loyd Baker en 2000 [BCC+00] qui définit le Model Driven System Design (MDSD). Cette approche se différencie d'une approche classique des processus d'IS centrés sur les documents en mettant les modèles au cœur de l'ingénierie. En modélisant les multiples aspects du système au cours de son cycle de vie, les auteurs attendent de nombreux gains de productivité et de qualité du produit résumés sur le tableau 1.1. En d'autres termes, ils proposent d'étendre les principes de l'IDM[3] du domaine logiciel au domaine des systèmes.

Définition 8 : *Ingénierie Système guidée par les Modèles*

L'Ingénierie système guidée par les modèles est une démarche qui propose l'extension des principes de l'IDM à l'ingénierie des systèmes. Il s'agit de considérer l'ensemble des parties du systèmes comme des modèles et non plus uniquement ses composants logiciels. Dans ce contexte, les opérations de transformation de modèles sont guidées par les processus de l'ingénierie système.

Cette approche se distingue de celles présentées ci–dessus (activités propres de l'IS, approches centrées processus ou approches centrées système) pour deux raisons principales :

1. elle demande de **disposer d'un processus d'ingénierie du système défini et formalisé, *i.e.* de disposer d'un réel modèle du processus** ;
2. la représentation du système ne se fait pas par un langage unique de type SysML mais par une multitude de DSL. A chaque étape de son cycle de vie, et selon les besoins, plusieurs modèles du système écrits dans différents langages co–existent et évoluent.

Toute la problématique de la méthode est alors centrée sur l'organisation des modèles dans le cycle de vie du système.

[3] Voir le chapitre 2.2 de ce document qui reviendra en détails sur l'ingénierie des modèles.

[4] Il est nécessaire de nuancer ici l'optimisme annoncé par Baker [BCC+00]. Si de nombreux avantages sont attendus de la conception système guidée par les modèles, elle reste encore pour une large part du domaine de la recherche. Des revues de projet par interrogation de modèles automatisées, une vérification automatisée ou une traçabilité intégrale sont à considérés comme des avantages attendus mais pas encore opérationnels.

		Conception système dirigée par les modèles	**Conception système centrée documents**
Fonctions	*Référentiel des informations*	Modèles	Documents
	Revues	Par interrogation de modèles (automatisée)[4]	Lecture et interprétation du texte puis comparaison
	Vérification	Implicite, incrémentale, automatisée[4], construite sur le processus	Processus d'audit humain
	Communication	Reproductible et consistante	Les réponses peuvent dépendre des perspectives des lecteurs
	Validation	Exécutée dans différents contextes (*e.g.* contexte client, en ligne)	Révisions, revues de papiers
	Traçabilité	Intégrale[4]	Au prix d'un effort intensif
	Réutilisation	Librairies, « plug and play »	paragraphe standard uniquement
	Adoption culturelle	Nouveau paradigme	Status Quo
Infrastructures	*Poste de travail* & ordinateurs	Ressources logicielles additionnelles	Inférieures à celles de l'approche dirigée modèle
	Outils	Peu disponibles	Largement disponibles
	Processus	Immatures	Des processus existent mais varient selon les compagnies
	Formation	Immature	Disponible
	Navigation	Potentiellement facile puisque dépendante des données	Parcours facile d'un document individuel mais pas de conception rationnelle, corrélation entre documents difficile

TAB. 1.1: *Comparaison des approches de conception système basées sur documents et basées modèles d'après [BCC+00]. D'une manière globale, une conception système dirigée par les modèles, i.e. faite selon les recommandations de l'ingénierie système guidée par les modèles, simplifie la conception en automatisant un grand nombre de tâches et en assurant une reproductibilité des opérations. En revanche, ce type de conception manque de maturité et demanderait, pour être opérationnel, que soient développés des outils et des méthodologies dédiés.*

Actuellement cette méthodologie reste une proposition. L'IS guidée sur les modèles n'est pas encore opérationnelle et, même en se limitant au domaine du logiciel, de nombreux problèmes restent à résoudre avant d'arriver à un stade opérationnel.

A noter que, si l'IS s'inspire des techniques de l'IDM, on trouve aussi des réflexions en IDM inspirées par l'IS. L'IDM a récemment pris conscience de la nécessité de maîtriser les processus d'ingénierie et de nombreux projets apparaissent autour de cette thématique.

On peut citer :

- le projet SPEEDS (SPEculative and Exploratory Design in Systems engineering) [EMW06] qui va définir une nouvelle génération de méthodologies, processus et outils supports pour la conception de systèmes embarqués critiques,
- le projet DOMINO (DOMaINes et prOcessus méthodologique) [IRI07] qui propose une démarche basée sur la description d'un système par divers modèles exprimés dans des langages de modélisation dédiés différents,
- le projet Modelplex [IST07] qui vise à développer une solution basée sur l'IDM pour la modélisation de système d'ingénierie complexe.

1.3.4.2 Projets connexes à l'ingénierie système guidée par les modèles

Indépendamment de l'initiative de Baker, on a vu apparaitre ces dernières années des projets proches de la philosophie de l'ingénierie système guidée par les modèles. Ces projets sont portés par de grands groupes industriels principalement aéronautiques et automobiles qui tentent de mettre en œuvre des outils permettant la manipulation de modèles du système. Ces outils sont souvent centrés sur le métier de leur concepteur (systèmes embarqués pour l'automobile, pour l'avionique, *etc.*) et pêchent souvent par l'absence d'un processus de développement associé. Ils démontrent cependant que la manipulation de modèles peut sortir du domaine du logiciel pour être appliquée dans le contexte plus général des systèmes : ils sont un premier pas vers une ingénierie système guidée par les modèles.

On recense actuellement les projets suivants [APFPB+06] :

EAST-EEA (Embedded Electronic Architecture) [ITE] est un projet regroupant des équipementiers, des constructeurs automobiles et des académiques. Il définit les spécifications des services d'un intergiciel (middelware) assurant l'interopérabilité et la portabilité de composants distribués. Il définit le langage d'architecture EAST-ADL comme support d'un processus de développement [HH04, HKK04, DSLT04, VdBH+04].

AUTOSAR (AUTOmotive Open System ARchitecture) est un projet qui regroupe les constructeurs automobiles, les équipementiers et leurs fournisseurs. Il établit un standard ouvert pour l'architecture électrique/électronique automobile. Il inclut la standardisation des fonctions système de base et des interfaces fonctionnelles, la capacité d'intégrer et de transférer des fonctions, et l'amélioration substantielle des mises à jour et des mises à niveau logicielles durant la vie des véhicules.

OpenDevFactory [SYS] est un projet du pôle de compétitivité System@tic. Son objectif est de fournir une plate–forme d'intégration normalisée des développements technologiques portant sur les outils logiciels de modélisation. Ce projet produira les éléments de base sur lesquels les outils domaines : automobile, sécurité, télécommunication, pourront être dérivés à moindre effort. Construit sous la forme d'outils interopérants il permet de ne déployer que des parties limitées pour constituer des ateliers de développement utiles au contexte et besoins des différents utilisateurs.

OpenEmbeDD [Rés] est un projet dont l'objectif est de développer une plate–forme libre (open source) pour mettre l'ingénierie dirigée par les modèles au service des applications temps–réels embarquées.

TOPCASED (Toolkit in Open-source for Critical Application & SystEms Development) [CNR] propose la construction d'un atelier de développement libre s'appuyant, d'une part sur les technologies de l'IDM dans le cadre de la plate–forme Eclipse et sur des approches de vérifications formelles et, d'autre part sur **la définition d'un processus commun basé sur les modèles dérivés de l'EIA 632** mettant en œuvre des transformations de modèles.

Parallèlement à ces travaux, on note aussi des réflexions académiques qui cherchent une meilleure intégration des produits et des processus [oT06, oT07]. Ces travaux, réalisés avec moins de moyens que les précédents, sont souvent informels et rarement accompagnés d'outils ou, quand ils le sont, se limitent à un domaine très spécifique comme par exemple le temps–réel embarqué [SV07].

1.3.5 Récapitulatif des outils, modèles et méthodologies de l'ingénierie système

Le tableau 1.2 résume la classification que nous avons faite des outils, modèles et méthodologies de l'IS. Ces derniers sont classés selon le type de processus qu'ils utilisent et le nombre de langages de représentation du système. Aux deux extrêmes on trouve les approches

méthodologiques, centrées sur les processus, et les outils spécifiques des activités d'IS, centrés sur une activité et qui ne prennent pas en compte le processus global. Les modèles universels de systèmes sont plus ambigus. Ils ne sont pas eux–mêmes accompagnés d'un processus mais laissent libre l'utilisateur d'en définir un. Ils reposent d'ailleurs sur l'idée sous–jacente qu'un processus les complète. C'est pour cette raison que les éditeurs ont développé des méthodologies liées à leurs outils. Malheureusement celles–ci sont souvent trop spécifiques et pas assez formalisées pour être pleinement exploitables. L'idéal semble être l'ingénierie système

		Système		
		Pas de modèle du système	*Langage de modélisation unique*[5]	*Langages de modélisation multiples*
Processus	*Indéfinis*		Modèles universels de système [1.3.3]	Outils des activités d'IS [1.3.1]
	Informels	Approches méthodologiques [1.3.2.1]	Approches méthodologiques liées à un outil [1.3.2.2]	IS guidée par les modèles (état actuel) [1.3.4]
	Spécifiés			IS guidée par les modèles (objectif)

TAB. 1.2: *Classification des outils modèles et méthodologies d'IS.*

guidée par les modèles, qui combine à la fois une réflexion méthodologique profonde sur les processus à adopter et la possibilité, par l'utilisation de langages de modélisation multiples du système, d'utiliser à chaque étape de la conception des modèles réellement spécialisés tout en permettant l'évolution de ces modèles au cours du cycle de vie du système.

Malheureusement, pour être pleinement opérationnelle, cette méthodologie demande :

– de formaliser les processus de gestion de projets qui vont définir les étapes de l'ingénierie du système,
– de savoir transformer les modèles de systèmes sur la base des processus formalisés.

Deux conditions qui n'avaient jusqu'alors pas été remplies. Dans ce travail de thèse, nous allons voir et réaliser la première d'entre elles et nous verrons, dans le chapitre 4.4, comment la méthodologie proposée dans cette thèse peut s'adapter à une ingénierie système guidée par les modèles.

[5]Langage unique ne signifie pas modèle unique. Plusieurs modèles peuvent se succéder dans le même formalisme aux différentes étapes du cycle de vie du système. C'est le cas pour des langages très généraux comme UML, SysML…

1.4 Les normes d'ingénierie système

Les normes (définition 9) de l'ingénierie système sont sans doute la partie la plus aboutie de cette discipline. Elles concentrent l'essentiel des avancées méthodologiques dans des référentiels de plus en plus utilisés dans les entreprises. En guidant les praticiens, elles démontrent que l'application de processus de référence structurés conduit à de multiples et importantes améliorations dans la conduite d'un projet : performances, compétitivité, amélioration de la qualité, respects des contraintes de coûts et de délais...

Définition 9 : *Norme* [ISO04]

Document, établi par consensus et approuvé par un organisme reconnu, qui fournit, pour des usages communs et répétés, des règles, des lignes directrices ou des caractéristiques, pour des activités ou leurs résultats, garantissant un niveau d'ordre optimal dans un contexte donné.

1.4.1 Rôle des normes d'ingénierie système

1.4.1.1 Objectifs de la normalisation

L'objectif de l'ingénierie système, c'est d'abord de maîtriser les processus d'organisation, afin de pouvoir garantir la qualité d'un produit ou d'un service, et donc la satisfaction d'un client. Dans ce cadre, les normes d'ingénierie système définissent des processus sur lesquels peuvent se baser les développeurs dans l'ingénierie ou la ré–ingénierie d'un système. Elles servent de référence pour gérer le système de son concept initial à sa mise à disposition en passant par son développement et sa réalisation. Il ne s'agit pas, pour elles, de définir une des activités d'un service de l'entreprise ou les responsabilités d'une personne mais de coordonner l'ensemble des activités d'ingénierie de manière à atteindre un but commun.

Ces normes reposent sur l'idée qu'il existe des concepts communs à tous les projets, quel que soit le domaine d'activité ou le système à développer [BG98]. En identifiant les bonnes pratiques et en assurant la cohérence des activités d'ingénierie, les normes favorisent [Gro97] :

- l'adéquation aux besoins et la qualité des produits,
- l'anticipation des problèmes et la maîtrise des risques concernant tant le projet que le système et son environnement, tout au long du cycle de vie,
- la maîtrise de la complexité des grands systèmes et des produits complexes,

- la tenue des délais et des temps de développement,
- la maîtrise des coûts, avec notamment une anticipation très en amont du coût global du cycle de vie,
- l'efficacité dans la maîtrise de la coopération de la transdisciplinarité et des multiples acteurs,
- la satisfaction de toutes les parties prenantes,
- une meilleure optimisation du compromis global.

1.4.1.2 Cadre d'application de la normalisation

Les processus décrits dans les normes peuvent s'appliquer sur l'ensemble du cycle de vie des systèmes en incluant la conception, le développement, la production, l'utilisation, le support et le retrait de service. Ils peuvent être appliqués de manière concurrente, itérative ou récursive à un système et à ses composants.

Les systèmes considérés peuvent être de petite ou de grande taille, simple ou complexe, unique ou de grande série, être logiciel, matériel, des services ou une composition de ces derniers. On ne peut identifier de système type tant ils peuvent varier selon les domaines et les applications considérées.

Les recommandations des standards s'appliquent aux organisations lorsqu'elles agissent dans le rôle de fournisseur aussi bien que dans celui d'acquéreur. Elles concernent des entités d'une organisation unique ou d'organisations différentes et peuvent aller d'un accord informel à un contrat.

1.4.1.3 Limitations d'une normalisation

Le caractère volontairement générique des normes d'ingénierie système impose les limitations suivantes :

1. Les normes ne détaillent pas les processus du cycle de vie du systèmes en termes de méthodes et de procédures.
2. Le format des données n'est pas précisé.
3. Dans le cas où l'on souhaite utiliser plusieurs normes simultanément, les conflits éventuels entre recommandations doivent être résolus.

Les processus de cycle de vie décrits par les normes d'ingénierie système sont génériques. Les méthodes et procédures qui permettraient de respecter les recommandations des normes

ne sont pas données car elles seront dépendantes soit du cadre de travail de l'entreprise soit du contexte lié au projet. En d'autre termes, **les processus décrits dans les standards doivent être complétés d'une part par les politiques et procédures de l'entreprise et d'autre part par des outils qui vont permettre de réaliser les projets.**

Ainsi, il existe une répartition des responsabilités entre :

l'industrie : qui établit les normes telles que l'EIA-632 ;

l'entreprise : qui précise les politiques et procédures *i.e.* les méthodes ;

le projet : qui fixe la réponse aux exigences des processus retenus *i.e.* qui met en place les outils.

La figure 1.2 illustre ce besoin de compléter les processus des normes d'ingénierie système ainsi que les responsabilités de chacun des acteurs.

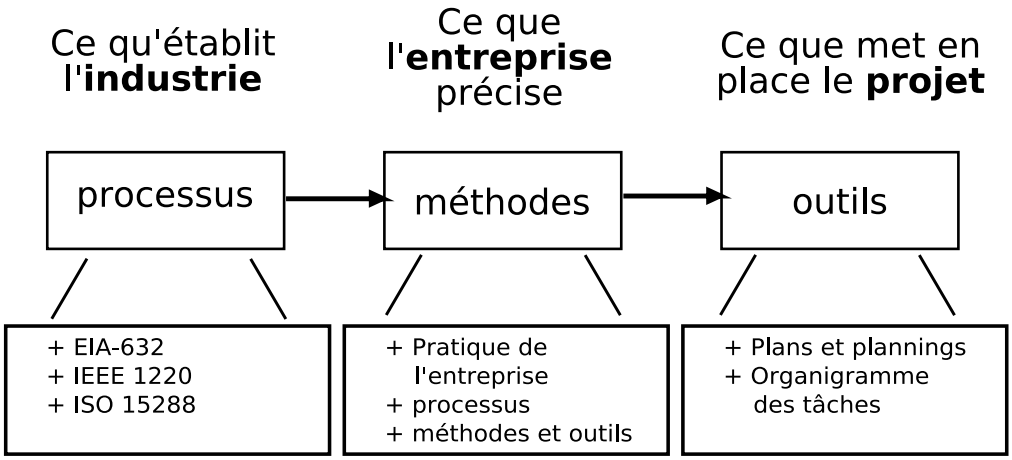


FIG. 1.2: *Processus, méthodes et outils en ingénierie système. Les grandes normes d'ingénierie système décrivent des processus généraux dans le cadre duquel se déroulent les projets. Ces processus sont complétés par les méthodes propres à l'entreprise qui dépendent de son domaine d'activité, de ses pratiques... La mise en place effective des processus et des méthodes passe par l'utilisation d'outils spécifiques utilisés dans le cadre d'un projet précis.*

Les normes d'IS ne spécifient donc pas de méthodes ou d'outils, laissés à la discrétion de l'entreprise. Le rôle de ces normes se limite à :

coordonner les activités des processus : un standard est comme une partition de musique. Il décrit les grandes lignes des processus mais doit être interprété pour former un ensemble harmonieux.

établir la vision partagée de la solution système : la cohérence des exigences et de l'architecture est maintenue et l'effort technique guidé.

De la même façon, les noms, les formats, les contenus, les structures ou les médiums des documents associés aux processus décrits vont dépendre du contexte d'application et sont laissés libres.

Il peut arriver que des normes entrent en conflit soit avec une autre politique ou procédure d'organisation soit avec une autre norme, loi ou réglementation. Dans ce cas, il convient d'identifier et de traiter ces conflits en choisissant l'un des référentiels comme support de sa démarche d'ingénierie système. Les questions de la résolution de ces conflits et de la conformité d'un processus ainsi modifié ne sont pas traitées par les normes. Ces questions sont à considérer au cas par cas et demandent aux entreprises de prendre leurs responsabilités quant aux conséquences éventuelles pouvant résulter de leur décision.

1.4.2 Les principales normes de l'ingénierie système

Le travail des associations d'ingénierie système (INCOSE, AFIS) mené en collaboration avec de grands organismes de normalisation comme l'AFNOR, l'ANSI ou l'IEEE a conduit à la rédaction des grandes normes d'ingénierie système.

Actuellement, trois grandes normes font référence :

– l'IEEE 1220,
– l'EIA–632,
– l'ISO 12588.

Le tableau 1.3 résume les principales informations concernant ces normes.

Ces standards décrivent tous le déroulement des projets en termes de processus en dressant une liste d'activités nécessaires au bon déroulement du projet. Le résultat attendu de ces activités ainsi que des exigences sur leur mise en application complètent ces descriptions. Ces normes définissent, chacune, un processus d'entreprise de référence (définition 10).

Définition 10 : *Procédure d'entreprise (ou processus d'entreprise)* [Jou07]

Une procédure d'entreprise est une procédure qui systématise l'organisation et la politique d'une entreprise dans le but d'atteindre certains des objectifs de cette entreprise.

Norme	Nom	Organisme	Ref.
IEEE 1220	Standard for application and Management of the Systems Engineering Process	Institute of Electrical and Electronics Engineers (IEEE)	[oEE99]
EIA–632	Processes for Engineering a System	Electronic Industries Alliance (EIA) et American National Standards Institute (ANSI)	[Ele99]
ISO 15288	System Engineering – System Life-Cycle Processes	association française de normalisation (AFNOR)	AFNOR Z 67-288 [AFN03b]

TAB. 1.3: *Tableau récapitulatif des principales normes d'ingénierie système.*

Ce choix d'une approche par processus se base sur l'idée que les types d'activité à réaliser sont invariants par rapport aux projets et secteurs d'application. Il est donc naturel, pour les organismes de normalisation, de définir des processus plutôt que des cycles de vie. Cette approche a pour avantage de donner une cohérence aux projets inter–entreprises et de laisser les projets définir l'application des processus dans le cadre des cycles de vie qu'ils choisissent.

On verra que les trois principales normes d'ingénierie système diffèrent principalement sur les types de processus qu'elles décrivent : en découle une couverture des activités techniques sur le cycle de vie du système différente pour chacune d'entre elles.

1.4.2.1 La norme IEEE 1220

L'IEEE Std 1220-2005, *Standard for application and Management of the Systems Engineering Process* définit les tâches interdisciplinaires requises pour transformer, au long du cycle de vie du système, les besoins des parties intéressées, exigences et contraintes en une solution système [Dor06]. Ce standard est resté relativement stable depuis sa première version de 1995.

Cette norme se décompose en huit processus représentés figure 1.3 : les processus d'analyse des exigences, de validation des exigences, d'analyse fonctionnelle, de vérification fonctionnelle, de synthèse et de vérification physique s'adressent à la transformation des exigences

en une solution de conception. Le processus d'analyse du système est un support à la résolution de conflit entre exigences, décomposition fonctionnelle, allocation de performances, sélection de solutions de conception, appréciation de l'efficacité du système et gestion des risques. Le processus de contrôle concerne la gestion et la documentation des activités d'ingénierie système.

La norme IEEE 1220 se focalise sur les processus techniques d'ingénierie système allant de l'analyse des exigences jusqu'à la définition physique du système. La portée de cette norme dans le cycle de vie du système est limitée à la phase de définition du système.

1.4.2.2 La norme EIA-632

En 1995, l'EIA lance un groupe d'étude pour étendre le *interim standard EIA/IS 632*. En 1998, la norme EIA/IS–632 *Processes for Engineering a System* est diffusée. Elle étend le champ de la norme précédente à tous les processus techniques d'ingénierie d'un système [Mar98a].

Les processus de conception technique sont associés aux processus d'acquisition et de fourniture. Processus techniques et processus contractuels sont pris en compte. Le cycle de vie du système couvert explicitement par la norme, s'étend de la définition du système à son transfert vers l'exploitation.

La section 1.5 est consacrée entièrement à cette norme. Elle détaillera les processus utilisés par l'EIA-632 ainsi que les concepts clefs définis par cette norme.

1.4.2.3 La norme ISO15288

L'ISO/IEC 15288 – *System Life Cycle Processes* [AFN03b] est le premier standard ISO à traiter des processus de cycle de vie des systèmes. Paru en octobre 2002, plus tardif que les deux précédents, ce standard s'adresse à l'ensemble du cycle de vie d'un système industriel. Il inclut les processus d'exploitation, de maintien en condition opérationnelle et de retrait de service (figure 1.4).

La norme ISO 15288 s'applique à l'ingénierie des systèmes contributeurs qui ont leur propre cycle de vie (systèmes de fabrication, de déploiement, de soutien logistique, de retrait de service). Elle complète les processus s'appliquant aux projets par des processus d'entreprise qui ont pour objectif de développer le potentiel de l'organisme d'ingénierie système en gérant les domaines communs aux différents projets d'ingénierie système. En ce sens, elle

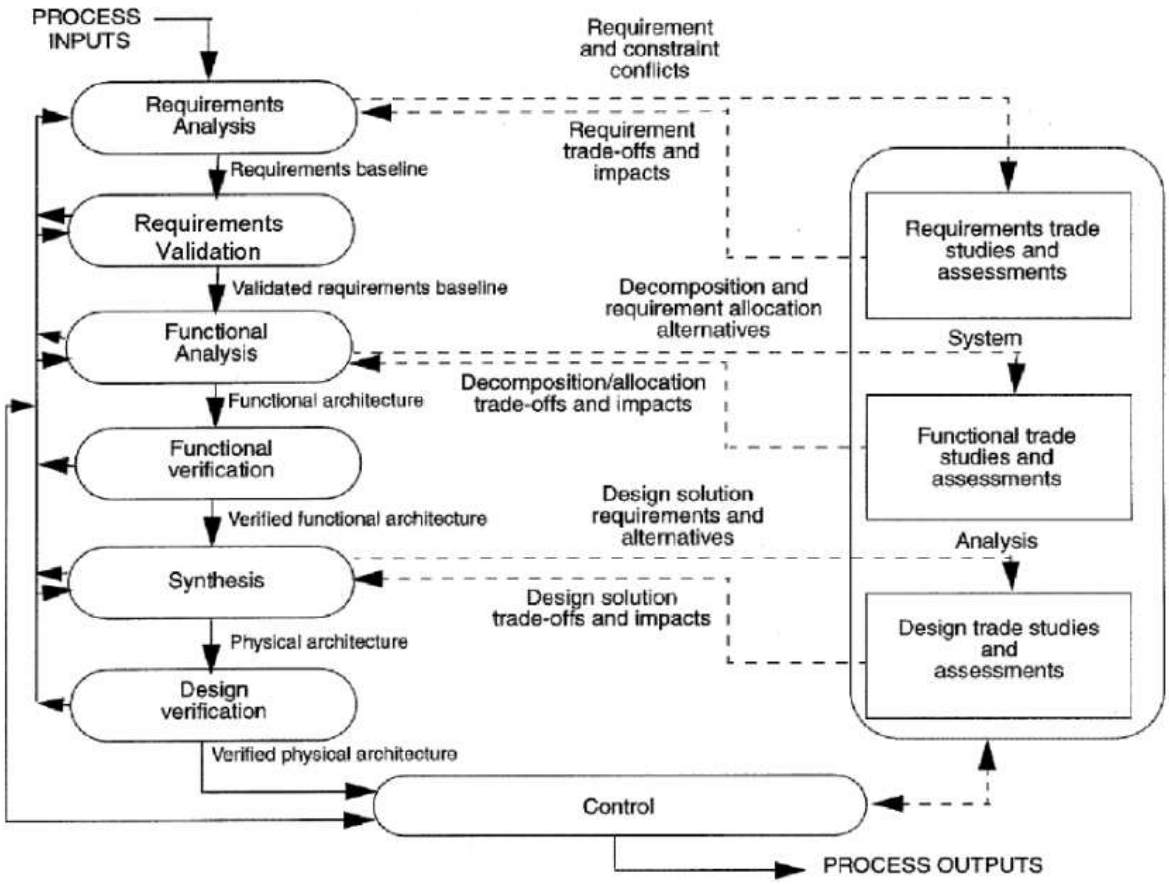


FIG. 1.3: *Processus de l'IEEE 1220 [oEE99]. Les trois processus : analyse des exigences, analyse fonctionnelle et allocation, et synthèse comprennent chacun leur sous-processus de vérification ou de validation. Le processus d'analyse système a pour but d'analyser les problèmes issus des processus principaux comme les conflits d'exigences ou les solutions alternatives dans un cadre pluridisciplinaire. Le processus de maîtrise (control) concerne tout particulièrement la gestion technique de l'ingénierie système et la maîtrise de l'information tant du système que du projet.*

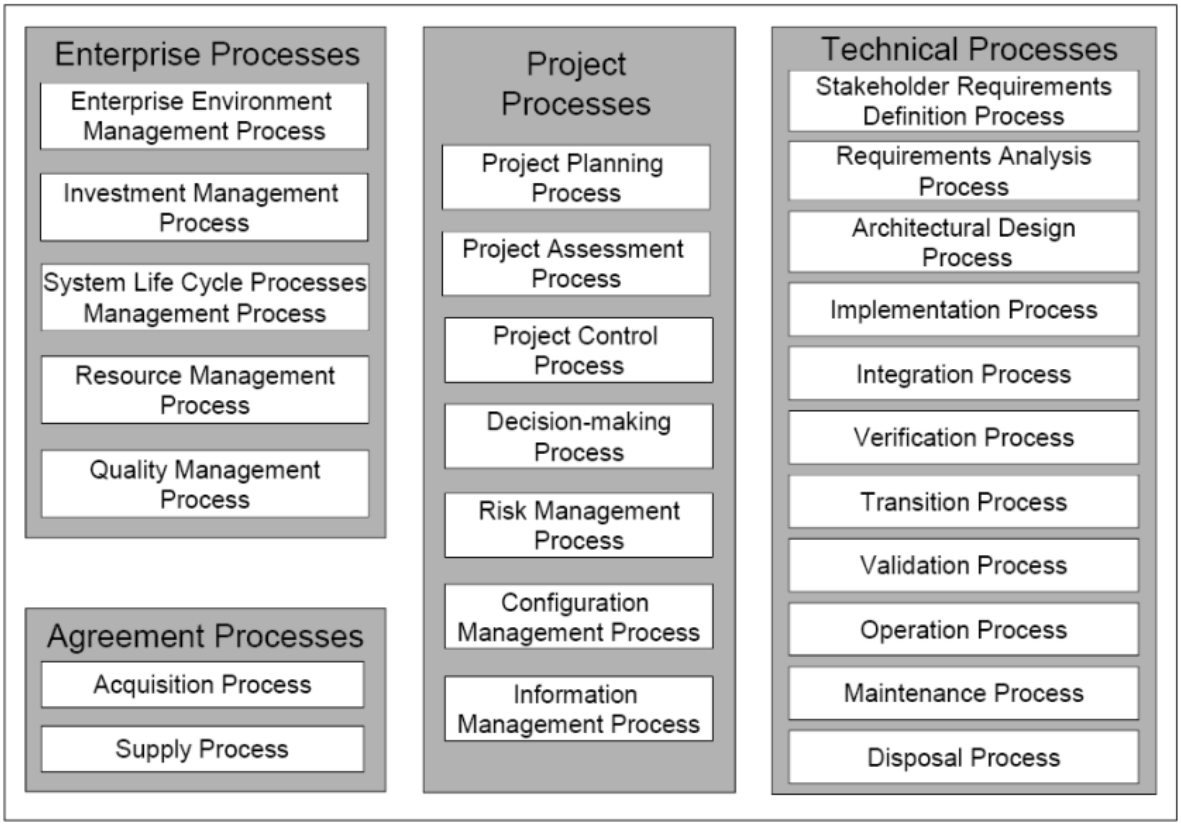


FIG. 1.4: *Processus de l'ISO 15288 [AFN03b]. Tous les types de processus sont couverts par la norme : processus techniques, de management, contractuels et processus d'entreprise. Les processus d'entreprise considèrent le développement et l'amélioration des processus d'ingénierie système au même titre que les processus d'ingénierie du système.*

est complémentaire aux deux précédentes normes et peut s'utiliser de manière conjointe avec l'une ou l'autre de ces normes.

L'ensemble des types de processus est considéré :

les processus techniques qui participent à la transformation des besoins en solution,

les processus de management qui participent à la maîtrise du projet,

les processus contractuels qui assurent les relations avec le ou les clients et les sous–traitants,

les processus d'entreprise qui ont pour rôle de développer le potentiel en IS de l'entreprise en gérant les domaines communs aux différents projets d'IS.

La totalité du cycle de vie du système est couverte par la norme.

La norme SO/IEC 19760 *System Engineering – A Guide for the Application of ISO/IEC 15288 System Life Cycle Processes* [Bri04] vient compléter l'ISO 15288. Ce document fournit des guides pour l'application de l'ISO 15288.

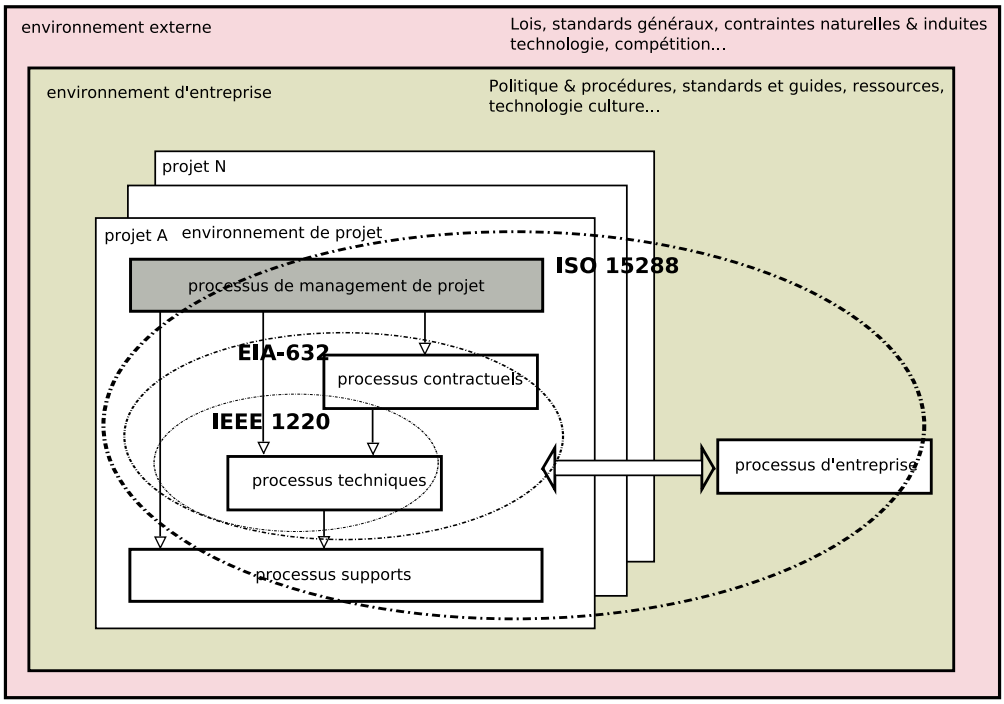
1.4.3 Analyse des similitudes, différences et complémentarité normatives

Au travers des descriptions ci–dessus, on voit que les normes d'ingénierie système diffèrent essentiellement par le type des processus qu'elles couvrent (figure 1.5(a)) : la couverture du cycle de vie du système, voir figure 1.5(b), est donc plus ou moins étendue selon la norme considérée.

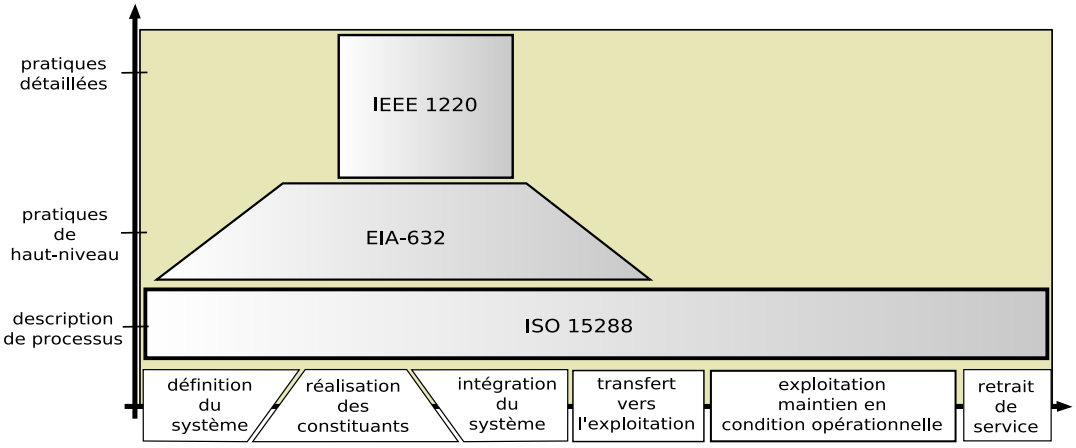
Il faut préciser que plus la norme est étendue, moins la description des processus qui la compose est précise. Si l'IEEE 1220 semble peu intéressante du point de vue de sa couverture du cycle de vie du système, sa description des opérations de définition du système est en revanche inégalée.

De la même manière, l'EIA-632 hiérarchise les processus techniques (qui participent à la transformation des besoins en solution) et leur processus support (qui garantissent la convergence vers une solution vérifiée et optimisée) alors que l'ISO 15288 ne le fait pas.

Selon ses besoins, un industriel peut choisir de mettre en place des procédures en accord avec les recommandations de l'un ou l'autre de ces standards. Leur complémentarité permet de prendre en considération plusieurs de ces standards simultanément. Dans cette éventualité, le couplage de l'IEEE 1220 avec l'ISO 15288 est facilité car les informations nécessaires



(a) Couverture des types de processus par les normes d'IS [Gro97]



(b) Couverture du cycle de vie système par les normes d'IS (d'après [Est07])

FIG. 1.5: *Couverture des normes d'IS. La couverture des normes d'ingénierie système diffère selon les types de processus couverts (figure 1.5(a) et selon les phases du cycle de vie du système (figure 1.5(b)).*

Concernant les processus, l'IEEE 1220 se concentre uniquement sur les processus techniques et l'EIA-632 sur les processus techniques et contractuels. L'ISO 15288 englobe non seulement tout type de processus de projet mais aussi les processus d'entreprise.

Les normes ont alors des couvertures du cycle de vie du système différentes : limitées à la définition du système pour l'IEEE 1220, jusqu'au transfert vers l'exploitation pour l'EIA-632 et ensemble du cycle de vie pour l'ISO 15288.

au couplage de ces deux normes sont directement inclues dans les normes : l'analyse des différences conceptuelles et des modifications nécessaires au couplage sont données.

Le couplage des normes est rendu possible car elles reposent sur quelques concepts de base communs, comme par exemple :

- des processus génériques peuvent être définis sur l'ensemble du cycle de vie du système,
- il existe d'autres clients que ceux qui achètent des produits ou des service dont les exigences doivent être prises en compte : les **parties intéressées**[6] ou parties prenantes,
- un système est une entité décomposable dont on peut identifier une structure générique…

La mise en relation des normes d'ingénierie système n'est malgré tout pas évidente : le couplage des normes est rendu difficile par les différences sémantiques des concepts qu'elles manipulent. Si certains termes présents dans les trois normes semblent, à première vue, désigner les même choses, de légères différences subsistent qui rendent délicate une fusion directe des normes.

Par exemple, dans le cadre de l'IEEE 1220, la structure du système est bornée : elle considère, du plus gros au plus petit, les notions de produits, sous–système, assemblage, composant, sous–assemblage et sous–composant de sorte que la décomposition du système ne peut être plus fine. Dans le cadre de l'EIA-632, en revanche, chaque système est composé de sous–systèmes pouvant eux–mêmes inclure leurs propres sous–systèmes et ainsi de suite sans limite pré–établie dans la décomposition du système initial.

1.4.4 Traitement des besoins complémentaires

Selon les besoins de l'entreprise, ou du projet, les ingénieurs système peuvent décider d'introduire des compléments aux trois normes principales. Il peut s'agir d'éléments :

- spécifiques à un domaine d'activité,
- d'analyse et de conception,
- de gestion de programmes et de projets,
- de support et de soutien logistique ou
- de modèles de maturité…

Les recommandations contenues dans ces éléments complètent, en général, celles des grandes grandes normes d'IS. Elles peuvent se substituer à celles–ci dans des contextes particuliers souvent liés au domaine d'activité de l'entreprise. Nous recensons ci–dessous des exemples de « normes complémentaires » :

[6] Voir glossaire page 207.

Normes spécifique à un domaine Similaires aux grandes normes d'ingénierie système, elles définissent des processus génériques sur le cycle de vie du système. Leur cadre d'application se limite à un domaine d'activité restreint. On peut citer l'ECSS-E-10A et l'ISO TC20/SC14 dans le domaine du spatial ou la DOD-STD-2167A, le MIL-STD-498A et l'AFNOR Z 67-150 pour le logiciel.

Normes d'analyse et de conception Certains aspects spécifiques de l'activité de conception sont couverts par des normes dédiées. L'analyse fonctionnelle et analyse de la valeur par les normes NFX 50-150, NFX 50-150, NFX 50-150 et NFX 50-150, l'architecture des systèmes par la norme IEEE P1471 et le développement logiciel par le GAMT 17.

Normes de gestion de programmes et de projets Le management des programmes ou des projets peut être soumis soit à des recommandations générales (ISO 10006, EMBOK [SRB+06] ou PMBOK [PMI96, Ins03]) soit aux recommandations d'un domaine d'activité précis ; DGA AQ 902 pour l'armement, RG Aéro 00040 et BNAE RG AERO 77 pour l'aéronautique ou SWEBOK pour le génie logiciel.

Normes de support et de soutien logistique La gestion de configuration (normes MIL-STD-973, EIA 649 et ISO 1007), le soutien logistique (normes MIL-STD 1388-1A/2A, MIL-PRF-495006 et MIL-HDBK 502) et l'échange de données d'IS et de données logistiques (normes ISO 10303-AP233, ISO 10303-AP139, AECMA 1000D et AECMA 2000M) possèdent eux aussi des normes dédiées.

Modèles de maturité Les modèles de maturité viennent compléter la panoplie des normes d'ingénierie système. Contrairement aux normes précédentes, ils ne concernent pas directement les objectifs des projets mais s'adressent à la maîtrise de l'activité d'ingénierie système en elle–même. Ils définissent des méthodes d'évaluation de l'IS pouvant être génériques (SE-CMM, SECAM et EIA 731) ou spécifiques à un domaine (SW-CMM et ISO 15504 pour le logiciel, FAA-iCMM pour l'aviation et CMMI pour la défense).

1.5 Notre norme référante : l'EIA-632

Nous avons vu quelles étaient les principales normes d'ingénierie système ainsi que les mécanismes sur lesquels elles reposent. Nous allons à présent détailler celle que nous avons choisie pour valider notre démarche : l'EIA-632. Nous présenterons quels sont les concepts manipulés par cette norme et quels sont les processus qu'elle recommande.

En premier lieu, utiliser l'une des trois grandes normes de l'ingénierie système s'est imposé car :

- elles sont toutes des référentiels internationaux reconnus par l'industrie et par le milieu académique,
- elles représentent chacune la synthèse d'une compilation de bonnes pratiques établies sur de nombreuses années,
- conformément aux objectifs de l'ingénierie système que nous poursuivons, elles couvrent une grande part du cycle de vie du système,
- elles sont très génériques et touchent de nombreux domaines d'activités allant du médical au militaire en passant par les services.

1.5.1 Pourquoi choisir l'EIA-632 comme référence ?

Nous avons choisi de travailler sur la base des processus décrits par l'EIA-632 pour plusieurs raisons :

Le choix de l'EIA-632 en particulier est survenu en raison de la couverture du cycle de vie : l'IEEE 1220 est trop centrée sur les opérations de conception pour valider notre démarche ; l'ISO 15288 définit les processus d'entreprise et vient en complément des deux autres que l'on aurait dû alors intégrer au préalable.

Le choix de l'EIA-632 n'est en rien restrictif : la méthodologie que nous proposerons dans la suite de cette thèse ne lui est pas spécifique. Il sera tout à fait possible de l'appliquer sur la base d'une autre norme d'ingénierie système voire, éventuellement, un autre processus de référence sous réserve qu'il soit assez générique.

1.5.2 Quelques rappels généraux sur l'EIA-632

L'EIA-632 [Ele99, Mar98b] est un standard développé conjointement par l'EIA et l'INCOSE intitulé « processes for engineering a system ». Il remplace le précédent standard EIA/IS 632.

Le rôle de l'EIA-632 est de fournir un ensemble de processus fondamentaux pour aider un développeur dans la conception ou l'amélioration d'un système. Le respect des processus et des recommandations de cette norme doit permettre aux développeurs :

1. d'établir et de faire évoluer un ensemble complet et cohérent d'**exigences** qui permettront la livraison de solutions systèmes réalisables et rentables,

2. de satisfaire ces exigences selon des contraintes de coûts, de délais et des contraintes liées aux risques encourus,
3. de fournir un système, ou une portion de système, qui satisfasse les parties intéressées tout au long des durées de vie des produits qui constituent le système,
4. de fournir une mise à disposition ou un retrait du système sûr et rentable.

Son emploi est, en principe, basé sur une démarche d'acceptation volontaire. Cette norme :

– ne doit pas être imposée par contrat,
– doit être appliquée dans un esprit de développement des bonnes pratiques de l'entreprise.

Comme pour les autres normes d'IS, l'implantation des exigences de l'EIA-632 passe par le filtre des politiques et procédures spécifiques à l'entreprise et au projet considéré selon un schéma (figure 1.6) qui nous servira à guider la définition de nos propres propositions.

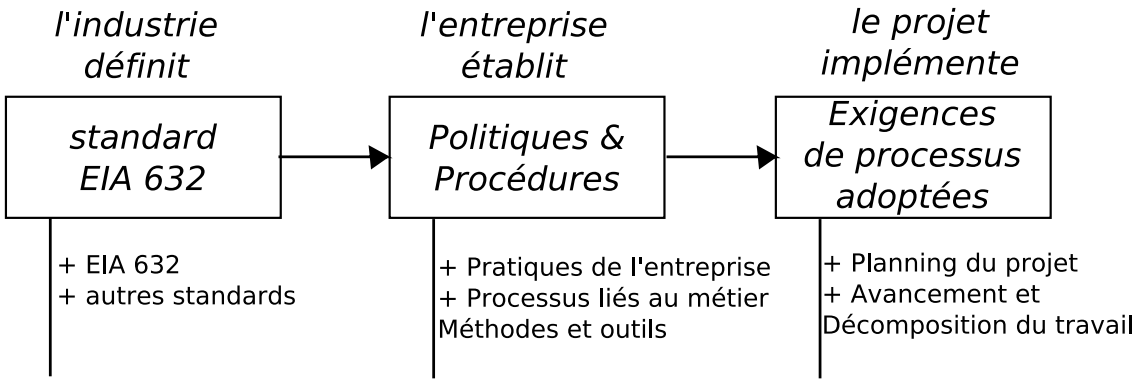


FIG. 1.6: *Rôle de l'EIA-632 par rapport aux pratiques métier et projet. L'objectif de l'EIA-632, ou d'une partie de l'EIA-632, est de permettre par son application l'établissement de politiques et procédures définissant les exigences pour l'implantation des projets [Ele99].*

L'EIA-632 est une démarche ouverte. Elle précise qu'il n'est pas de sa responsabilité de répondre aux trois questions suivantes :

– ce qu'est l'ingénierie système,
– ce que l'ingénieur système est supposé faire,
– ce que l'organisation d'ingénierie système est supposée faire.

Les réponses à ces trois points doivent provenir d'autres sources que ce standard.

1.5.3 Quelques rappels sur le contenu et les concepts de l'EIA-632

L'EIA-632 comporte 6 clauses et 7 annexes. Les titres et les contenus des clauses et des annexes sont reportés sur le tableau 1.4.

La clause 4 présente les exigences associées aux processus d'ingénierie d'un système. Elle est sans doute la plus importante de la norme. D'une part, par sa taille et, d'autre part, par le fait que les autres clauses ou annexes lui font référence pour préciser le cadre d'application des processus ou pour définir des concepts complémentaires nécessaires à l'application des processus.

Nous présentons succinctement ici les principaux concepts utiles à l'ingénierie d'un système selon l'EIA-632.

1.5.3.1 Le projet et son environnement

Les processus techniques prennent place dans le cadre d'un projet dont la finalité première est le développement d'un système. Mais, ce projet se déroule dans une entreprise qui doit y intégrer tous les éléments de son environnement. La figure 1.7 donne une illustration des éléments qui peuvent être liés au projet, à l'entreprise ou à l'environnement et qui doivent être pris en compte.

La mise en place des processus techniques, au–delà des spécificités de l'environnement, doit être adaptée en fonction des processus internes à l'entreprise et des processus propres au projet, comme on l'a vu à la figure 1.6.

1.5.3.2 Proposition d'une structure générique du système

L'EIA-632 propose une structure de système générique regroupant le concept de système : le produit final, les produits contributeurs, les blocs de construction et les sous–systèmes dont les termes sont définis ci–dessous :

Système :

La définition d'un système (définition 11) selon l'EIA-632 est peu restrictive. Elle désigne aussi bien un ensemble de composants matériels ou logiciels que des bâtiments, des données, des services, des personnes, des fournitures, des techniques ou tout autre produit selon le métier de l'activité considérée.

Définition 11 : *Système (selon EIA-632)*

Agrégation de produits finaux et de produits contributeurs dans un but donné.

Section	Titre	Contenu
Clause 1	Objectifs	Définition des objectifs et des processus sur lesquels appliquer la norme.
Clause 2	Références normatives	Liste des normes référencées dans le texte et normes nécessaires à l'application de l'EIA-632.
Clause 3	Définitions et acronymes	Définition des termes spéciaux et des acronymes.
Clause 4	Exigences	Les exigences sur les processus d'ingénierie d'un système. Les tâches représentatives des processus sont définies.
Clause 5	Contexte d'application	Description du contexte dans lequel appliquer les processus décrits.
Clause 6	Concepts d'application clefs	Décrit les concepts clefs nécessaires à l'application des processus de la clause 4 à la génération et l'application des couches de produits nécessaires à l'ingénierie d'un système.
Annexe A	Glossaire	Définition des termes employés d'une manière spécifique dans la norme.
Annexe B	Cycle de vie basé entreprise	Décrit la gestion des phases du cycle de vie d'un système ou d'une portion de système développé incrémentalement.
Annexe C	Résultats attendus	Les résultats attendus des tâches identifiées en clause 4.
Annexe D	Documents attendus	Liste des principaux documents relatifs à l'ingénierie d'un système.
Annexe E	Revues techniques du système	Décrit les revues techniques nécessaires.
Annexe F	Développement sans et avec précédents	Décrit l'application des exigences de la conception système selon que le développement du système soit avec ou sans précédent.
Annexe G	Relations entre exigences	Définit les types d'exigences, leurs relations entre elles et avec les solutions physiques et logiques.

TAB. 1.4: *Contenu de l'EIA-632*

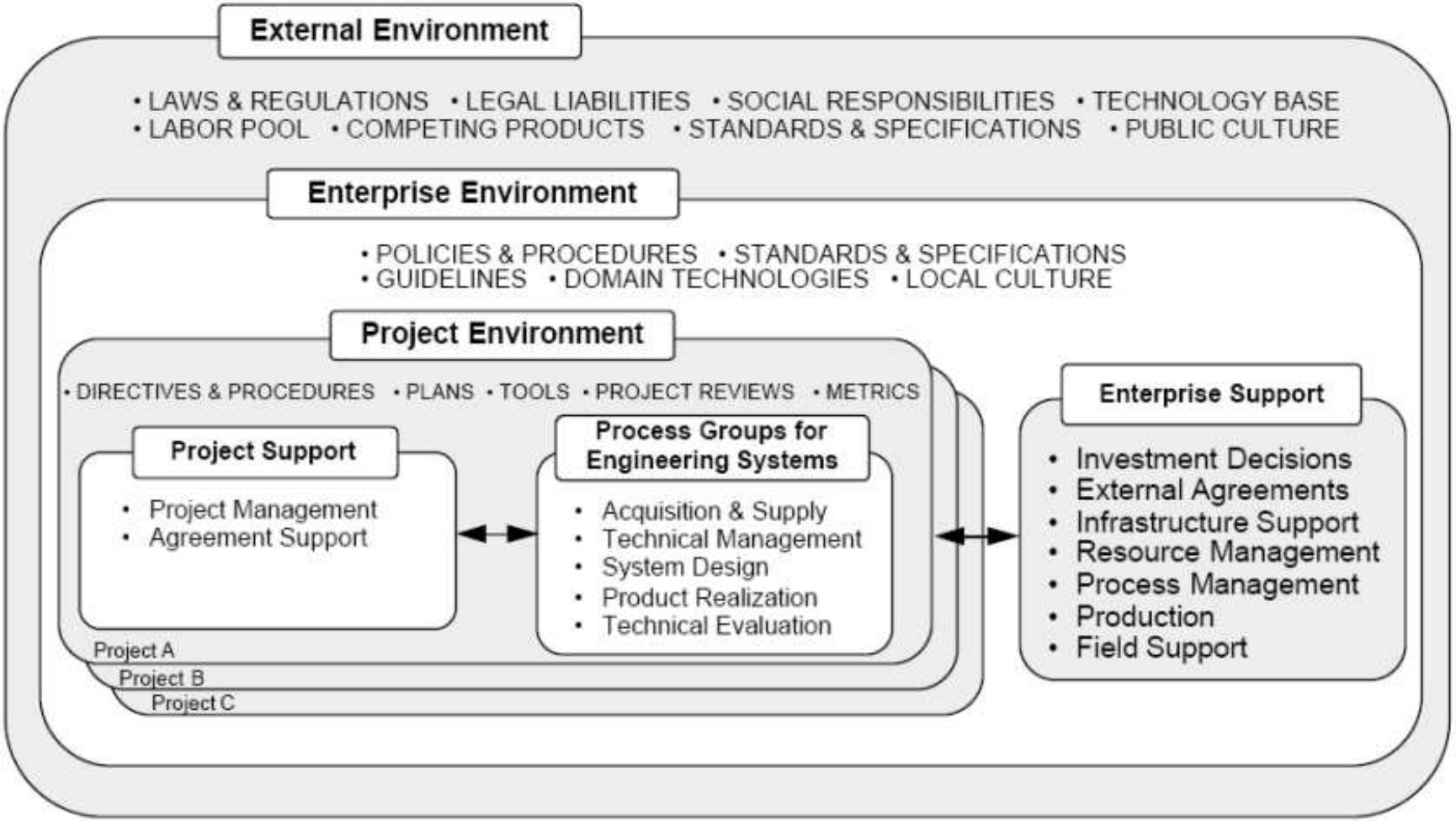


FIG. 1.7: *Environnement de projet selon l'EIA-632. L'environnement du projet est composé de son environnement immédiat, de l'environnement lié à l'entreprise (politiques et procédures, services de support) et de l'environnement externe à l'entreprise (lois, réglementations, concurrence, culture, etc.) (figure 5.0 de l'EIA-632 [Ele99]).*

Produit final et produit contributeur :

Comme illustré à la figure 1.8, le système est constitué non seulement de *produits finaux* (end products) mais aussi de *produits contributeurs* (enabling products). **Les produits finaux sont les produits qui vont réaliser les fonctions opérationnelles du système** et qui seront livrés au client et utilisés par le consommateur. **Les produits contributeurs**[7] ne réalisent pas de fonctions opérationnelles mais « **permettent** » **aux produits finaux d'être développés, testés, mis à disposition, produits, déployés, supportés ou encore utilisés pour la formation du personnel.**

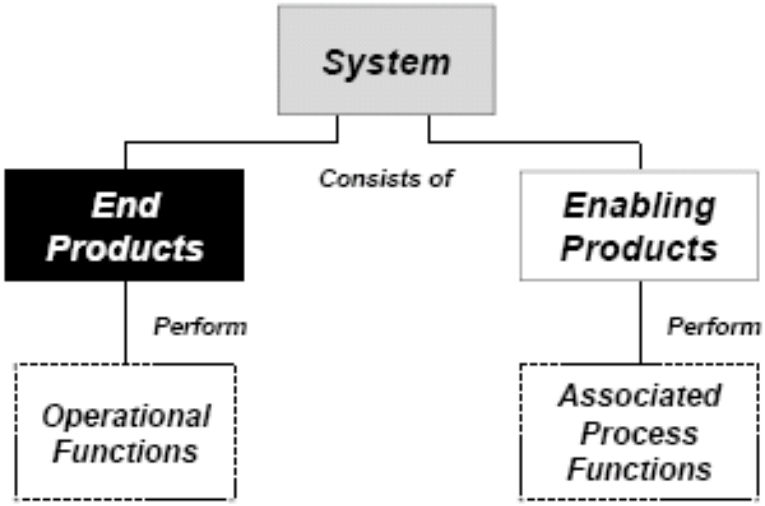


FIG. 1.8: *Concept de système vu par l'EIA-632. Un système est constitué d'un ou plusieurs produits finaux réalisant des fonctions opérationnelles et éventuellement de produits contributeurs qui permettent de réaliser les fonctions liées aux processus (figure 6.1 de l'EIA-632 [Ele99]).*

Les produits contributeurs sont classés selon le processus auquel ils se rapportent. On trouve ainsi des :

- produits de développement,
- produits de test,
- produits de formation,
- produits de retirement,
- produits de production,

[7] Comme précisé dans l'EIA-632, le concept de produit contributeur inclut implicitement le personnel qui développe, produit, teste, utilise, supporte et retire les produits du système aussi bien que ceux qui forment les personnes impliquées dans les fonctions du système et les facteurs humains associés à ces personnels. Les facteurs humains liés au personnel sont inclus dans l'application de ces processus aux blocs de construction dérivés du système.

- produits de déploiement,
- et des produits supports.

Cette liste n'est cependant pas exhaustive et peut être enrichie selon les besoins du projet. On peut ajouter par exemple les procédures industrielles, le personnel, les services, *etc.*

Cette vision générique des systèmes laisse la possibilité de l'adapter à chaque métier. Par exemple, dans l'industrie télévisuelle, on peut avoir les produits finaux suivants :

matériel : caméras, moniteurs,

logiciel : outils de planification, algorithmes de compression,

personnel : cameramans, présentateurs,

locaux : studio, station de transmission,

données : script, guide de programme,

matériaux : images, histoires,

services : transports, téléphone,

techniques : méthodes de présentation, procédures d'édition,

médias : radio, internet.

Bloc de construction et développement en couches :

Il peut arriver que tous les produits finaux et produits contributeurs soient disponibles « sur étagère ». Dans ce cas, il n'est pas nécessaire de pousser la décomposition du système plus loin. Mais, dans la majorité des cas, il est nécessaire pour ces produits finaux ou contributeurs qui doivent être décomposés et développés à leur tour d'appliquer les mêmes processus et recommandations de l'EIA-632. Pour ce faire, l'EIA-632 introduit la notion de « bloc de construction » ou building block (figure 1.9).

Un bloc de construction (définition 12) est un concept combinant tous les éléments nécessaires au développement du système. Il comprend ainsi la décomposition en produits finaux et produits contributeurs du système, les exigences, le travail et toute autre information nécessaire à la réalisation du système. En cela, il est bien plus qu'une simple décomposition architecturale du système.

Définition 12 : *Bloc de construction*

Il est un élément dans la décomposition structurelle du système. Il conduit à une représentation du cadre conceptuel du système projeté utilisé pour organiser les exigences, le travail, et les autres informations associées à l'ingénierie système.

L'utilisation des blocs de construction permet ainsi au développeur de s'assurer que l'ensemble du cycle de vie des produits finaux a été pris en compte.

Dans un bloc de construction, les produits finaux peuvent être décomposés en *sous-systèmes*. Chaque sous-système nécessite un développement propre dans lequel il est considéré comme un système de base. En associant un bloc de construction de couche inférieure aux sous–systèmes (figure 1.10(a)), on doit pousser la décomposition du système jusqu'à ce que tous les sous–systèmes soient définis, spécifiés, développés, jusqu'à ce qu'ils soient disponibles sur étagère ou puissent être acquis auprès de fournisseurs. On parle alors de décomposition en couches.

De la même manière, un produit contributeur qui nécessite d'être développé se voit lié à un bloc de construction (figure 1.10(b)).

Les différents processus ne sont pas obligatoirement sous la responsabilité d'un projet unique. Ils peuvent être partagés entre différents projets ou sous-projets. La figure 1.11 donne un exemple de développement en couches.

1.5.3.3 Les processus constitutifs de l'EIA-632

La clause 4 de l'EIA-632 décrit 13 processus et les résultats attendus de ces processus, ainsi que 33 exigences. Ces processus sont classés en cinq groupes représentés aux figures 1.12 et 1.13. Ce regroupement est fait pour faciliter l'organisation du standard mais n'est pas requis en tant que structure pour l'implantation des processus. La liste des exigences liées aux processus est donnée au tableau 1.5 page 45.

En fait,groupes, processus et exigences de l'EIA-632 décrivent tous des activités à réaliser et le résultat attendu de ces activités. L'ensemble des processus se présente sous forme hiérarchique : les processus de l'EIA-632 sont des sous–processus des disciplines et les exigences des sous–processus des processus de l'EIA-632. Les derniers éléments de cette décomposition sont les tâches élémentaires associées aux exigences.

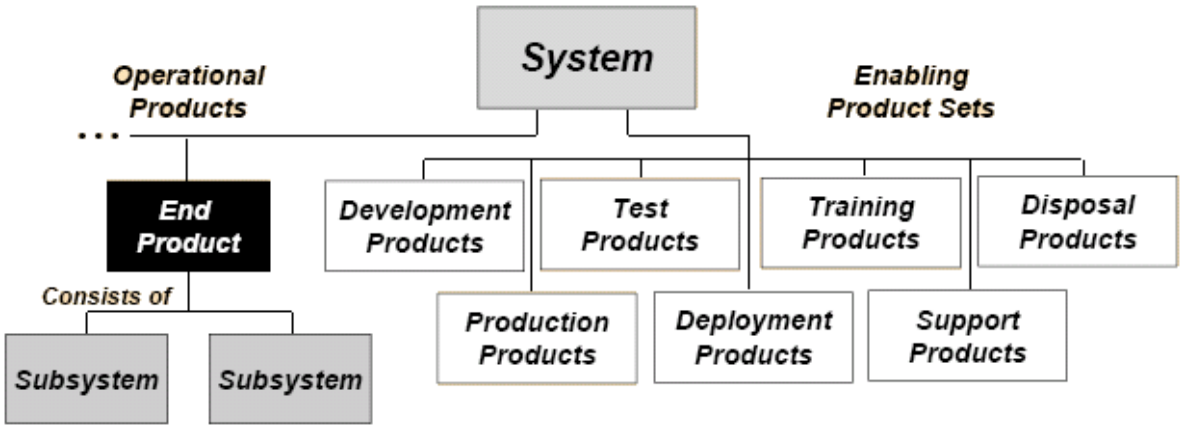
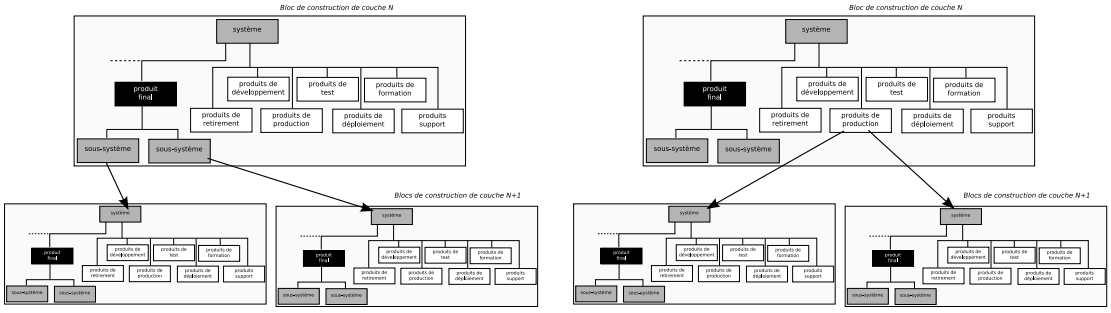


FIG. 1.9: *Concept de bloc de construction. Les éléments liés à la réalisation d'un système (exigences, travail ou autres informations) sont classés dans le cadre conceptuel d'un building block selon qu'ils concernent : un produit final composé d'au moins deux sous–systèmes, ou, un produit contributeur permettant le développement, le test, la formation, la mise en service, la production, le déploiement ou le support (figure 6.1.1 de l'EIA-632 [Ele99]).*



(a) Développement en couches des sous-systèmes

(b) Développement en couches des produits contributeurs

FIG. 1.10: *Principe de développement en couches. Les composants d'un système, qu'il s'agisse de sous–systèmes ou de produits contributeurs, sont développés en couches. Les exigences identifiées pour ces composants dans un bloc de construction mènent à la création d'un nouveau bloc de construction de couche inférieure où le composant est considéré comme un système à part entière.*

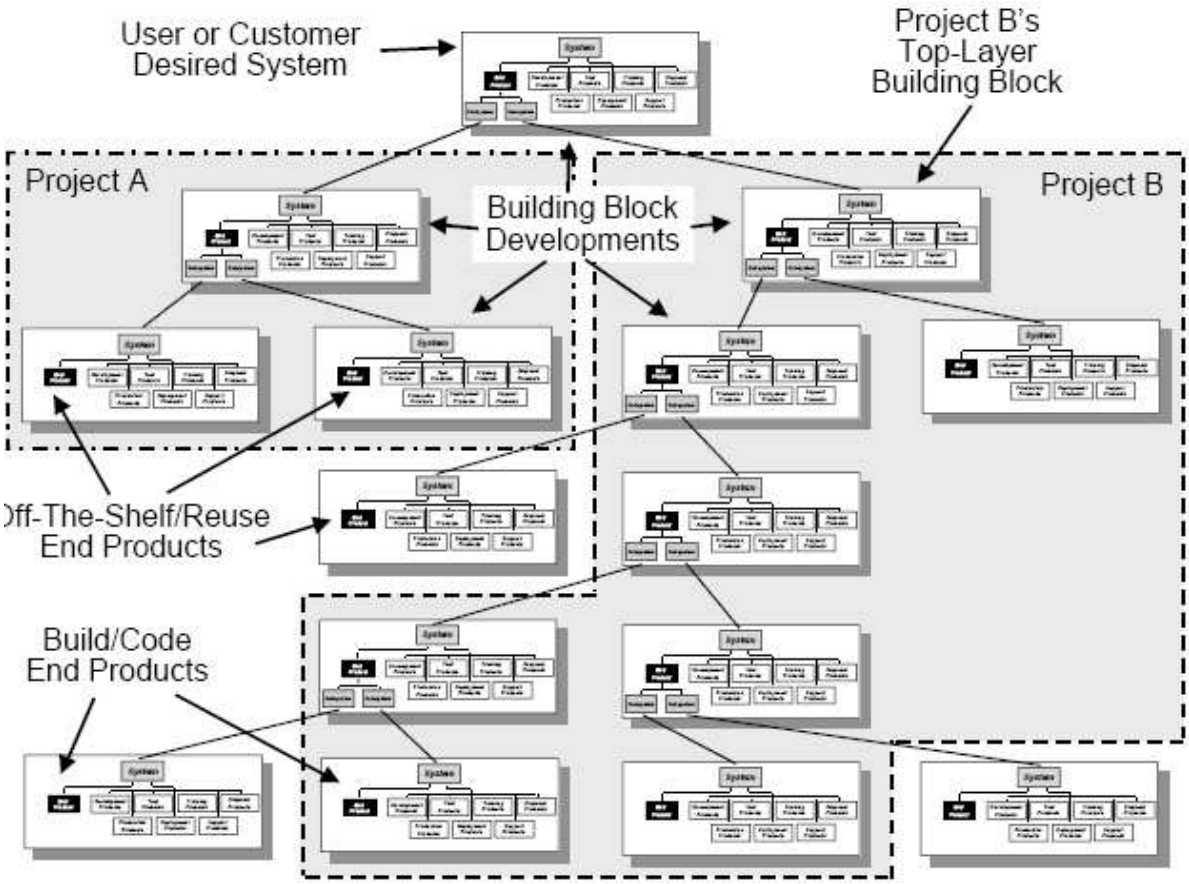


FIG. 1.11: *Exemple de développement en couches. Les systèmes identifiés dans les blocs de construction sont décomposés jusqu'à des niveaux où les produits finaux sont soit développés soit sur étagère. Dans cet exemple, les deux premiers blocs de construction sont la base de deux projets distincts nécessaires à la réalisation du système désiré par l'utilisateur ou le client (figure 6.2.1a de l'EIA-632 [Ele99]).*

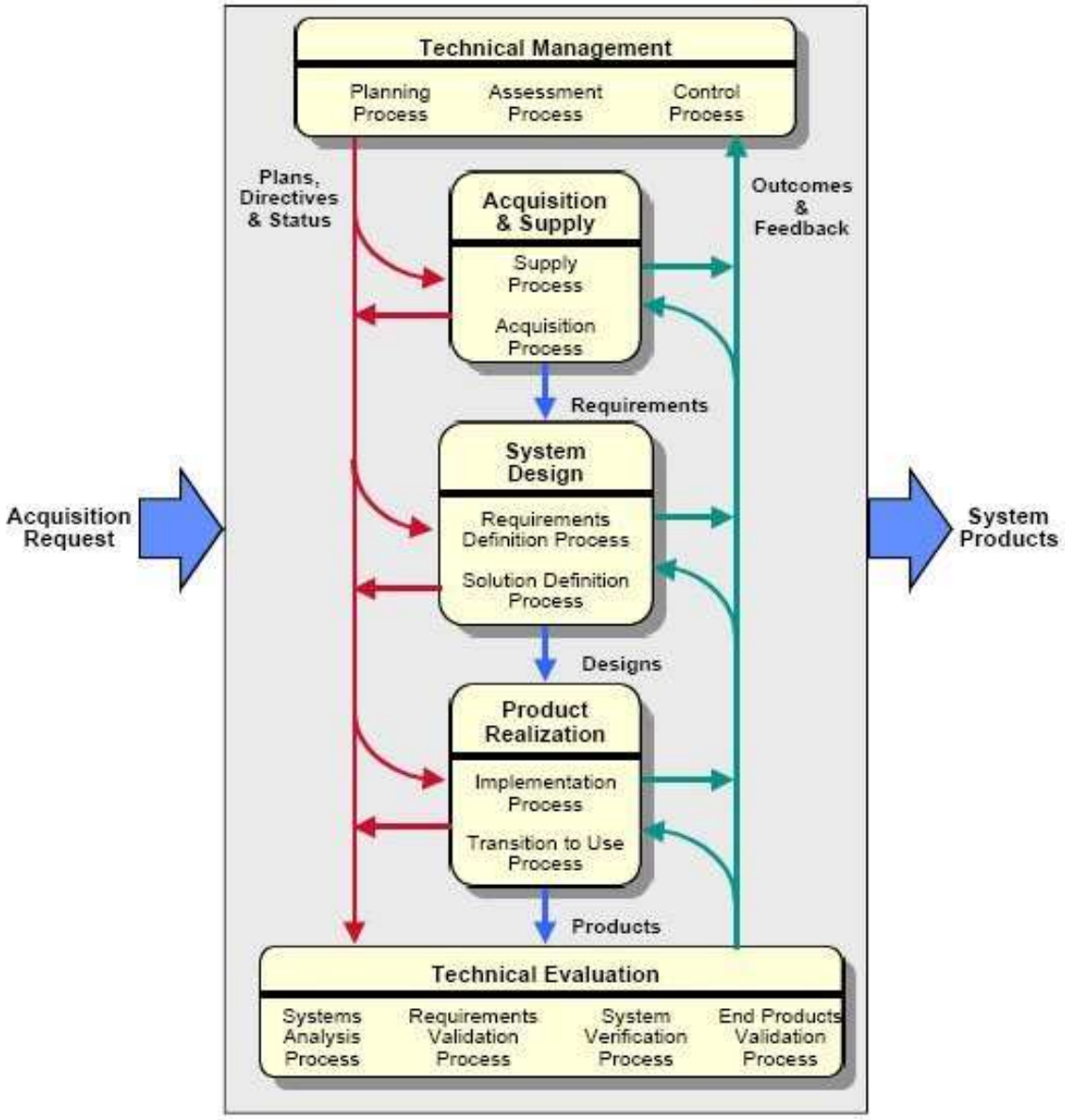
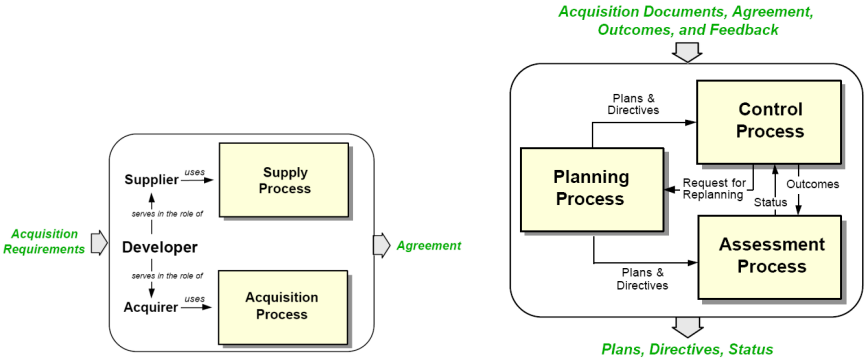
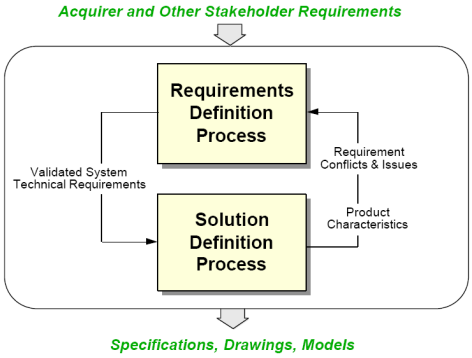


FIG. 1.12: *Interconnexion des processus de l'EIA-632. Les 13 processus de l'EIA-632 sont classés en 5 groupes : Technical Management, Acquisition & Supply, System Design, Product Realization et Technical Evaluation. Ce regroupement n'est pas requis en tant que structure pour l'implantation des processus (figure 4a de l'EIA-632 [Ele99]).*

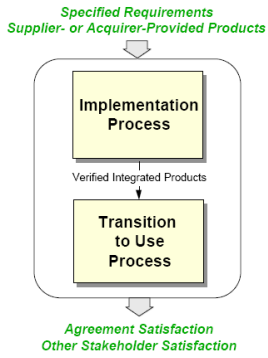


(a) Acquisistion and Supply

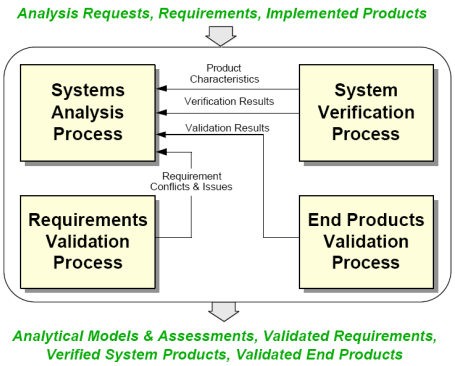
(b) Technical Management



(c) System Design



(d) Product Realization



(e) Technical Evaluation

FIG. 1.13: *Groupes des processus de l'EIA-632. Répartition des processus dans les groupes et relations inter–processus (figures 4.1, 4.2, 4.3, 4.4 et 4.5 de l'EIA-632 [Ele99]).*

SUPPLY PROCESS REQUIREMENTS	12 – Outcomes Management	22 – Effectiveness Analysis
1 – Product Supply	13 – Information Dissemination	23 – Tradeoff Analysis
		24 – Risk Analysis
ACQUISITION PROCESS REQUIREMENTS	REQUIREMENTS DEFINITON PROCESS REQUIREMENTS	
2 – Product Acquisition	14 – Acquirer Requirements	REQUIREMENTS VALIDATION PROCESS REQUIREMENTS
3 – Supplier Performance	15 – Other Stakeholder Requirements	25 – Requirement Statements Validation
	16 – System Technical Requirements	26 – Acquirer Requirements Validation
PLANNING PROCESS REQUIREMENTS	SOLUTION DEFINITION PROCESS REQUIREMENTS	27 – Other Stakeholder Requirements Validation
4 – Process Implementation Strategy	17 – Logical Solution Representations	28 – System Technical Requirements Validation
5 – Technical Effort Definition	18 – Physical Solution Representations	29 – Logical Solution Representations Validation
6 – Schedule and Organization	19 – Specified Requirements	
7 – Technical Plans		SYSTEM VERIFICATION PROCESS REQUIREMENTS
8 – Work Directives	IMPLEMENTATION PROCESS REQUIREMENTS	30 – Design Solution Verification
ASSESSMENT PROCESS REQUIREMENTS	20 – Implementation	31 – End Product Verification
9 – Progress Against Plans and Schedules	TRANSITION TO USE PROCESS REQUIREMENTS	32 – Enabling Product Readiness
10 – Progress Against Requirements	21 – Transition to Use	EEN PRODUCTS VALIDATION PROCESS REQUIRMENTS
11 – Technical Reviews		33 – End Products Validation
CONTROL PROCESS REQUIREMENTS	SYSTEMS ANALYSIS PROCESS REQUIREMENTS	

TAB. 1.5: *Liste des exigences liées aux processus de l'EIA-632. La traduction française de ce tableau est donnée en annexe 4.5.*

L'application de ces processus aux différents blocs de construction permet d'assurer la cohérence globale de l'ingénierie du système. **Elle guide le passage des besoins du client en une solution logique puis une solution physique du système et valide le produit fabriqué ou acquis avant sa livraison.**

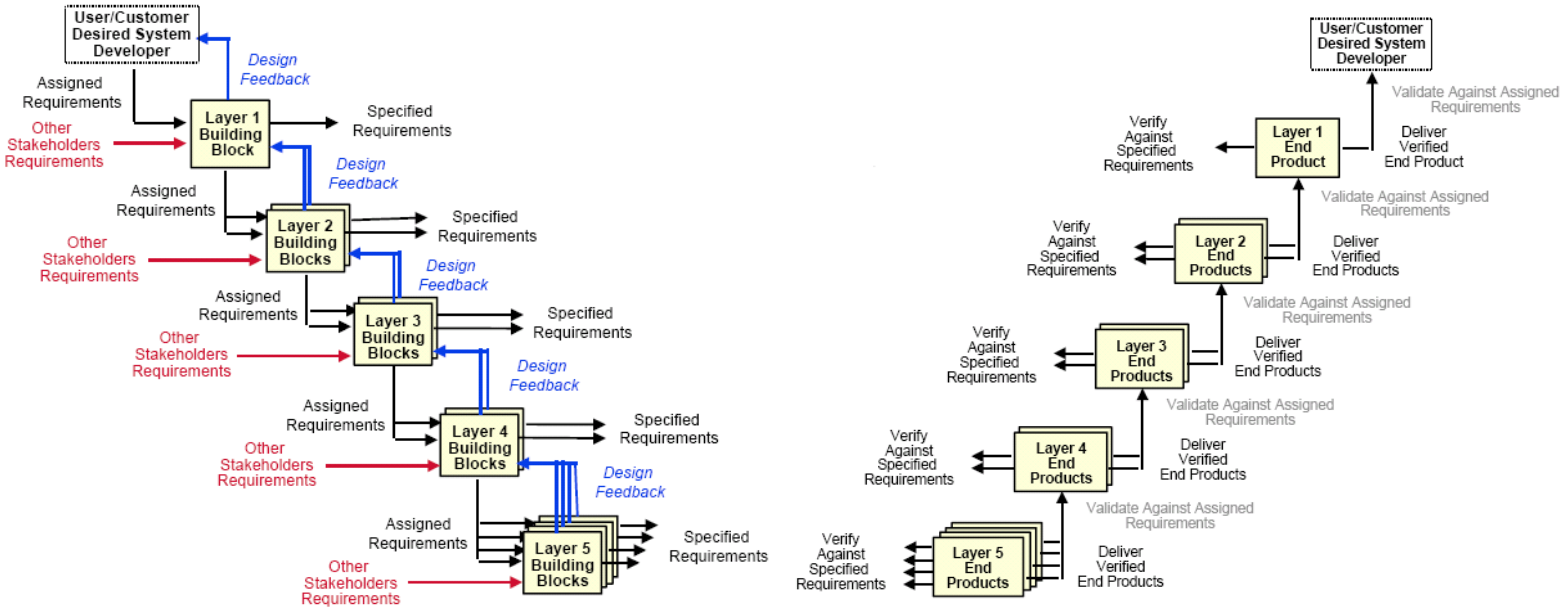
1.5.3.4 Le concept de cycle de vie du système

Les processus de conception, vérification et validation s'appliquent à tous les blocs de construction. L'EIA-632 recommande de suivre une démarche descendante des spécifications au prototypage et une réalisation physique ascendante. Dans cette démarche descendante (figure 1.14(a)), les besoins identifiés à un niveau vont servir d'exigences pour les blocs de construction de niveaux inférieurs. Une fois tous les niveaux spécifiés, la réalisation ascendante se fait en vérifiant chaque niveau (figure 1.14(b)). En agissant ainsi, on s'assure de la conformité du système global développé par rapport aux exigences initiales que ce soit pour les produits finaux le constituant ou pour ses produits contributeurs.

Une fois tous les blocs de construction spécifiés, les produits correspondants sont acquis ou fabriqués. C'est l'assemblage de ces éléments qui conduit à la construction du système principal. Une fois assemblé, il doit être validé par rapport à ses spécifications.

L'EIA-632 porte une vision globale de l'ingénierie d'un système. En pratique, l'EIA-632 propose d'adapter les principes de développement que nous venons de décrire en fonction de la phase du cycle de vie du système traité. Un exemple est reporté figure 1.15. Dans cet exemple, cinq phases du cycle de vie d'un produit sont évoquées :

- Dans la phase d'évaluation préliminaire, l'EIA-632 s'applique à une démarche descendante d'évaluation qui se conclut par une première série de spécification pour le système.
- Dans la phase de décision et de lancement, il s'agit d'effectuer les premières estimations par simulation d'un prototype virtuel ou fonctionnel pour évaluer les risques et étudier les perspectives. La démarche reste descendante.
- Dans la phase de conception, jusqu'aux premières réalisations, on trouve toute la richesse et toute la rigueur des recommandations de l'EIA-632, où s'entremêlent des démarches systématisées descendantes et ascendantes jusqu'à pouvoir définir le système final, les outils de production, de test, de support…



(a) Développement descendant

(b) Réalisation ascendant

FIG. 1.14: *Développement et réalisation des blocs de construction. Le développement consiste à pousser à bout le processus de décomposition des blocs de construction en couches. A chaque couche, les exigences initiales, reportées jusqu'aux niveaux les plus bas, se voient complétées par des exigences spécifiques à chaque bloc de construction. Ce processus descendant assure la diffusion et la cohérence des exigences. La réalisation suit un processus ascendant. Le résultat de chaque bloc de construction est validé par rapport aux exigences correspondantes avant validation des niveaux supérieurs (figures 6.2.1b et 6.2.2 de l'EIA-632 [Ele99]).*

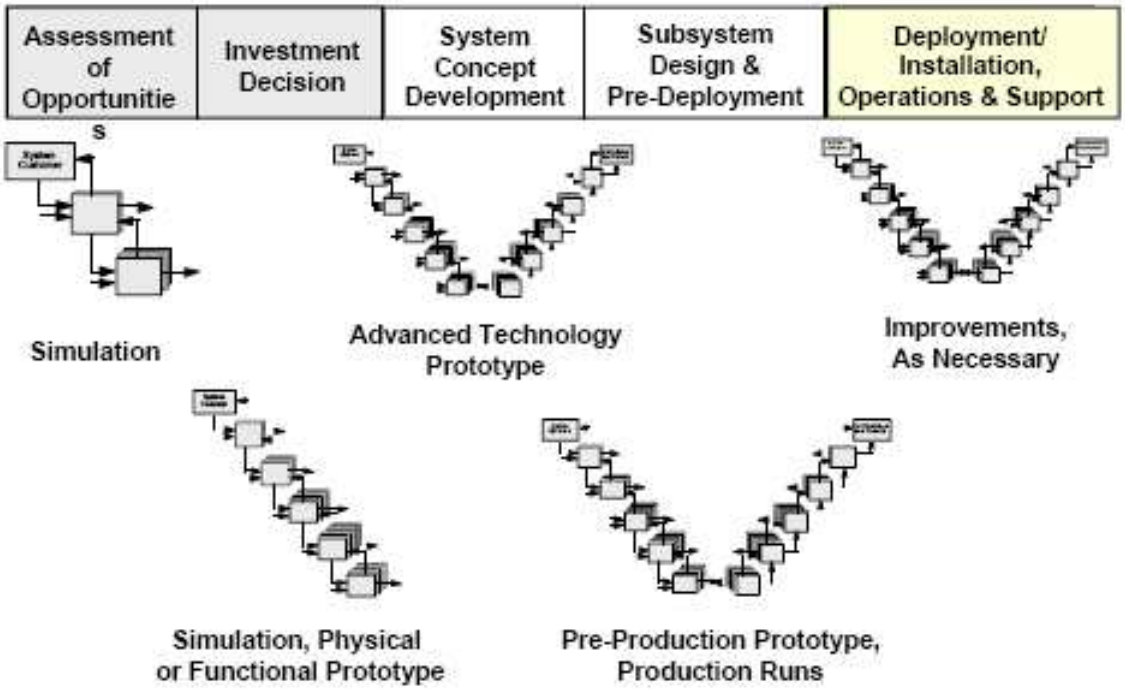


FIG. 1.15: *Développement itératif du système en fonction du cycle de vie. Cet exemple illustre cinq phases typiques d'un cycle de vie. On constate qu'en fonction des besoins en un instant donné, différents processus sont actifs. Dans les phases prédictives, les processus sont en cascade alors que par la suite, ils sont de type descendant/ascendant (figure B.1 de l'EIA-632 [Ele99]).*

– La figure 1.15 montre encore comment, en phase de déploiement, il reste essentiel d'appliquer le standard EIA-632 pour que ce déploiement respecte le cahier des charges fixé à l'origine.

1.5.3.5 Les exigences de l'EIA-632

Les exigences sont au cœur des recommandations de l'EIA-632. L'enchaînement des processus proposé et la structuration du système sont pensés de manière à organiser au mieux la collecte des exigences, leur utilisation, leur traitement et leur suivi jusqu'à la fin du cycle de vie.

Chaque bloc de construction reçoit des blocs de niveaux supérieurs ses exigences entrantes qui peuvent être des :

exigences utilisateur : Les exigences des utilisateurs sont souvent non-techniques et peuvent être contradictoires les unes avec les autres. Elles doivent être traduites en *exigences techniques* qui sont spécifiques au métier et à la technologie employée par le système ;

exigences client : Elles proviennent de l'acquéreur du produit. Celui-ci peut avoir ses propres exigences ou percevoir les exigences des utilisateurs ;

exigences d'une autre partie prenante : En fonction de l'environnement, d'autres parties intéressées peuvent apporter leur lot d'exigences sur le système.

Ces exigences vont être traitées jusqu'à obtenir un ensemble d'exigences techniques cohérentes avec le processus de conception envisagé. Les exigences spécifiées pour un sous-système seront considérées comme assignées dans le bloc de construction de couche inférieure. Les différents types d'exigences et leurs traitements sont présentés figure 1.16. Les liens de ces exigences avec les processus de l'EIA-632 apparaissent figure 1.17.

Ce sont ces relations entre exigences, guidées par l'application des processus, qui permettent d'assurer la cohérence de la structure du système et de valider les blocs de construction réalisés dans les phases descendantes.

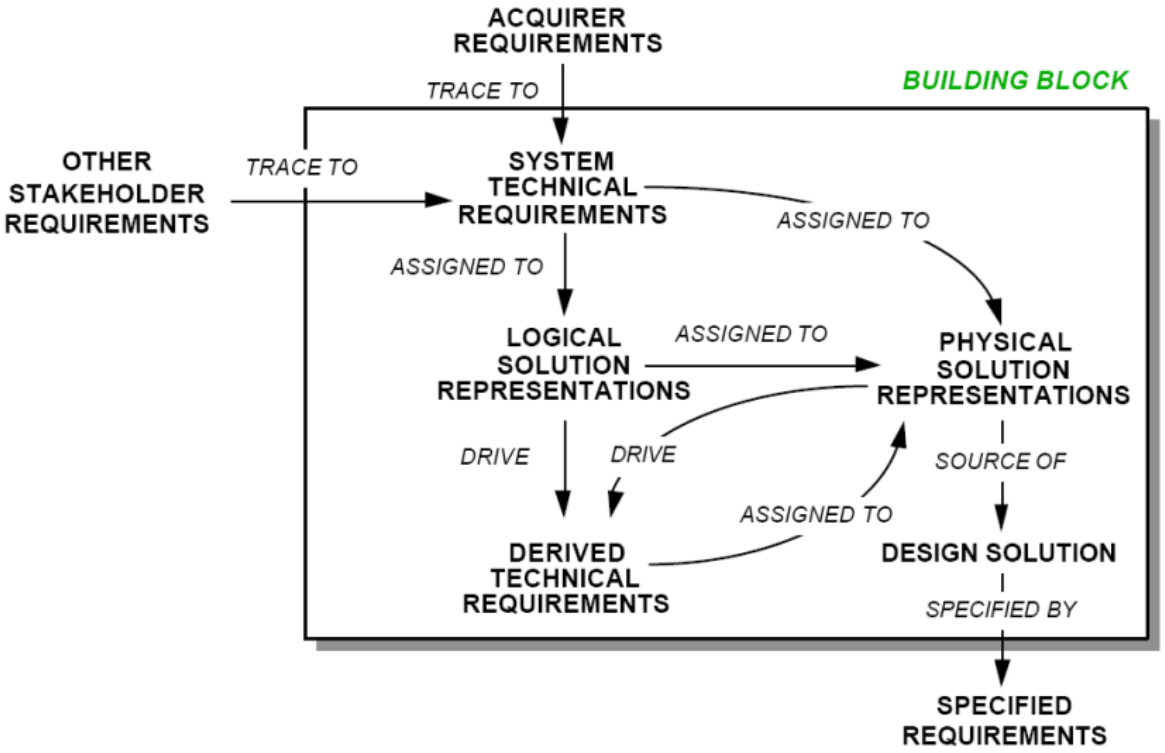


FIG. 1.16: *Relations entre exigences. Les exigences techniques du système sont déduites à partir des exigences des clients, utilisateurs ou des autres parties prenantes et des exigences issues du bloc de construction de niveau supérieur. Les solutions logiques et physiques sont déduites de ces exigences techniques et sont à la base de nouvelles exigences techniques ou d'exigences spécifiées qui seront la base de blocs de construction de niveaux inférieurs (figure G.1 de l'EIA-632 [Ele99]).*

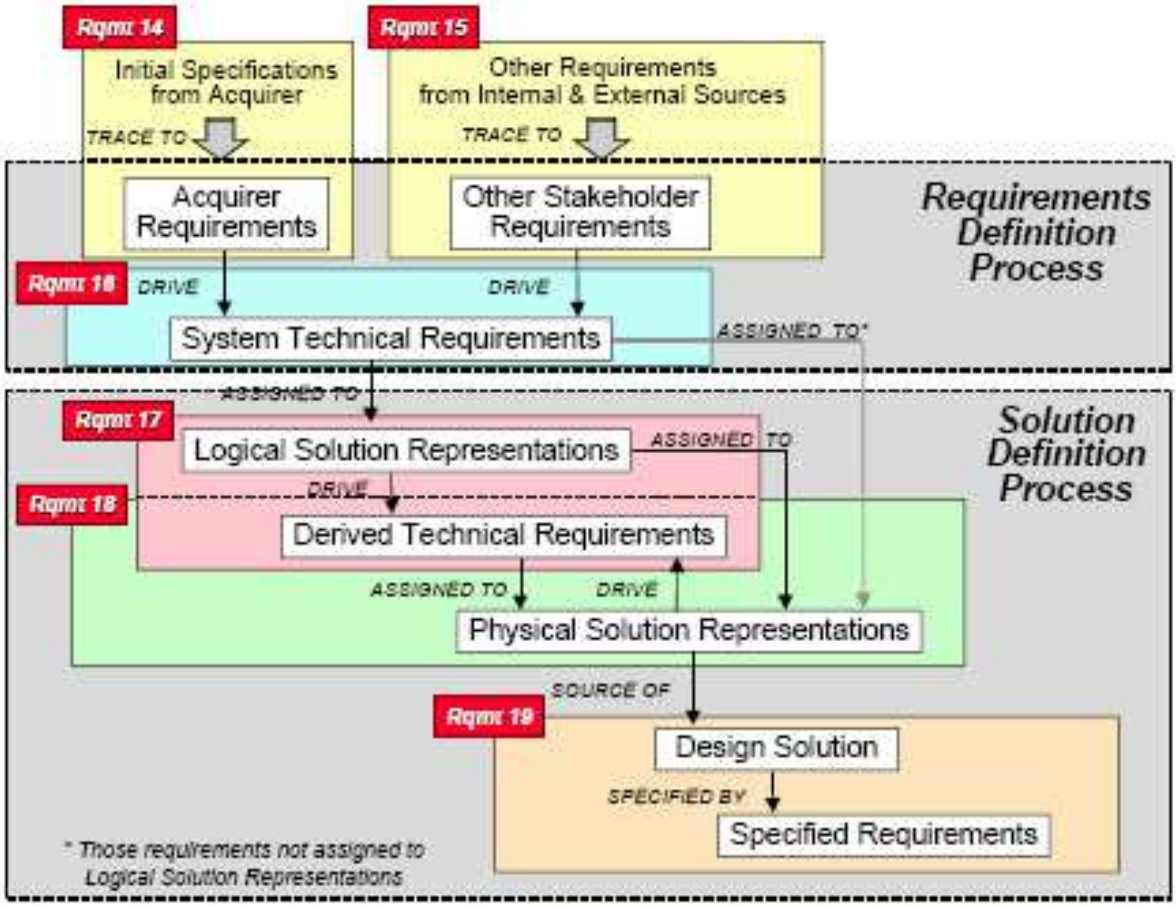


FIG. 1.17: *Relations entre exigences et éléments de l'EIA-632. Le passage des exigences non–techniques issues des acquéreurs ou des autres parties prenantes en exigences techniques du système est réalisé par le processus de définition des exigences. Le processus de définition des solutions vient compléter le traitement des exigences. Il reprend les exigences techniques et les fait évoluer jusqu'aux solutions de conception. Ces deux processus sont complétés par les exigences de processus 14 à 19 qui couvrent chacune des transformations entre exigences du système (figure G.4 de l'EIA-632 [Ele99]).*

1.6 Notre problématique

La conception des systèmes est un exercice difficile et essentiel au développement économique, notamment dans les entreprises et les pays techniquement très développés. On attend de la conception qu'elle fixe des objectifs de fabrication précis pour la réalisation des systèmes tout en souhaitant qu'ils soient performants, fiables et de coûts contrôlés…

Les difficultés sont nombreuses et diverses : il faut co–gérer des problèmes de matériaux divers, de fonctions multiples et interactives, de disciplines différentes… Il faut innover mais de façon pondérée pour pouvoir bénéficier d'acquis antérieurs et de leur ré-utilisation. Il faut être sûr de ses choix en appliquant systématiquement des procédures de vérification dès les premières étapes de la conception.

Toute erreur, la plus insignifiante, peut, dans ses conséquences, bouleverser le processus global en remettant en cause les étapes suivant la phase d'introduction de l'erreur mais aussi les étapes antérieures si l'erreur est insuffisamment qualifiée ou si le processus de conception n'est pas strictement descendant. On voit ici apparaître deux règles majeures de la conception moderne d'un système complexe :

Règle 1 : Le processus général de la conception d'un système complexe doit être strictement descendant du cahier des charges aux spécifications de fabrication.

Règle 2 : Toute solution partielle proposée pour intégration dans le processus général a fait l'objet de spécifications cohérentes, précises et complètes qu'elle vérifie rigoureusement (spécification–vérification).

Mais, ce processus général descendant est constitué de processus multiples qui ne sont pas nécessairement indépendants les uns des autres. Des échanges paramétriques et architecturaux sont nécessaires pour faire les ajustements et parfaire les choix, mais ces choix doivent respecter les règles d'une double vérification des deux processus impliqués. De plus, on ne se situe pas ici dans un problème strictement technique. Les étapes permettant de passer de spécifications à une ébauche de solution technique, qu'elles soient constituées d'un produit ou de spécifications pour un sous-produit, sont sous la responsabilité d'équipes de conception dédiées qui vont les mener à bien. La problématique est aussi organisationnelle. La conception technique n'est qu'un des aspects du projet. Il s'agit de ne pas trouver uniquement une réponse aux contraintes techniques mais de s'assurer, de manière préventive, que la solution proposée est valide par rapport à tous les éléments de l'environnement du projet.

La réponse à cette problématique difficile doit être apportée par le développement d'une **ingénierie système** : la démarche d'ingénierie système doit donc proposer une approche mul-

tidisciplinaire dans laquelle toutes les activités des disciplines d'ingénierie seront coordonnées. En l'état des connaissances et des pratiques, cette coordination des activités est guidée par des processus génériques. Ces processus décrits dans différentes normes d'ingénierie système décrivent la conception, la réalisation et la diffusion de systèmes, ou de services, selon une démarche descendante composée d'étapes multiples de spécifications–validations dans le but d'assurer la conformité du système par rapport aux exigences initiales. Dans une démarche d'ingénierie système, la problématique posée par le projet doit être considérée de manière globale : le technique (performances du système en termes d'exigences techniques, de qualité, de sûreté de fonctionnement, de fiabilité, *etc.*) sera traité en relation avec les aspects commerciaux (budget disponible, coûts du projet ou encore respect des délais) ou industriels (passage du prototypage à une production de masse, compatibilité technologique, *etc.*) et ce sur tout le cycle de vie du système. L'ensemble du projet doit être établi sur une prise en compte des contraintes et des objectifs technico–économiques provenant de toutes les parties intéressées. Dans un tel contexte, les compromis sont incontournables mais doivent être gérés. C'est le rôle des normes. La description de processus va alors guider et harmoniser les pratiques individuelles pour assurer un déroulement global cohérent du projet et un produit final parfaitement conforme au cahier des charges.

Dans les paragraphes précédents nous avons fait le point de l'état des propositions faisant autorité en matière d'ingénierie système. Compte tenu de la complexité et la pluridisciplinarité, ces propositions se présentent sous la forme de normes et de standards. Nous avons privilégié la norme EIA-632 dont la présentation plus détaillée met en évidence les intérêts et les limites.

L'utilisation de standards décrivant des processus de référence comporte deux intérêts majeurs : d'une part, ils favorisent la compréhension et la communication de tous les acteurs en les regroupant autour d'un référentiel commun et, d'autre part, ils synthétisent les savoirs accumulés et permettent leur partage.

Mais, si aujourd'hui ces processus sont bien inventoriés, il faut se demander comment les mettre en œuvre concrètement : les recommandations qui y sont présentées devraient permettre le développement de politiques et de procédures adaptées à l'environnement de l'entreprise et du projet. Ces standards ne fournissent pas explicitement les indications qui guideraient cette adaptation des processus vers le traitement effectif de projets spécifiques.

Les ingénieurs chargés de mettre en place une démarche d'ingénierie système doivent donc répondre d'eux même aux questions suivantes :

- Par où commencer, quels processus retenir ? Pour une norme donnée, le nombre de concepts importants rend difficile leur mise en œuvre. Des choix doivent être faits quant aux éléments à appliquer. Lesquels sont les plus importants pour ma problématique ? Quels sont ceux que je peux laisser de coté ? Si certains sont supprimés, la cohérence de l'ensemble est-elle toujours respectée ?
- Comment exploiter des recommandations textuelles ? Les normes d'ingénierie sont présentées sous un format textuel qui se prête mal à la manipulation de données et de processus complexes : il ne facilite pas la communication et ne permet pas la vérification.
- Comment ajuster un standard général avec les exigences d'un métier, d'un projet particulier ? Les recommandations des normes sont génériques alors qu'elles doivent être appliquées dans le contexte d'une entreprise et d'un projet particulier. Comment alors affiner des recommandations pour y intégrer le contexte ?
- Comment s'assurer du respect des recommandations générales ? Les procédures de l'entreprise sont–elles compatibles avec les recommandations du standard ? Que modifier pour que ce soit le cas ?
- Dans le cas où plusieurs normes sont utilisées conjointement, comment m'assurer de la cohérence de l'ensemble ?
- Comment consigner et réutiliser les acquis ? L'effort réalisé sur un projet est–il à refaire entièrement sur le suivant ? Comment valider et réutiliser des acquis ?

Toutes ces questions sont à expliciter et à résoudre selon une démarche globale allant des recommandations génériques à une pratique opérationnelle du développement d'un produit prenant en compte les contraintes de chaque métier particulier du projet et du profil de l'entreprise. C'est l'objet de notre travail. **Notre hypothèse est de ne pas remettre en question les standards d'ingénierie système tels que nous les avons présentés au début de ce chapitre.** Les processus identifiés résultent d'une expérience cumulée très importante. Nous considérons donc qu'ils sont valides et permettent, à condition de trouver et de se donner les moyens de les mettre en œuvre, de répondre à un grand nombre de problèmes d'ingénierie.

La démarche que nous défendons est celle de progresser encore dans la **formalisation des processus** de l'ingénierie système pour faire face à la complexité et permettre davantage encore de modélisation doublement ouverte sur l'estimation et la vérification. L'étape que nous allons d'abord franchir est celle de la formalisation des processus génériques. Au–delà, la difficulté principale que nous tenterons de résoudre est celle d'adapter ce processus générique à des problématiques particulières : prise en compte des métiers, spécificités des projets.

Nous pourrons revenir au terme de l'étude sur d'éventuelles insuffisances dans ces standards. A ce stade, nous considérons que les résultats que l'on peut attendre de l'étude sont :

– de faciliter les échanges,
– de mettre de la cohérence et de la vérification,
– de faire naître de nouveaux outils,
– de gagner du temps,
– de favoriser l'adoption de « bonnes pratiques ».

Dans cette thèse, nous nous intéressons donc à la définition des méthodes et des outils d'une démarche d'ingénierie système opérationnelle sur la base des recommandations normatives existantes. Nous voulons montrer que l'application des processus d'ingénierie système peut être facilitée par leur formalisation et proposerons une démarche d'application qui réponde aux questions pratiques évoquées ci–dessus.

Cette problématique répond à deux types d'enjeux.

D'une part, à des **enjeux industriels** liés à la maîtrise et la mise en œuvre des processus d'ingénierie qui conduisent à une maîtrise des objectifs des projets comme le coût et les délais du projet ou, la qualité du produit. Cela explique notre participation à une convention CIFRE avec le service SW de la société Airbus.

D'autre part, à des **enjeux académiques** qui sont autant de contributions à l'ingénierie système. Cela explique l'engagement, depuis plusieurs années, du laboratoire LATTIS (Laboratoire Toulousain de Technologie et d'Ingénierie des Systèmes), en collaboration avec le LAAS–CNRS, dans l'approfondissement d'une démarche de conception système.

1.7 Conclusion

Ce chapitre introductif a rappelé les enjeux de l'ingénierie système et retracé les évolutions qu'elle a lancées depuis sa création jusqu'aux recommandations et normalisations les plus récentes. Nous avons retenu la norme EIA-632 comme support de nos travaux. Cette norme a l'avantage de synthétiser les réflexions de plus de cinquante ans d'efforts scientifiques et d'expérimentations techniques et industrielles. L'étape que nous proposons est d'explorer la « personnalisation de ces normes » vers des modèles indispensables aux travaux d'optimisation, d'estimation et de vérification, dans des environnements complexes et pluridisciplinaires.

Cette étape se concentrera sur la formalisation d'un processus générique. On s'attardera dans les chapitres suivants sur les questions soulevées par l'adaptation pratique de ce proces-

sus d'ingénierie système générique à des domaines spécifiques : métier, projet.

Les chapitres suivants détailleront nos principales contributions. Ainsi, le chapitre 2 est dédié aux méthodes et outils de formalisation des processus d'ingénierie système que nous avons développés. Le chapitre 3, quant à lui, ira au–delà de la transcription des processus et donnera une démarche de spécialisation des processus. Enfin, le chapitre 4 complètera nos travaux en introduisant la vérification des processus construits jusqu'alors ainsi qu'une extension vers l'ingénierie système guidée par les modèles.

Chapitre 2

Formalisation des normes d'ingénierie système

« *Les enfants ont plus besoin de modèles que de critiques.* »

Joseph Joubert, Pensées

2.1 Introduction

Les normes d'ingénierie système se présentent sous la forme de recueils textuels. Si les infrastructures actuelles se sont bien adaptées au traitement de documents textuels, elles auraient de nombreux avantages à passer à une conception basée sur les modèles. Nous avons déjà vu, dans le chapitre précédent (tableau 1.1), les multiples désavantages des documents par rapport aux modèles : interprétation du texte, audit non automatisable, non reproductibilité des réponses, validations lourdes, traçabilité très difficile, réutilisation limitée, information difficile à trouver…

Outre les avantages immédiats dans l'application de l'ingénierie système que permettrait une transposition des textes normatifs décrivant des processus en modèles de processus, leur formalisation permet :

– de les spécialiser selon le contexte d'application, comme on le montrera dans le chapitre 3 de cette thèse,

– de faire le premier pas vers une ingénierie système guidée par les modèles ; le second pas étant l'harmonisation du traitement des modèles de système sur la base du modèle de processus.

La mise en place d'une démarche d'ingénierie système doit donc commencer par une reformulation des normes.

Un modèle est une abstraction de la réalité [Pel03] qui s'appuie sur un vocabulaire et des règles de représentation. Un modèle est une vue subjective mais pertinente de la réalité. Dans le cadre de la modélisation de systèmes informatiques, Booch [BRJ00] identifie quatre objectifs que permet d'atteindre la modélisation :

1. Les modèles aident à visualiser un système tel qu'il est ou tel que nous voudrions qu'il soit.
2. Les modèles permettent de préciser la structure ou le comportement d'un système.
3. Les modèles fournissent un canevas qui guide la construction d'un système.
4. Les modèles permettent de documenter le système.

Dans notre contexte, un modèle de norme d'ingénierie système est :
– une aide à l'élaboration et à la structuration des idées,
– un vecteur de communication entre personnes,
– la première étape d'un traitement informatique de la norme,
– un support au suivi et à la simulation de projet.

On voit bien qu'un modèle est plus qu'une simple représentation et que sa qualité va dépendre des traitements qu'il est amené à subir. C'est particulièrement vrai dans notre cas où les modèles sont spécialisés, *i.e.* dérivés, transformés à partir du modèle initial de la norme. Dans ce cadre, formaliser demande :
– de choisir un cadre théorique de formalisation (ingénierie des modèles),
– de sélectionner un langage de modélisation (SPEM/UML),
– de construire le modèle lui–même (application à l'EIA-632).

Dans ce chapitre, nous montrerons comment formaliser une norme d'ingénierie système ou, plus exactement, les processus qui les composent. Le chapitre est composé de trois sections. La première section présente le cadre théorique sur lequel se basera notre formalisation :

l'ingénierie des modèles. La seconde section présentera les langages existants de modélisation de systèmes et les langages existants de modélisation de processus qui seront les outils de construction de notre modèle de processus générique. Enfin, la dernière section montrera comment nous avons formalisé l'EIA-632 et sera consacrée à la présentation du modèle de ses processus.

2.2 L'ingénierie dirigée par les modèles

L'IS entretient depuis longtemps des liens avec l'ingénierie logicielle. Les particularités du développement logiciel ont poussé très tôt les informaticiens à se pencher sur la problématique du cycle de développement et de nombreuses similitudes existent entre les deux disciplines [Oli95].

Dans cette section, nous présenterons le paradigme d'ingénierie des modèles. Nouveau venu dans le domaine du logiciel, il commence à y être de plus en plus répandu et, comme d'autres concepts avant lui, il se voit peu à peu adapté pour répondre aux problématiques des systèmes. Dans nos travaux, il sera utile à la définition des concepts de « modèle », de « méta–modèle » ou encore de « transformation de modèles » largement utilisés par la suite.

2.2.1 Rôle de l'ingénierie des modèles

L'ingénierie dirigée par les modèles (IDM) propose des pistes pour permettre aux organismes de surmonter l'évolution des exigences du développement logiciel. En IDM, toute ou partie d'une application informatique est générée à partir de modèles.

Les modèles occupent une place de premier plan parmi les artéfacts de développement des systèmes et doivent en contrepartie être suffisamment précis afin de pouvoir être interprétés ou transformés par des machines. **Le processus de développement des systèmes peut alors être vu comme un ensemble de transformations de modèles partiellement ordonné, chaque transformation prenant des modèles en entrée et produisant des modèles en sortie, jusqu'à obtention d'artéfacts exécutables [Col06].** On retiendra la définition 13 ci–dessous pour l'ingénierie dirigée par les modèles.

Définition 13 : *Ingénierie Dirigée par les Modèles (IDM)*

L'Ingénierie dirigée par les modèles renvoie à l'utilisation systématique de modèles comme artéfacts de conception primaires dans le cycle de vie. L'IDM est un domaine de l'informatique qui met à disposition des outils, des concepts et des langages pour créer et transformer ces modèles.

Les principaux intérêts de l'IDM sont :
- la ré–ingénierie et l'évolution d'une application [GFD05],
- la réutilisation de modèles sur étagère,
- la conception par aspects [SGE+04],
- la modélisation par sujets [Cla02],
- la gestion de famille de produits (famille de modèles),
- l'interopérabilité,
- la portabilité.

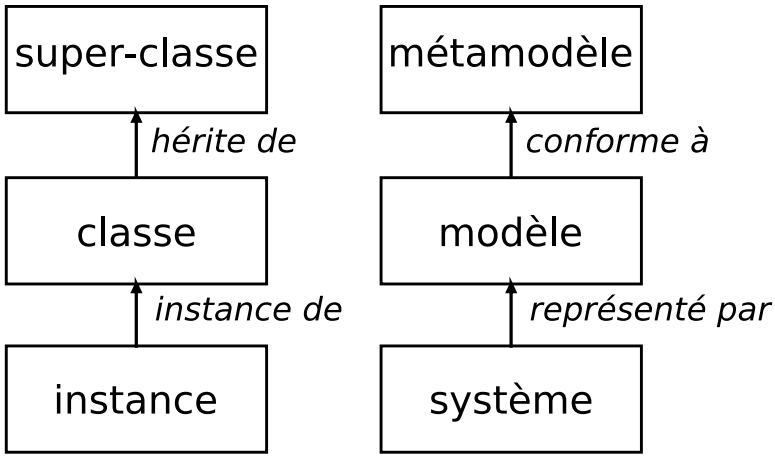
2.2.2 Principaux concepts de l'ingénierie des modèles

2.2.2.1 Notions de modèle et de méta–modèle

L'apparition du concept de modèle dans le domaine de l'informatique répond à une évolution naturelle. A l'origine fonctionnelle, la programmation est devenue « objet » pour faciliter l'intégration d'entités complexes. A son tour, le paradigme objet est complété par le concept de modèle [Béz05].

Dans le paradigme objet, les *instances* d'une classe *héritent* de ses propriétés, qui peuvent elles–mêmes avoir été héritées d'une super–classe (figure 2.1(a)). Ces notions ne sont cependant pas suffisantes pour répondre à tous les besoins de modélisation et il est nécessaire de les compléter. Pour cela, on va employer le paradigme d'ingénierie des modèles (ou MDE, Model Driven Engineering). Dans celui–ci, on considère qu'un système peut être modélisé par un modèle spécifique lui–même rédigé dans le langage de son modèle ou méta–modèle. Les relations entre éléments sont alors *représenté par* et *conforme à*.

L'avantage de disposer du cadre théorique du MDE est d'éviter de tomber dans le piège de la représentation au détriment du lien de conformité. L'erreur serait de considérer la représentation d'un objet comme son modèle alors que l'on ne peut pas statuer sur sa conformité.



(a) Notions de base en technologie « objet ».

(b) Notions de base en ingénierie des modèles.

FIG. 2.1: *Notions de base en technologie objet et ingénierie des modèles. En objet, les instances héritent des propriétés des classes mères. Un modèle permet de représenter un système tout en étant conforme à son méta–modèle.*

Les paradigmes objet et modèle ne sont pas incompatibles. Au contraire, ils se situent à des niveaux de représentations différents et vont se compléter.

2.2.2.2 Le méta–méta–modèle

L'intérêt essentiel des méta–modèles est de faciliter la fragmentation des représentations selon les préoccupations. Lorsque l'on considère un système donné, on peut travailler avec différentes vues de ce système, chacune de celles–ci étant caractérisée de façon précise par un méta–modèle donné. Quand plusieurs modèles différents ont été extraits du même système à l'aide de méta–modèles différents, ces modèles restent liés et pourront être recomposés par la suite. Pour que ceci puisse être largement appliqué, il est nécessaire de disposer d'une organisation régulière des modèles composites [Béz04].

Un postulat essentiel du MDE est de considérer que les méta–modèles sont des modèles et sont exprimés par un méta–modèle unique : le méta–méta–modèle. La conformité des méta–modèles au méta–méta–modèle permet de comparer, transformer, regrouper, trouver les différences, *etc.* entre des modèles exprimés sur la base de formalismes différents. Ce postulat essentiel permet d'éviter la fragmentation des modèles selon l'espace considéré.

Généralement, on utilise la notation suivante pour représenter les différents niveaux de modélisation :

M0 : pour décrire le niveau des systèmes réels,

M1 : correspond au niveau des modèles de systèmes,

M2 : pour le niveau des méta–modèles,

M3 : le méta–méta–modèle.

Cette hiérarchie se retrouve quels que soient les espaces technologiques considérés (Ecore dans le domaine technologique d'Eclipse, les grammaires, les schémas XML et le MDA de l'OMG).

Par exemple, dans le cas du langage XML, on a, au niveau M0, les données du système, au niveau M1 les données modélisées en XML, au niveau M2 les DTD XML et au niveau M3 le langage XML lui–même. La figure 2.2 donne un exemple d'organisation des modèles sur la base du langage XML.

2.2.2.3 Propositions de l'OMG

Les propositions de l'OMG se différencient de l'IDM selon deux axes.

Le premier d'entre eux est que l'OMG utilise comme méta–méta–modèle le MOF (Meta-Object Facility). L'originalité du MOF est qu'il établit une relation de conformité avec lui même. On dispose donc d'un méta–méta–modèle unique et auto–référent.

Le second axe de différence est l'existence, au niveau M2, d'une collection de méta–modèles standards reportée sur le tableau 2.1. L'OMG propose donc d'utiliser des méta–modèles pré–définis ; chacun spécialisé dans la représentation d'un type de modèle.

Sigle	Nom	Utilisation
UML	Unified Modeling Language	modélisation de logiciels
SysML	Systems Modeling Language	modélisation de systèmes
CWM	Common Warehouse Metamodel	modélisation de données
SPEM	Software Process Engineering Metamodel	modélisation de processus
QVT	Query/View/Transformation	transformation de modèles

TAB. 2.1: *Liste des langages proposés par l'OMG et de leurs utilisations.*

La figure 2.3 illustre les différents aspects couverts par les méta–modèles proposés par l'OMG.

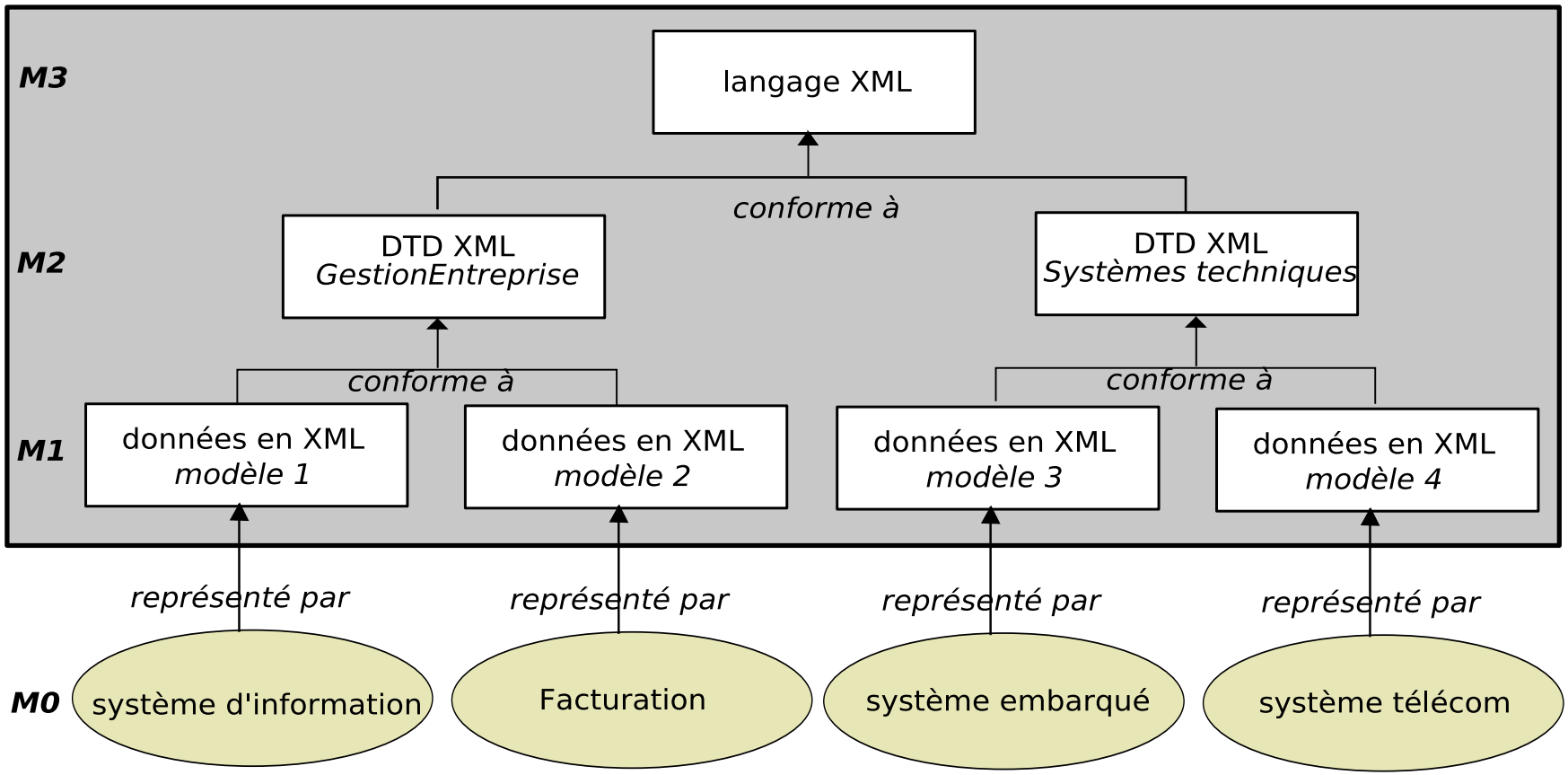


FIG. 2.2: *Exemple d'organisation des modèles XML. Chacun des systèmes réels est représenté par un modèle spécifique. Selon les cas, ces modèles peuvent être conformes à une DTD relevant de la représentation de gestion d'entreprise ou à une DTD pour les systèmes techniques. Chacune d'entre–elles est conforme au langage XML.*

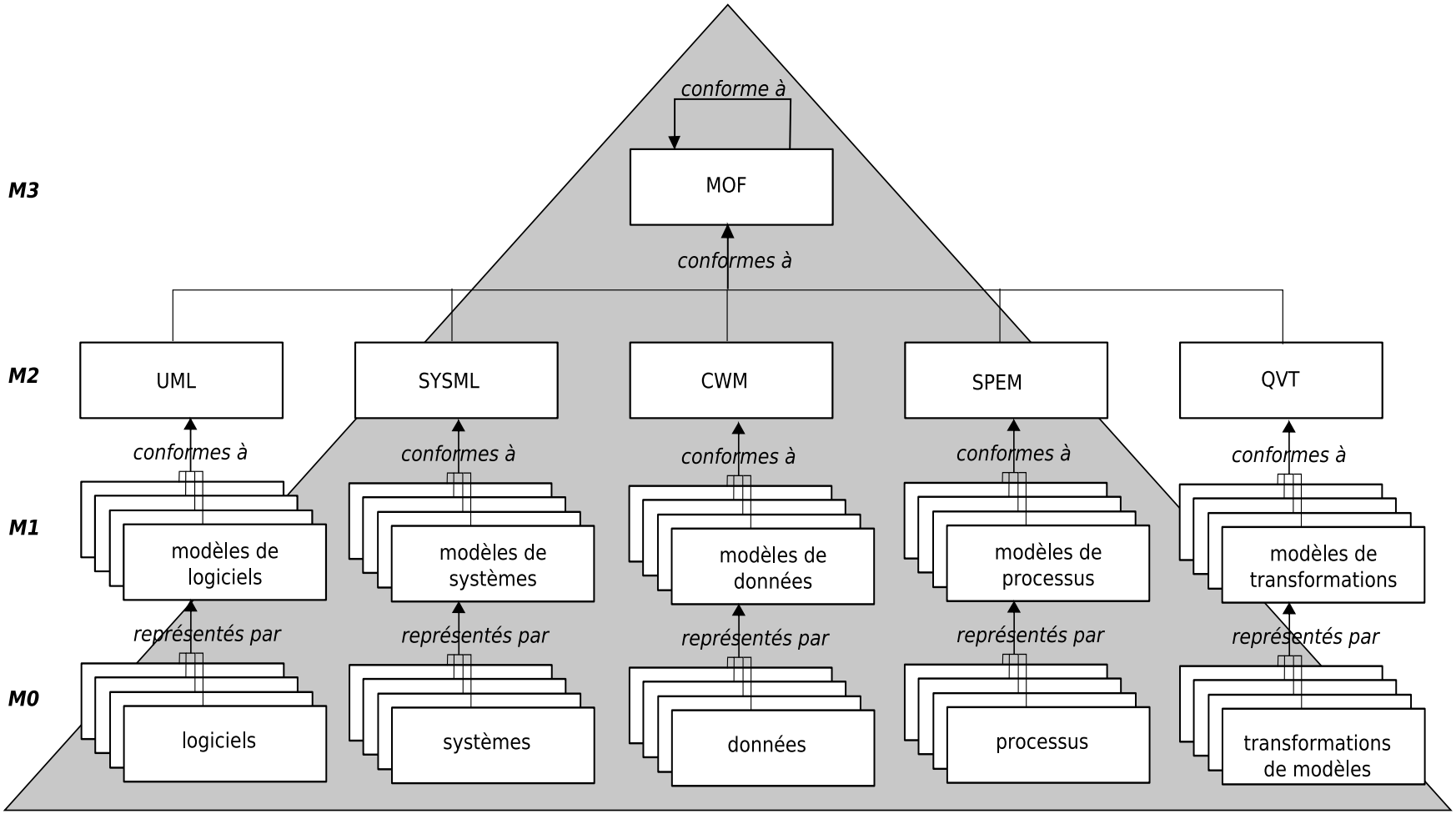


FIG. 2.3: *Organisation 3+1 de l'OMG. On retrouve une organisation en 4 niveaux. M0 pour les systèmes et M1, M2 et M3 pour leurs modèles, méta–modèles et méta–méta–modèle. Le MOF ou méta–méta–modèle est unique et auto–référent.*
Les méta–modèles, chacun dédié à la modélisation d'un domaine spécifique, sont définis et proposés par l'OMG.

2.2.3 Les langages de l'ingénierie des modèles

De nombreux langages ont été développés pour l'IDM. Ils peuvent être classés en quatre catégories :

les langages de méta–données : Ecore d'IBM, EMOF de l'OMG ou les schémas XML du W3C ;

les langages de transformation : QVT de l'OMG, ATL de l'INRIA, GreaT de l'Université de Vanderbilt, XSLT du W3C ;

les langages de requêtes : OCL de l'OMG ou XQUERY du W3C ;

les langages d'actions : Action Semantics de l'OMG, Xion de Objection.

On peut ajouter à cette classification Kermeta qui est à la fois un langage de méta–modélisation et un langage d'actions.

2.2.4 Les transformations de modèles

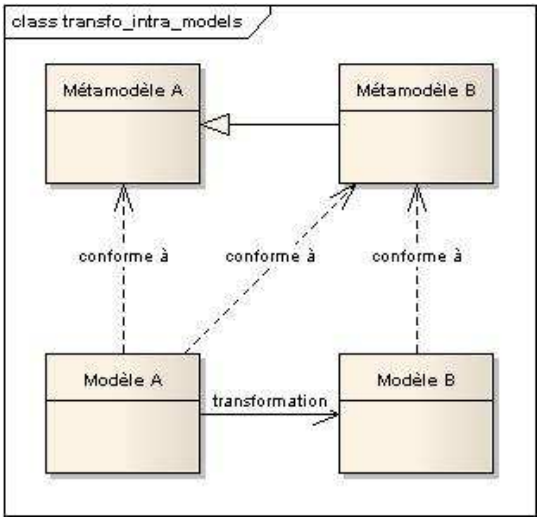
Dans la démarche MDE, les transformations sont au cœur du processus de conception. Le terme de transformation regroupe des opérations très variées. On distinguera ici deux types de transformation :

- les transformations endogènes (intra–modèle) ;
- et les transformations exogènes (extra–modèle).

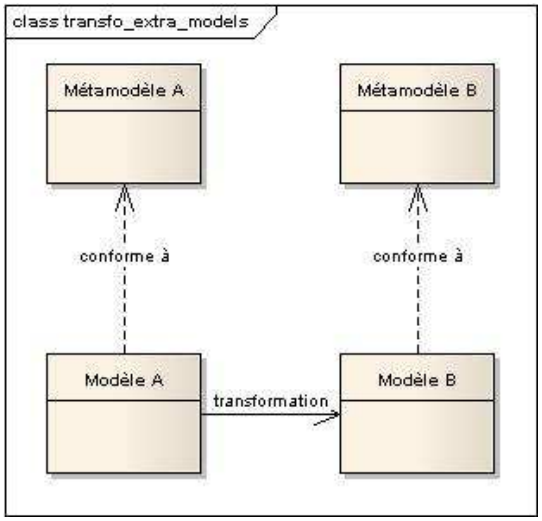
Lors d'une transformation intra–modèle (figure 2.4(a)) le modèle transformé restera dans le méta–modèle du modèle initial. Dans certains cas, il peut s'agir d'un sous–ensemble du méta–modèle initial.

Lors d'une transformation extra–modèle (figure 2.4(b)), les deux méta–modèles sont indépendants et l'on ne peut assurer ni qu'un modèle résultant de la transformation est effectivement conforme à son méta–modèle, ni qu'une opération inverse permette de retrouver le modèle initial à partir du modèle transformé.

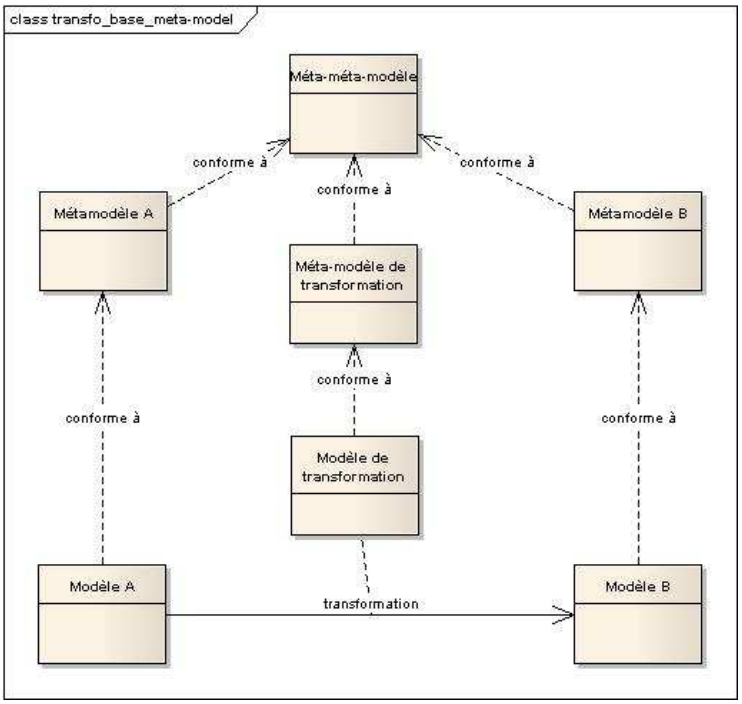
L'utilisation d'un méta–méta–modèle pourrait permettre de palier à ce problème en introduisant des propriétés communes entre les méta–modèles. Ce type de relation est particulièrement utilisé dans les transformations à base de méta–modèles qui utilisent des langages comme QVT ou ATL eux–même considérés comme des modèles (figure 2.4(c)). Ils permettent des transformations de modèles indépendamment du fait que les méta–modèles soient distincts ou non.



(a) Transformation intra–modèle



(b) Transformation extra–modèle



(c) Transformation basée sur les méta–modèles

FIG. 2.4: *Types de transformations de modèles. Représentation des transformations intra, extra–modèles et basées sur les méta–modèles. Une transformation basée sur les méta–modèles peut être intra ou extra–modèle.*

2.3 Choix d'un langage et d'un outil de modélisation

La seconde étape de la formalisation des normes d'ingénierie système consiste à choisir un langage de description sur la base duquel réaliser les modèles.

Dans cette section, nous présenterons rapidement les langages disponibles pour la modélisation des processus ainsi que ceux dédiés à la modélisation des systèmes. On verra, lors du choix du langage de modélisation qu'il est en effet indispensable d'intégrer des aspects structuraux complémentaires à la description des processus.

2.3.1 Les langages et les outils existants

2.3.1.1 Langages de modélisation de processus

La modélisation de procédure d'entreprise ou BPM[1] est un domaine très actif actuellement. Le nombre de langages de modélisation et la multitude d'outils existants suffisent à démontrer sa vivacité. Nous ne présenterons ici que les principaux langages de modélisation. On peut cependant se reporter, pour plus d'informations, à la thèse de Sonia Jamal [Jam05a] qui dresse un état de l'art complet de ces langages.

Parmi les langages de BPM, il convient de différentier modèles de workflow (OSSAD,BPMN ou SPEM) des langages de workflow comme BPML ou BPEL. Les premiers permettent une représentation, souvent graphique, des processus. Ils se concentrent sur la représentation des processus et leur communication. Les seconds sont des langages d'exécution des processus. Ils sont dédiés à l'exécution des processus par des moyens informatiques et ne disposent pas d'une représentation graphique qui facilite l'analyse ou la communication.

Dans ce contexte, XPDL (XML Process Definition Language) est un cas particulier. S'il n'est pas un langage d'exécution, la représentation XML n'est pas orientée non plus vers la modélisation. Son rôle est de servir de langage support entre la modélisation et l'exécution. Il est utilisé pour stocker les modèles dans un format intermédiaire.

Nous nous intéresserons ici plus particulièrement à la modélisation des workflows en présentant les principaux langages que sont OSSAD, UML, BPMN et SPEM.

[1] BPM est ici employé au sens de *Business Process Modeling*. Dans le reste de ce document, ce terme renvoie au pilotage des procédures d'entreprise ou *Business Process Management* et non plus seulement à leur modélisation.

OSSAD

OSSAD [DC90] est une méthode du domaine public dédiée à l'organisation pour analyser et modéliser des processus et procédures d'entreprise. Parue en 1989, c'est l'une des premières méthodes de modélisation de processus d'entreprise. L'objectif de cette méthode est de « mettre au point et diffuser une méthode originale d'analyse et de de conception de système de soutien du travail de bureau ou système bureautique ».

Elle propose l'utilisation de deux types de modèles[2] :

Modèle abstrait : « ce qui doit être fait et pourquoi ». Ce modèle donne les moyens pour représenter les caractéristiques propres et les frontières du système à étudier. Ces moyens sont, à la fois des concepts comme les fonctions, sous–fonctions, activités ou paquets d'information, mais aussi des graphes et matrices permettant de représenter graphiquement le système.

Modèle descriptif : « qui fait quoi et comment ». Ce modèle représente les choix passés et futurs concernant les personnes, les moyens techniques, l'organisation, la configuration spatiale et physique, *etc.* Ce modèle contient aussi des critères ou des éléments d'évaluation suffisants pour les prises de décision.

Innovante pour l'époque, cette méthode semble maintenant peu employée et semble faire la place à d'autres formalismes de modélisation de processus d'entreprise.

Le langage UML

Bien que développé dans un cadre plus large, l'utilisation d'UML est possible pour la modélisation de processus. Des 13 diagrammes UML, cinq peuvent être employés pour décrire des processus d'entreprise [MHLH05] :

le diagramme de communication : qui met en évidence les interactions entre objets ;

le diagramme de séquence : qui est une variante du diagramme de communication. Il permet de visualiser la séquence des messages entre objets ;

le diagramme d'états : qui associe des états aux objets et fait apparaître un ordonnancement entre ces états ;

le diagramme d'activité : qui est une variante du diagramme d'états ou les états sont des activités ;

[2] Cette décomposition sera reprise plus tard par la méthode Merise qui utilise, dans sa partie de description des processus, un niveau conceptuel (le « quoi ») et un niveau organisationnel (« qui et comment »).

le diagramme de cas d'utilisation : qui représente des interactions entre les acteurs et le système au travers de cas identifiés.

UML n'est pas spécifique à la modélisation de processus. En l'absence d'une sémantique claire, les diagrammes décrivant les processus peuvent être sujets à interprétation.

Le langage BPMN

BPMN (Business Process Model Notation) [OMG06a, RO03] est une notation graphique de représentation des processus métier. Récente (première parution en 2004 puis version finale en 2006), elle est portée par le BPMI (Business Process Management Initiative), un consortium des principales entreprises du BPM. Cette notation est portée par OASIS, OMG, W3C et WfMC.

Son objectif est double. D'une part, elle permet de représenter un processus métier découplé des informations techniques. D'autre part, elle fournit une correspondance vers les langages d'exécution en permettant de transcrire les modèles réalisés en BPML ou en BPEL4WS (aussi appelé BPEL).

Les éléments des diagrammes se décomposent en quatre catégories [Whi04] :

les objets de flux : les objets des processus. Il peuvent être des évènements, des activités ou des portes.

les objets de relation : les relations entre objets de flux représentent des flux de séquence, de messages ou des associations.

les couloirs (swimlanes) : ils permettent une organisation des objets en catégories. Ils permettent la représentation de fonctionnalités ou de responsabilités différentes dans un même processus.

les objets symboliques (artefacts) : les objets symboliques ajoutent de la flexibilité dans la représentation des processus. Ils permettent de lier des données aux activités, de les grouper ou de les annoter.

Le langage SPEM

SPEM, *Software Process Engineering Metamodel* est un métamodèle de processus de développement logiciel. Il fait partie des langages de modélisation proposés par l'OMG. Il fournit une définition des termes usuels et une description de la notion de composant de processus.

Ce métamodèle a connu deux versions : la version 1.1 [OMG05] de janvier 2005 et la version 2.0 [OMG07] adoptée le 3 mars 2007. Du fait de sa sortie tardive, la version 2.0 n'a pas été utilisée au cours de cette thèse. De nombreux ajouts et améliorations, présents dans la dernière version, auraient pourtant mérité d'être intégrés dans ces travaux. Nous traiterons donc uniquement avec SPEM 1.1.

SPEM a la particularité d'être défini à la fois comme un métamodèle à part entière et comme un profil UML.

De manière simplifiée, SPEM décrit un processus comme étant composé (figure 2.5) :

d'activités : les unités de travail. Elles précisent les opérations à appliquer sur les produits de travail ;

de produits de travail : les résultats ou les pré–requis des activités ;

de rôles : les responsabilités par rapport aux produits de travail et à la réalisation des activités.

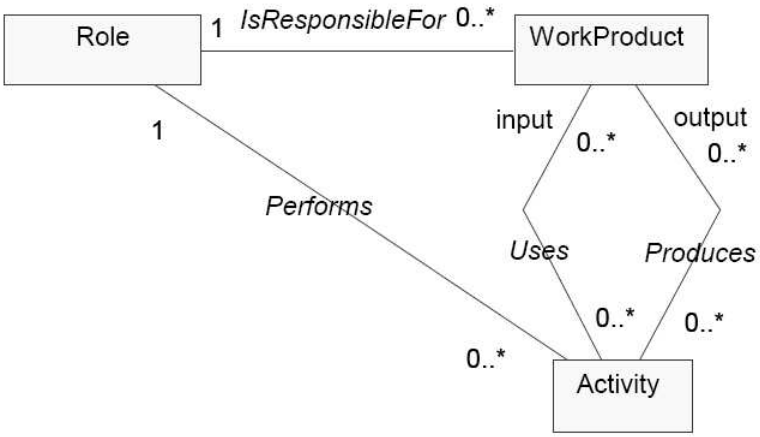


FIG. 2.5: *Modèle conceptuel de SPEM : activités, rôles et produits de travail. Les activités sont réalisées par un rôle unique et peuvent produire ou utiliser un nombre quelconque de produits de travail. Un même rôle peut réaliser plusieurs activités et être responsable de plusieurs produits de travail (figure 3-2 de [OMG05]).*

Une structure plus détaillée des éléments du métamodèle SPEM est donnée figure 2.6.

Outre la définition des éléments de base utilisés et la définition d'un processus, SPEM définit la notion de composant de processus. Il permet ainsi une décomposition des processus

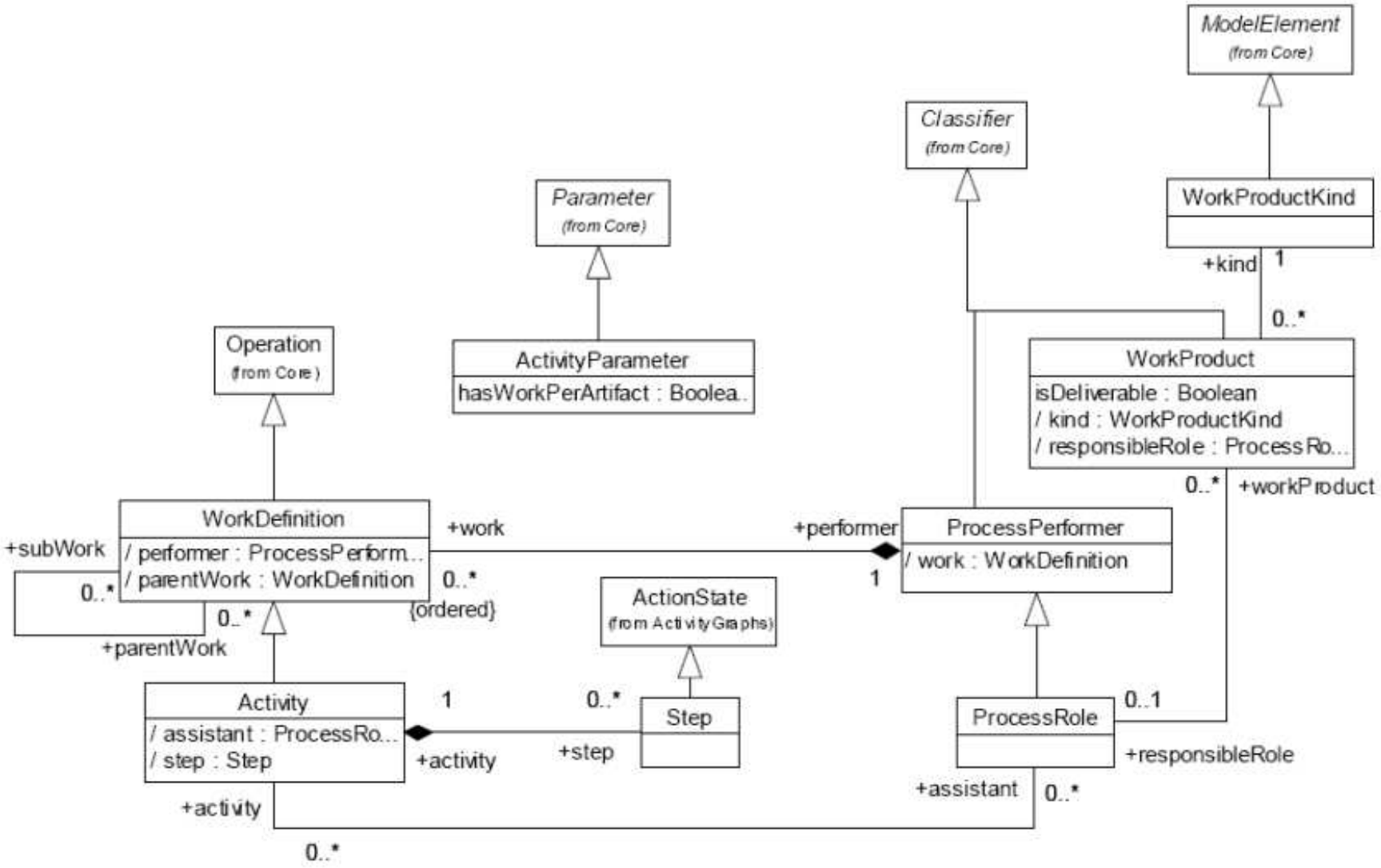


FIG. 2.6: *Structure de processus en SPEM. Une représentation simplifiée de ce schéma est donnée figure 2.5. En plus des éléments déjà présentés sur ce dernier, on peut remarquer que les définitions de travail, i.e. les activités peuvent comprendre des sous–travaux. On peut imbriquer des activités les unes dans les autres. Les activités peuvent aussi comprendre des étapes qui correspondent à la plus petite unité de description des processus (figure 7-1 de [OMG05]).*

en plusieurs autres. La notion de cycle de vie du processus vient compléter ces éléments et permet de venir encadrer les processus définis.

Les principales propriétés des langages de description de processus sont répertoriées au tableau 2.2.

	OSSAD	BPMN	UML	SPEM
Sémantique de processus	oui	oui	non	oui
Métamodèle	non	non	oui	oui
Concept de cycle de vie	non	non	non	oui
Supporté par des outils	oui	oui	oui	en tant que profil UML

TAB. 2.2: *Récapitulatif des propriétés des langages de description de processus*

2.3.1.2 Langages de description de systèmes

Conjointement aux évolutions de la représentation des processus d'entreprise, la représentation des systèmes a elle–aussi évolué.

Il est difficile de parler d'*un* langage de description de système, d'une part, car les représentations du système évoluent au cours du cycle de vie du système, et d'autre part, car les besoins de représentation sont multiples, quel que soit le moment considéré. D'après la typologie des modèles de l'AFIS[3], les modèles peuvent être :

- cognitifs ;
- normatifs (prescriptifs ou constructifs) ;
- prédictifs (formels ou analytiques) ;
- dédiés à la communication.

S'ajoutent à cela des besoins spécifiques de représentation ou des moyens de modélisation liés à la technologie employée.

Un système passe donc par une multitude de représentations au cours de son cycle de vie, et pour l'essentiel d'entre–elles, des DSL sont utilisés.

En fait, il ne s'agit pas ici de décrire les innombrables moyens de modéliser un système mais plutôt de s'intéresser aux langages de modélisation des systèmes selon le point de vue de l'ingénierie système.

[3] http://www.afis.fr/praout/modelisation/modelis2.html

Dans le domaine de la représentation des systèmes informatiques, UML s'est imposé comme référence. Son extension vers les systèmes a conduit l'OMG et l'INCOSE au développement de SysML.

Ces deux formalismes seront rapidement présentés ici.

Le langage UML

UML (Unified Modeling Language) est la référence en matière de modélisation logicielle. Ce langage, construit par l'association de trois langages antérieurs (Booch, OMT, OOSE), permet une représentation des concepts objets.

En normalisant les concepts objets, ce langage permet d'exprimer et d'élaborer des modèles objet, indépendamment de tout langage de programmation.

UML définit un système au travers de vues qui décrivent le système selon des considérations particulières appelées points de vue. La combinaison de toutes les vues est représentative du système complet. La vue « 4+1 » proposée par Kruchten [Kru95] est la plus couramment utilisée. Elle regroupe une vue des cas d'utilisation, une vue logique, une vue d'implémentation, une vue des processus et une vue de déploiement pour représenter l'architecture d'un système.

Treize diagrammes, dont la liste est donnée au tableau 2.3, permettent de décrire le contenu des vues. Ces diagrammes se décomposent en deux catégories : les diagrammes statiques et les diagrammes comportementaux. Un même diagramme peut appartenir simultanément à plusieurs vues.

diagrammes statiques	**diagrammes comportementaux (ou dynamiques)**
diagramme de classes	diagramme des cas d'utilisation
diagramme d'objets	diagramme états-transitions
diagramme de composants	diagramme d'activité
diagramme de déploiement	diagramme de séquence
diagramme de paquetages	diagramme de communication
diagramme de structure composite	diagramme global d'interaction
	diagramme de temps

TAB. 2.3: *Liste des diagrammes UML.*

UML n'étant pas une méthode, il est nécessaire de lui associer des processus d'élaboration des modèles tel que UP ou ses dérivés (RUP, EUP, XUP, AUP, 2TUP, EssUP...) qui précisent quels diagrammes employer et à quelles étapes du développement.

Le langage SysML

SysML [OMG06b] est une extension d'UML à la représentation de systèmes. Le succès rencontré par UML a conduit à envisager son utilisation en ingénierie système. Le langage était cependant peu adapté à la représentation de systèmes complexes et a poussé l'OMG et l'INCOSE à repenser un langage plus adapté.

SysML permet la spécification, l'analyse, la conception, la vérification et la validation de systèmes. Il est défini comme une extension d'un sous–ensemble d'UML.

Par rapport à UML, SysML offre les avantages suivants pour la spécification de systèmes [Par07] :

- La sémantique, proche du domaine de l'ingénierie système, est plus adaptée que celle proposée par UML.
- SysML est plus réduit et donc plus facile à apprendre.
- Les tables d'allocation introduites par SysML facilitent la validation et la vérification et l'analyse de couverture.

Étonnamment, un autre avantage listé par [Par07] est la conformité des modèles SysML à l'IEEE-Std-1471-2000 (IEEE Recommended Practice for Architectural Description of Software-Intensive Systems), norme orientée logiciel et non pas système.

Concrètement, les différences entre SysML et UML viennent des diagrammes que ces langages proposent. Sept des treize diagramme UML sont utilisés, deux nouveaux diagrammes sont introduits et les tables d'allocation dérivées des diagrammes SysML apparaissent. Les diagrammes respectifs de UML et SysML sont représentés au tableau 2.4.

En formalisant les exigences et la décomposition du système, en introduisant une vision fonctionnelle, en permettant l'expression du continu et la V&V, SysML permet une description des systèmes en accord avec la vision de l'ingénierie système.

Au même titre que pour UML, l'aspect méthodologique reste ouvert. Le formalisme SysML doit être accompagné de processus qui viendront préciser quels modèles utiliser et à quelles étapes du cycle de vie du système.

diagrammes SysML	Objectifs	diagrammes UML
Diagramme d'activité	Montre l'environnement du système comme des flots de données et de contrôle.	Diagramme d'activité
Diagramme de définition de bloc	Montre la structure du système comme des composants avec des propriétés, opérations et relations.	Diagramme de classes
Diagramme de bloc interne	Montre la structure interne des composants. Inclut leurs parties et connecteurs.	Diagramme de structure composite
Diagramme de paquetages	Montre l'organisation du modèle en paquetages, vues et points de vue.	Diagramme de paquetages
Diagramme paramétrique	Montre les contraintes paramétriques entre éléments structuraux.	⊘
Diagramme d'exigences	Montre les exigences du système et leurs relations avec les autres éléments.	⊘
Diagramme de séquence	Montre l'environnement du système comme des interactions entre composants du système.	Diagramme de séquence
Diagramme de machine à états	Montre l'environnement du système comme une succession d'états qu'un composant ou qu'une interaction prendra en réponse à des évènements.	Diagramme de machine à états
Diagramme des cas d'utilisation	Montre les exigences fonctionnelles du système comme des interactions avec les utilisateurs.	Diagramme des cas d'utilisation
Tables d'allocation	Montre différents types d'allocation (*e.g.* allocation d'exigences, allocation fonctionnelle, allocation structurelle).	⊘
⊘		Diagramme de composants
⊘		Diagramme de communication
⊘		Diagramme de déploiement
⊘		Diagramme global d'interaction
⊘		Diagramme d'objets
⊘		Diagramme de temps

TAB. 2.4: *Comparaison des diagrammes SysML et UML : Lorsque l'un des diagrammes n'existe pas dans un langage, il est marqué par le symbole ⊘.*

2.3.2 Le langage de modélisation utilisé

Un modèle de norme d'IS est une abstraction de la norme. Il doit permettre d'appréhender rapidement les concepts phares portés par la norme sans pour autant les dénaturer. Deux idées principales sont communes aux différentes normes. La première est qu'il existe des processus génériques de référence pour l'ingénierie d'un produit. La seconde est que les processus qui conduisent à la définition, à la réalisation et à la diffusion du système sont dépendants du cycle de vie du produit.

On constate une dualité entre les concepts de système et de processus vus par l'ingénierie système. Si l'application des processus permet de faire évoluer le système, tout au moins dans sa représentation, le système, et notamment sa décomposition structurelle, vont influer les processus. Par exemple, dans le cas de l'EIA-632 le système est d'abord perçu comme une liste d'exigences utilisateur, puis comme des spécifications techniques avant de devenir une solution logique, une solution physique et finalement le système lui–même. Les choix de conception conduisent à un découpage du système en sous–systèmes et chaque sous–système sera ensuite développé selon ses propres processus.

La modélisation des normes d'ingénierie système doit donc passer par :

- une représentation des processus recommandés par la norme ;
- une représentation de l'état de développement du système ;
- une représentation des interactions entre processus et système organisée autour de son cycle de vie.

Il est important d'avoir à l'esprit que l'on ne cherche pas à disposer d'un modèle du système orienté conception comme c'est généralement le cas. Les éléments système que nous cherchons à représenter doivent permettre de suivre les évolutions du système dans ses représentations et sa structure sans pour autant entrer dans une description technique. Par exemple, du point de vue du suivi de projet, il importe de savoir qu'une solution logique du système est définie. En revanche, les modèles de solution logique qui vont la décrire d'un point de vue comportemental (*e.g.* cas d'utilisation SysML) ne sont pas nécessaires.

Nous avons choisi comme langage de modélisation des normes d'ingénierie système le langage UML complété par des profils SPEM. Les raisons principales de ce choix sont la conservation d'une sémantique dédiée pour chacun des deux aspects du modèle et la facilité de représentation des interactions système/processus.

Les stéréotypes SPEM viennent compléter les concepts objets d'UML en introduisant les notions d'activité, de rôles, de produit de travail, de processus et de cycle de vie essentiels à la représentation des processus d'IS. La sémantique des composants des processus est donc définie par SPEM. La solution que nous avons choisie permet de travailler sur un modèle consistant dans lequel les éléments système et processus sont liés.

Le choix d'UML plutôt que de SysML est lié à une raison pratique. Il n'existe pas de profil SPEM pour SysML. L'utilisation d'UML n'est cependant pas pénalisante car, comme on l'a vu précédemment, on ne cherche pas un modèle exhaustif du système mais un modèle utile au suivi de l'état du système dans son cycle de vie.

2.4 Application à la modélisation de l'EIA-632

Nous présenterons dans cette section un exemple de modélisation d'une norme d'IS. Nous avons choisi de travailler sur l'EIA-632 car elle offre une bonne couverture du cycle de vie des systèmes. Plus étendue que l'IEEE 1220 elle est indépendante contrairement à l'ISO 15288 qui doit être complétée.

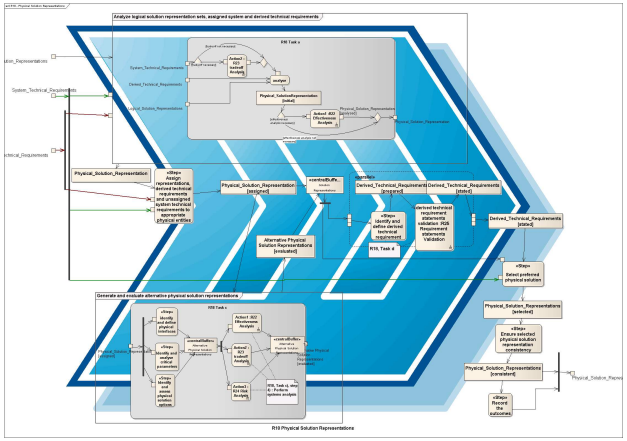
Le modèle qui sera présenté a été réalisé avec Enterprise Architect de Sparx systems. Dans sa dernière version, ce modèle inclut 69 paquetages pour 79 diagrammes et 1534 éléments. Ce modèle est laissé en libre accès à l'adresse ci–dessous :

`http://www.lesia.insa-toulouse.fr/~rochet/`.

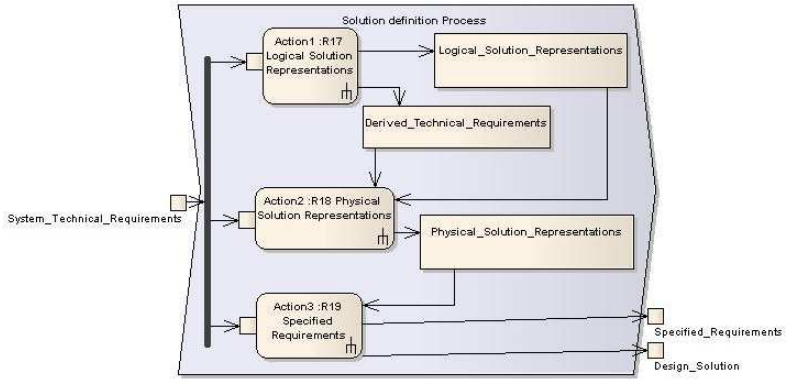
Nous verrons ici comment ce modèle a pu être construit et quelles analyses peuvent être conduites à partir de ce modèle.

2.4.1 Modélisation des processus de l'EIA-632

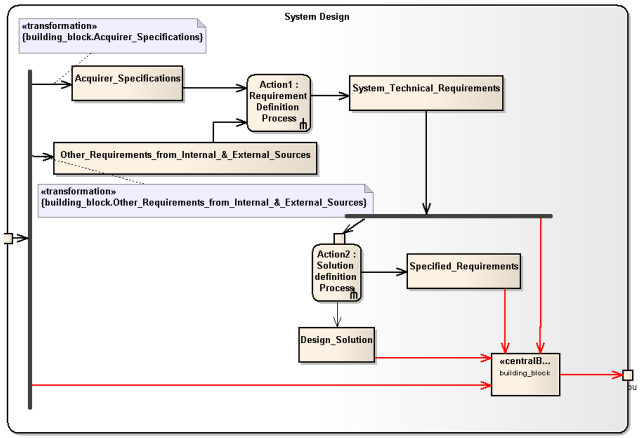
La transcription des processus de l'EIA-632 dans le modèle se fait en intégrant dans un modèle en SPEM des flux de travail déduits de l'EIA-632. En associant la décomposition hiérarchique des processus du chapitre 4 de l'EIA-632 avec les résultats attendus des tâches de l'annexe C de l'EIA-632, une représentation en flux des processus est construite. Chaque groupe, processus ou exigence est représenté par un diagramme d'activité dont les actions sont soit des tâches, soit des actions composites invoquant des sous–processus. L'emploi de stéréotypes SPEM, représentés au tableau 2.5, et l'organisation des diagrammes d'activité dans une arborescence similaire à la décomposition hiérarchique de l'EIA-632 précise la sémantique de ce modèle. Des exemples de diagrammes d'activité des processus de l'EIA-632 sont donnés sur la figure 2.7.



(a) Diagramme d'activité de l'exigence de représentation physique de la solution. Une vue détaillée de ce diagramme est donnée aux figures 6 et 7 en annexes.



(b) Diagramme d'activité du processus de définition de la solution



(c) Diagramme d'activité du groupe de conception système

FIG. 2.7: *Modèles de processus de l'EIA-632. Trois exemples de processus sont donnés, un pour chaque étage de la décomposition de l'EIA-632 en groupes, processus et exigences. Dans cet exemple, on remarque l'invocation des processus de niveau inférieur par des actions composites. Les objets représentent les résultats attendus des processus.*

EIA-632	Type UML	Stéréotype SPEM
Groupe	Activité	Discipline
Processus	Activité	Process
Exigence	Activité	Activity
Tâche	Action	Step

TAB. 2.5: *Correspondances des éléments de processus de l'EIA-632 en UML et en profils SPEM.*

2.4.2 Modélisation de la structure de système

La définition d'un système selon l'EIA-632 passe par la description :

du concept de système,

du concept de bloc de construction, constitué :

1. d'un système ;
2. de produits finaux ;
3. de sous–systèmes ;
4. de produits contributeurs.

du concept de structure de système.

Tous ces éléments sont intégrés dans le modèle par des diagrammes d'objets.

Dans le cadre d'une description des concepts liés à la structure de système, on pourrait se contenter de les représenter comme des objets. Ces éléments sont cependant issus des processus de l'EIA-632 et sont à considérer comme des produits de travail des processus. Pour représenter cela, le stéréotype SPEM *work product* est associé aux éléments qui sont des résultats attendus de l'exécution des processus de l'EIA-632.

Ces concepts seront repris brièvement ci–dessous. Les figures 2.8 et 2.9 montrent deux exemples de diagrammes d'objets de modélisation de la structure de système.

2.4.2.1 Modèle d'un système

Le système sur lequel s'appliquent les processus de l'EIA-632 se compose d'un ou de plusieurs produits finaux qui seront utilisés par l'acquéreur dans un but donné et de produits contributeurs qui rendent possible la création, la réalisation et l'utilisation du ou des produits

finaux. Les produits contributeurs sont utilisés pour réaliser les fonctions du système associées aux processus (développer, produire, tester, déployer et supporter les produits finaux), pour entraîner les opérateurs et l'équipe de maintenance des produits finaux et pour retirer ou pour mettre à disposition les produits.

Conformément à l'EIA-632, les produits contributeurs ne sont pas liés à un produit final mais à une fonction associée au système.

2.4.2.2 Modèle des blocs de construction

Le concept de système forme la base d'une structure plus large : le bloc de construction. Ce concept sert de cadre de référence à l'application des processus de l'EIA-632.

Il est utilisé pour affiner la décomposition du système en précisant les notions de produit final et de produit contributeur et en associant des rôles aux blocs de construction dans le cadre de leur utilisation dans les processus de l'EIA-632.

2.4.2.3 Modèle de la structure de système

Un seul bloc de construction est rarement suffisant pour décrire complètement la solution correspondant aux exigences du client et des autres parties intéressées. Un produit peut demander un développement plus poussé que celui qui lui est attribué dans le cadre d'un bloc de construction unique.

Notre modèle considère deux types de produit final[4] :

Composite_End_Product : Un produit final constitué d'au moins deux sous–systèmes (source de blocs de construction de couche inférieur dans la structure du système).

Implemented/Existing/Acquired_End_Product : Un produit final non décomposable qui :

1. sera implémenté en tant que produit final unitaire ;
2. dont les exigences sont satisfaites par l'utilisation d'un produit existant ;
3. ou, que le produit final soit fourni.

[4] L'EIA-632 ne fait pas de distinction explicite entre ces deux types de produits finaux. Dans la définition d'un bloc de construction donnée page 47 de la norme, un produit final *est* constitué de deux sous–systèmes ou plus. Cette définition est en contradiction avec le chapitre 6.2. dans lequel un nouveau bloc de construction est associé à chaque sous–système à moins que le produit final ne puisse être implémenté, ne soit existant ou ne puisse être acquis auprès de fournisseurs. Considérer uniquement des produits finaux composites entraînerait une infinité de blocs de construction dans la structure de système. C'est pourquoi deux types de produits finaux sont introduits dans le modèle en contradiction apparente avec la définition donnée des blocs de construction.

Pour permettre le développement des produits finaux composites, un bloc de construction de niveau inférieur est attribué à chacun de ses sous–systèmes. Du point de vue d'un de ces blocs de construction, le sous–système concerné sera considéré comme un système principal sur lequel seront appliqués les processus de l'EIA-632 et seront associés des produits finaux, sous–systèmes et produits contributeurs spécifiques.

En appliquant les processus au cours du cycle de vie, cette décomposition se poursuit :

1. jusqu'à ce que les produits finaux d'un bloc de construction puissent être implémentés ;
2. jusqu'à ce que les exigences d'un produit final puissent être satisfaites par un produit existant ;
3. ou, jusqu'à ce que les produits finaux puissent être acquis auprès d'un fournisseur.

Les exigences spécifiées pour un sous–système deviennent les exigences assignées à la couche inférieure de développement. De plus, chaque bloc de construction peut avoir d'autres exigences de parties intéressées qui ne sont pas relatives aux exigences de la couche supérieure.

Le modèle de la figure 2.8, dérivé de la structure d'un bloc de construction (figure 1.9), intègre la notion de structure de système. Ce modèle illustre que les sous–systèmes d'un produit final composite seront développés au sein d'un bloc de construction. Une généralisation des sous–systèmes en système est réalisée pour permettre que ces sous–systèmes soient considérés comme des systèmes à part entière au sein des blocs de construction qui leur seront associés.

La modélisation de la structure de système est la base de représentation de nombreux mécanismes de l'EIA-632. Les deux plus importants sont sans soute le traitement des exigences et le développement de produits contributeurs.

Exigences et structure de système

Les différents types d'exigences et leurs relations au sein d'un bloc de construction sont donnés par le modèle de la figure 2.9. Ce dernier reprend les relations exprimées sur le schéma G.1 en annexe de l'EIA-632 qui trace les différents stades au cours desquels les exigences du client et des parties intéressées vont être transformées en spécifications de solution de conception.

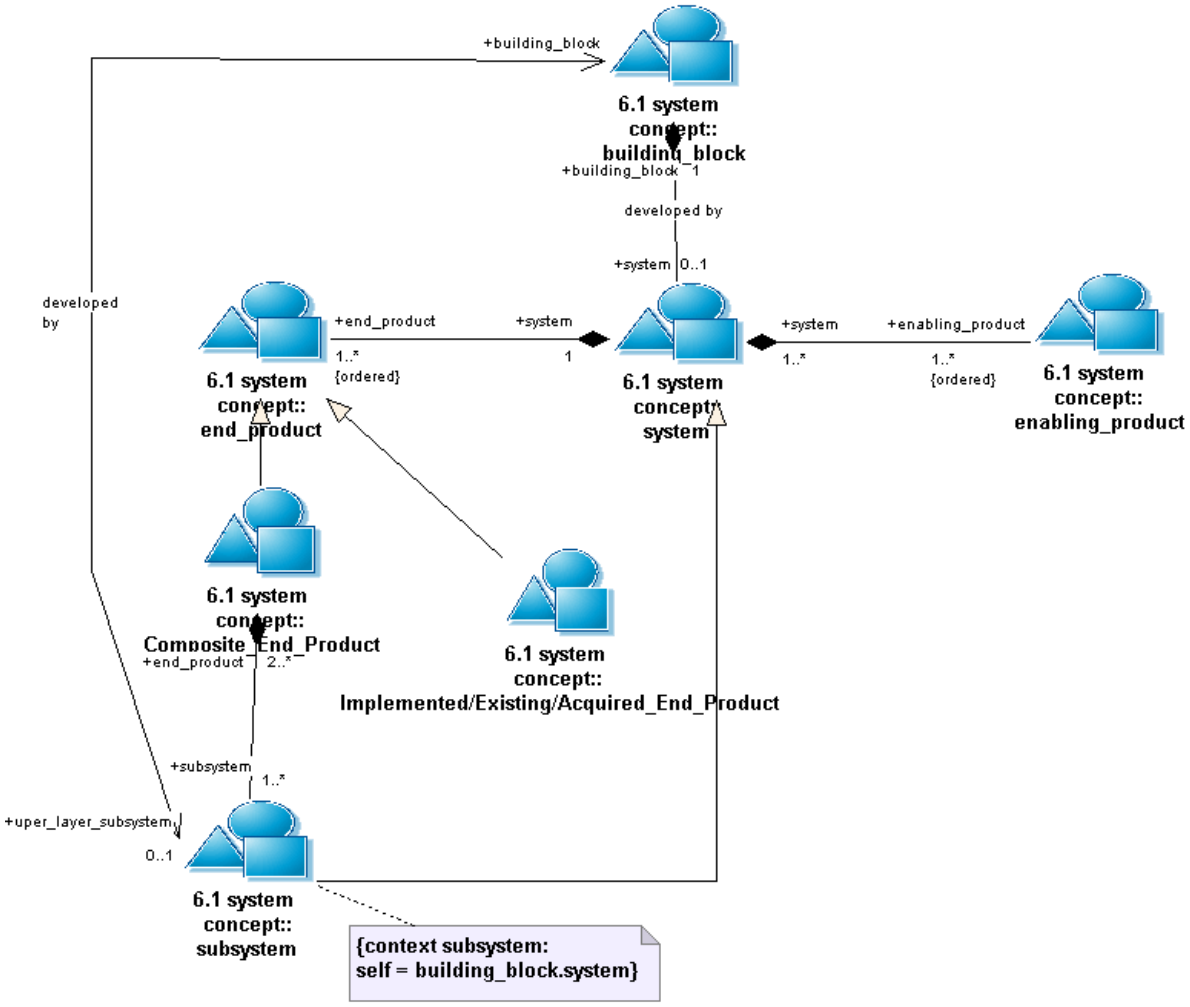


FIG. 2.8: *Concept de structure du système. Un sous–système est un type de système dont le développement est associé à un bloc de construction. Le système de ce bloc de construction est identique au sous–système considéré.*

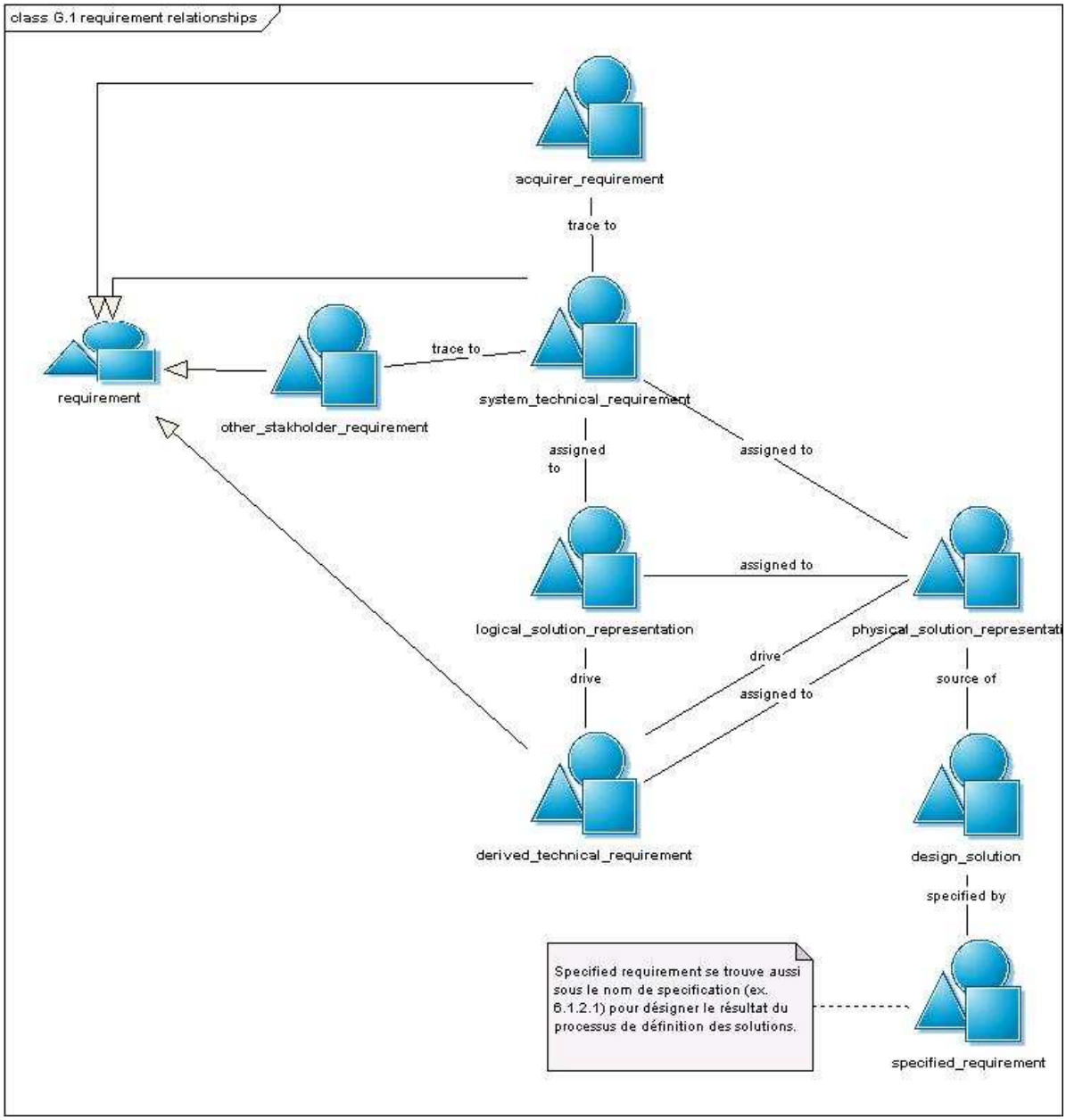


FIG. 2.9: *Relations entre les exigences. Représentation synthétique des relations entre exigences. Le passage d'un type d'exigence à un autre sera dicté par les modèles des processus de conception système.*

Bloc de construction et structure de système

La structuration en couches des blocs de construction ne se limite pas aux sous–systèmes. Lorsqu'un ensemble de produits contributeurs nécessite d'être développé selon les processus de l'EIA-632 un autre bloc de construction est formé. Le développement d'un bloc de construction de produit contributeur est généralement initié après que les produits finaux auxquels ils se rattachent aient été complètement définis et après que les exigences pour les produits contributeurs aient été identifiées. Un bloc de construction est alors formé dans lequel on considère le produit contributeur comme un système de développement, de production, de test, de déploiement, de formation, de support ou de mise à disposition.

2.4.3 Modélisation du cycle de vie d'ingénierie

L'association des blocs de construction aux processus de l'EIA-632 n'a pas jusqu'à présent été abordée. De la même manière, la synchronisation des processus menant à un développement descendant suivi d'une réalisation ascendante n'est pas rendue explicite.

De manière simplifiée, on dispose actuellement d'un modèle composé de deux ensembles :

- un modèle de description des processus de l'EIA-632 ;
- un modèle formalisant les concepts liés à l'EIA-632 regroupant les notions de système, bloc de construction et structure de système.

Ces deux ensembles sont, pour l'instant, disjoints. L'introduction du modèle de cycle de vie d'ingénierie du chapitre suivant va permettre de mettre en avant les points de rencontre entre ces ensembles et d'intégrer dans le modèle les aspects de synchronisation des processus.

D'après l'EIA-632, les phases du cycle de vie d'un produit sont organisées de la manière suivante :

phase de conception :

- définition du pré–système ;

phases de création :

- définition du système ;
- conception des sous–systèmes ;
- conception détaillée ;

phase de réalisation :

- intégration, test et évaluation du produit final.

Le tableau 2.6 ci–dessous donne la description de chacune des activités liées aux phases. Il précise également les processus actifs au cours de chacune des phases.

Phase	Description des activités
Définition du pré–système	La phase de départ du cycle de vie. Les processus de *gestion technique*, si possible, sont employés pour prévoir l'effort technique ou pour affiner l'effort technique décrit par les plans existants. Les processus de *conception système* sont appliqués, si possible, au *building block de la couche supérieure* du projet pour déterminer les meilleurs concepts de système satisfaisant aux exigences de l'acquéreur ou pour affiner un concept précédemment sélectionné.
Définition du système	Les processus de *conception système* et, si possible, les processus de *gestion technique* et d'*évaluation technique* sont appliqués au *building block de la couche supérieure* d'un projet pour établir les exigences spécifiées des produits finaux et pour définir les spécifications initiales incluant les spécifications d'interface pour les sous–systèmes de chaque produit final, et pour identifier les exigences des produits contributeurs. Les risques techniques sont limités durant cette phase. Avant le passage à la phase de conception des sous–systèmes, les revues techniques incrémentales appropriées et une revue de définition de système sont réalisées.
Conception des sous–systèmes	Les processus de *définition du système* et les processus de *gestion technique* et d'*évaluation technique* appropriés sont appliqués aux *building blocks de la seconde couche* du projet pour établir les exigences spécifiées des produits finaux et pour définir les spécifications initiales incluant les spécifications d'interface pour les sous–systèmes de chaque produit final qui demandent un développement plus poussé, et pour identifier les exigences des produits contributeurs. Les risques techniques pour les sous–systèmes, les produits finaux et les produits contributeurs sont évités durant cette phase. Avant le passage à la phase de conception détaillée, les revues techniques incrémentales appropriées et une revue de définition de système sont réalisées.
Conception détaillée	Les processus de *définition du système* ainsi que les processus de *gestion technique* et d'*évaluation technique* appropriés sont appliqués aux *building blocks de troisième couche ou inférieurs* du projet pour établir les exigences spécifiées et les dessins de définition ou documents des produits finaux et pour définir les spécifications initiales, incluant les spécifications d'interface pour les sous–systèmes de chaque produit final qui demandent un développement plus poussé, et pour identifier les exigences des produits contributeurs. Les risques techniques pour les produits finaux et produits contributeurs *des couches inférieures* sont évités durant cette phase. Avant le passage à l'effort de conception détaillée de niveau inférieur, les revues techniques incrémentales appropriées et une revue de définition de système sont réalisées pour les éléments de building blocks concernés. Lorsque la conception d'un produit final peut être faite par l'achat, la construction ou la réutilisation, le développement de ce produit final est terminé. Un préalable au passage à la phase suivante du cycle de vie du produit est que les revues techniques de préparation du test et de préparation de la production soient complétées.
…	…

Phase	Description des activités
Intégration, test et évaluation du produit final	Les produits finaux sont obtenus à partir des fournisseurs, des acquéreurs (dans le cas d'éléments fournis par le client), prêts à l'emploi ou fabriqués sur la base des spécification, documents ou dessins de conception détaillée. Les processus d'*implémentation*, de *gestion technique* et d'*évaluation technique* sont appliqués pour valider les produits finaux obtenus, pour assembler ou intégrer les produits finaux validés et pour vérifier que les éléments composés des produits finaux satisfont les exigences spécifiées. Le processus de *transition vers l'utilisation* est appliqué pour délivrer les produits finaux vérifiés à l'acquéreur de la couche supérieure en conformité avec les accords passés. Ensuite, les processus d'*implantation*, de *management technique*, d'*évaluation technique* et de *transition vers l'utilisation* sont appliqués, si nécessaire, aux *building blocks de couches supérieures* successifs jusqu'à la livraison des produits finaux et des produits contributeurs requis dans les accords établis pour le projet.

TAB. 2.6: *Description des activités liées à chacune des phases du cycle de vie des produits. Tableau extrait de la table 6.3 de l'EIA-632.*

L'organisation des processus et les interactions qu'ils rencontrent avec la structure du système sont représentées comme des activités appliquées aux blocs de construction. Le modèle de la figure 2.10 représente le cycle de vie d'un système. Les figures 1, 2, 3, 4 et 5 en annexe donnent chacune la vue détaillée de l'une des phases du cycle de vie.

2.4.4 Analyse de l'EIA-632 à partir de son modèle

La formalisation de l'EIA-632 a permis une meilleure connaissance des mécanismes induits par ses recommandations. L'écriture du modèle a mis en évidence des comportements annoncés mais non–explicites ou des comportements implicites. On peut citer :

- un comportement global de type descendant puis ascendant,
- des comportements différents pour des groupes ou des processus de même niveau,
- un séquencement des activités reposant sur le processus de fourniture,
- un flot de données favorisant la cohérence des éléments des blocs de construction.

2.4.4.1 Développement descendant, réalisation ascendante

On voit sur les modèles des phases de cycle de vie du système la construction successive des différentes couches de blocs de construction. Les deux premières phases de définition du pré–système et de définition du système s'attachent à définir le bloc de construction de la couche supérieure du projet. La conception des sous-systèmes va ensuite permettre la définition des blocs de construction de la seconde couche associée aux sous–systèmes et aux pro-

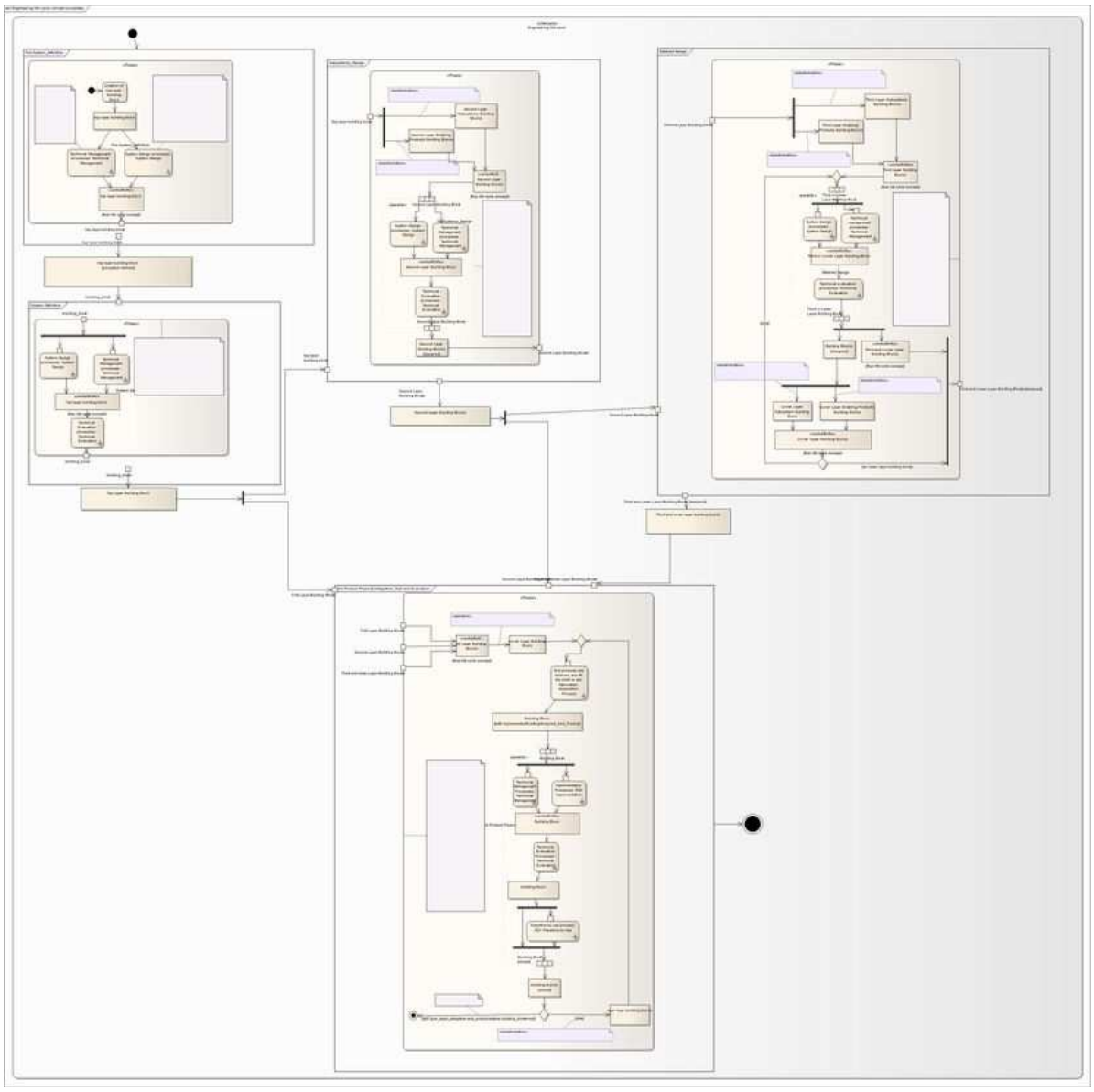


FIG. 2.10: *Concept de cycle de vie d'ingénierie. Modèle complet du cycle de vie selon l'EIA-632. Les représentations détaillées de chacune des phases : figure 1 pour la phase de pré–définition du système, figure 2 pour la phase de définition du système, figure 3 pour la phase de conception des sous–systèmes, figure 4 pour la phase de conception détaillée et figure 5 pour la phase d'intégration, test et évaluation du produit final sont données en annexes.*

duits contributeurs. Ce procédé de définition des blocs de construction est répété de manière récursive à chaque bloc de construction au cours de la phase de conception détaillée jusqu'à ce que tous les produits contributeurs et tous les produits finaux des couches les plus basses soient considérés comme unitaires. Ces quatre phases coordonnent les activités de développement descendantes. L'aspect ascendant de la réalisation est entièrement réalisé par la dernière phase d'intégration, test et évaluation du produit final (figure 5 en annexe). Les produits des couches inférieures sont validés puis associés pour être livrés aux blocs de construction de la couche supérieure. Les produits livrés sont alors re–validés et ré–associés jusqu'à satisfaction des exigences du bloc de construction de la couche supérieure.

Le cycle de vie est représenté de manière synthétique à la figure 2.11.

2.4.4.2 Types de groupes

Les processus de l'EIA-632 appliqués aux blocs de construction varient donc en fonction du cycle de vie d'ingénierie. La liste des processus invoqués en fonction des cycles de vie est donnée au tableau 2.7.

A partir de ce tableau, on peut dissocier les groupes de l'EIA-632 en : groupes de type « processus », groupes de type « paquetage » et groupe d'acquisition et de fourniture.

groupe de type « processus » : Tous les processus appartenant à ces groupes sont invoqués au même moment dans les phases du cycle de vie. L'ordre des processus est soit donné dans l'EIA-632 soit déduit du flux des produits de travail. Ces groupes sont donc eux–mêmes des processus à part entière. Ce type de groupe comprend :

- la gestion technique (figure 2.12) ;
- la conception système (figure 2.7(c)) ;
- l'évaluation technique (figure 2.13).

groupe de type « paquetage » : Les deux processus du groupe de réalisation du produit ne sont pas invoqués au même instant (voir tableau 2.7). La phase d'intégration, de test et d'évaluation des produits finaux invoque d'abord le processus d'implémentation, demande une évaluation technique puis enfin invoque le processus de transition vers l'utilisation. Un seul appel ne peut être fait au groupe. On ne peut donc pas le définir comme un processus. Ce groupe a un rôle strictement structurant vis–à–vis des processus.

le groupe *acquisition & fourniture* : Ce groupe inclut deux processus très distincts pour lesquels elle joue un rôle structurant similaire à celui joué par le groupe de réalisation. A ce titre, il peut être lui–aussi considéré comme étant de type « paquetage ». Il inclut :

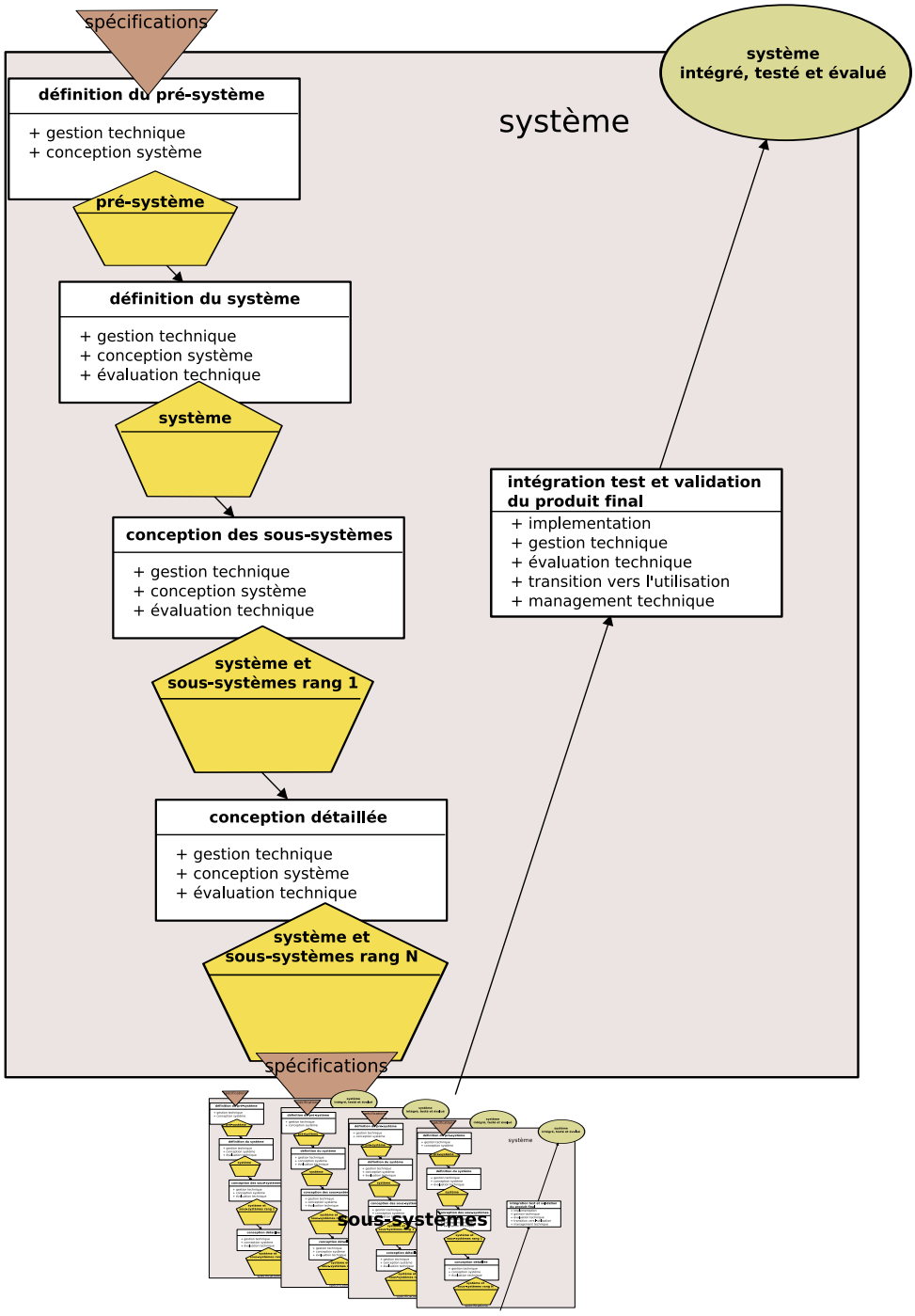


FIG. 2.11: *Vue synthétique du cycle de vie du système. Le système est obtenu par application de processus selon 5 phases. Les quatre premières sont descendantes et conduisent à la définition du système et la spécification des sous–systèmes alors que la dernière est ascendante et conduit à l'agrégation des composants, à leur test et à la validation par rapport aux spécifications initiales. Le cycle de vie s'applique sur le système et ses sous–systèmes.*

Engineering life cycle	**Life cycle phases**	**Invoque**	**Associated discipline**
Engineering life cycle	Pre-system definition	Technical mangement processes	Technical management
		System design processes	System design
	System definition	Technical mangement processes	Technical management
		System design processes	System design
		Technical evalution processes	Technical evaluation
	Subsystem design	Technical mangement processes	Technical management
		System design processes	System design
		Technical evalution processes	Technical evaluation
	Detailed design	Technical mangement processes	Technical management
		System design processes	System design
		Technical evalution processes	Technical evaluation
	End product physical integration, test and evaluation	Technical mangement processes	Technical management
		Technical evalution processes	Technical evaluation
		Acquisition process	Acquisition & Supply
		Implementation process	Product realization
		Transition to use process	Product realization

TAB. 2.7: *Processus invoqués selon les phases du cycle de vie.*

le processus d'acquisition : Ce processus est appelé de manière individuelle par les phases du cycle de vie au même titre que les processus d'implémentation et de transition vers l'utilisation. Comme pour ces derniers, les éléments entrants et sortants du processus sont un bloc de construction.

le processus de fourniture : Ce processus a la particularité de n'être jamais invoqué. Sur le tableau 2.7, on peut constater que ce processus n'est pas appelé par les phases du cycle de vie. Le groupe acquisition & fourniture n'étant pas un processus, le processus de fourniture n'est pas non plus appelé par un processus du groupe qui le contient. Le tableau 2.8 page 93 liste les invocations des processus entre eux. On peut constater sur ce tableau que le processus de fourniture n'est invoqué par aucun autre processus.

En revanche, une fois les exigences du client contractualisées, le processus de fourniture fait appel au cycle de vie d'ingénierie pour les réaliser. Contrairement aux autres processus qui font partie intégrante du cycle de vie, ce processus est donc la source de l'activité d'ingénierie et inclut par là–même l'ensemble des autres processus.

Groupes	Processus	Exigences	Invoque
Acquisition & supply	P1 – Supply process	R1 – Product supply	Engineering life cycle
	P2 – Acquisition process	R2 – Product acquisition	⊘
		R3 – Supplier performance	R10 – progress against requirements
			R33 – end products validation
			P4 – Assessment process
Technical mangement	P3 – Planning process	R4 - Process implementation strategy	⊘
		R5- technical effort definition	⊘
		R6 – schedule and organization	⊘
		R7 – technical plans	⊘
		R8 – work directives	⊘
	P4 – Assessment process	R9 – progress against plans and schedules	⊘
		R10 – progress against requirements	⊘
		R11 – technical reviews	⊘
	P5 – Control process	R12 – outcomes management	⊘
		R13 – information dissemination	⊘
System design	P6 – Requirement definition process	R14 – acquirer requirements	R26 – acquirer requirements validation
		R15 – other stakeholder requirements	R27 – otherr stakeholder requirements validation
		R16 – system technical requirements	R23 – tradeoff analysis
			R25 – requirement statements validation
			R28 – system technical requirements validation
	P7 – Solution definition process	R17 – logical solution representations	R25 – requirement statements validation
			R29 – logical solution representations validation
		R18 – physical solution representations	R22- effectiveness analysis
			R23 – tradeoff analysis

…

Groupes	Processus	Exigences	Invoque
			R24 – risk analysis
			R25 – requirement statements validation
		R19 – specified requirements	R30 – design solution verification
Product realization	P8 – Implementation process	R20 – Implementation	R31 – end product verification
			R32 – enabling product readiness
			R33 – end products validation
	P9 – Transition to use process	R21 – transition to use	⊘
Technical evaluation	P10 – System analysis process	R22- effectiveness analysis	⊘
		R23 – tradeoff analysis	R22- effectiveness analysis
			R24 – risk analysis
		R24 – risk analysis	⊘
	P11 – Requirements validation process	R25 – requirement statements validation	⊘
		R26 – acquirer requirements validation	⊘
		R27 – other stakeholder requirements validation	⊘
		R28 – system technical requirements validation	R15 – other stakeholder requirements
			R16 – system technical requirements
		R29 – logical solution representations validation	P6 – Requirement definition process
			R17 – logical solution representations
	P12 - System verification process	R30 – design solution verification	P6 – Requirement definition process
			P7 – Solution definition process
		R31 – end product verification	P6 – Requirement definition process
			P7 – Solution definition process
			P3 – Planning process
			P5 – Control process
			P12 - System verification process
		R32 – enabling product readiness	P6 – Requirement definition process
			P7 – Solution definition process
			P3 – Planning process
			P5 – Control process
			R32 – enabling product readiness
	P13 – End products validation process	R33 – end products validation	⊘

TAB. 2.8: *Processus et exigences invoqués selon les exigences associées aux processus de l'EIA-632.*

L'organisation de l'EIA-632 en groupes est essentiellement liée à des domaines d'activité. C'est d'ailleurs le point commun aux différents groupes que de servir de classifieurs. Pour le reste, il est important de conserver à l'esprit que les groupes peuvent recouvrir des rôles très différents. Certains d'entre–eux jouent uniquement un rôle de structuration thématique des processus alors que d'autres sont des processus à part entière.

2.4.4.3 Séquencement des processus

Le tableau 2.9 résume les principaux éléments du modèle. L'intégration du cycle de vie demande d'intégrer des différences au niveau des processus de l'EIA–632.

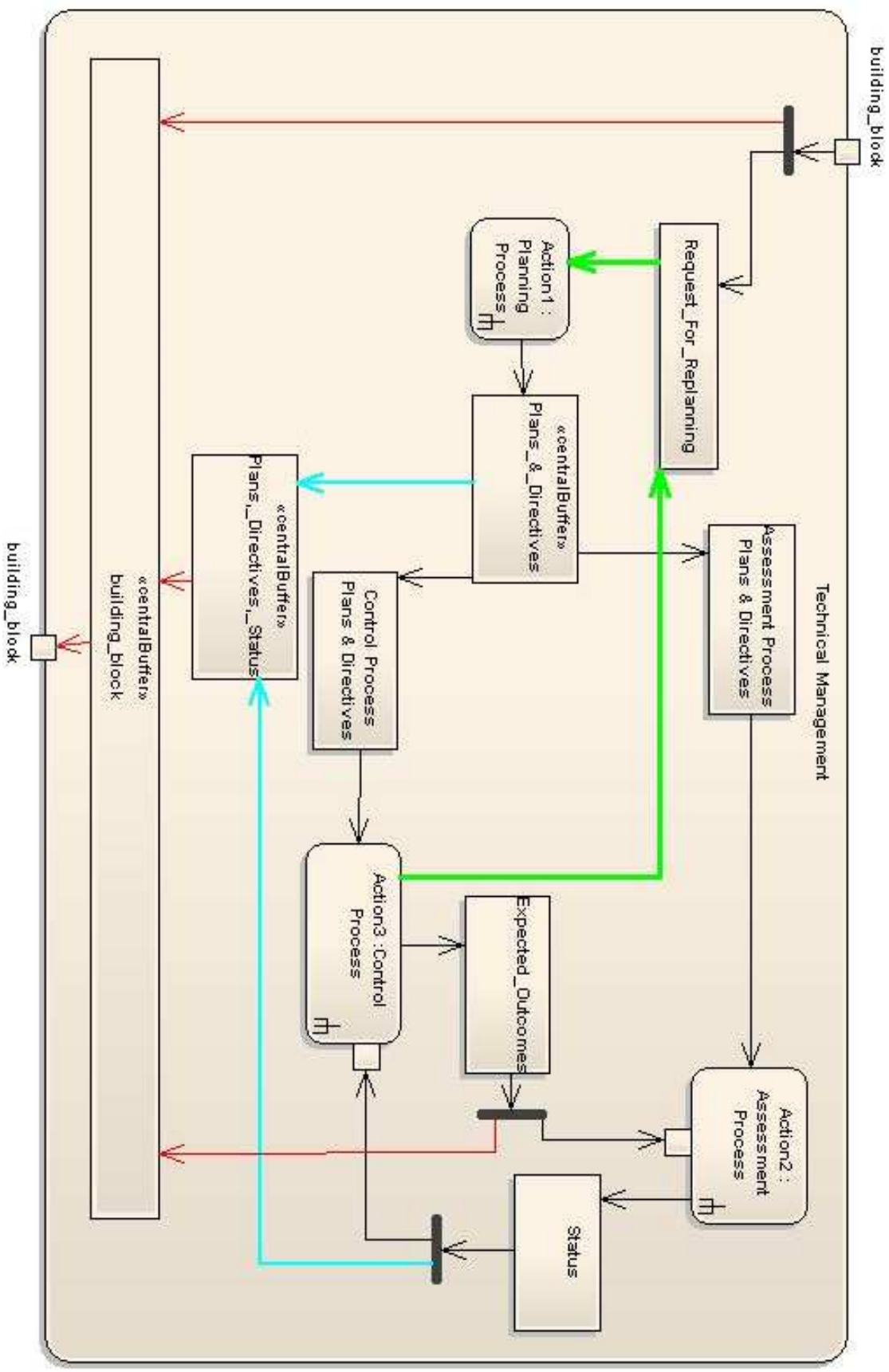


FIG. 2.12: *Groupe de gestion technique.*

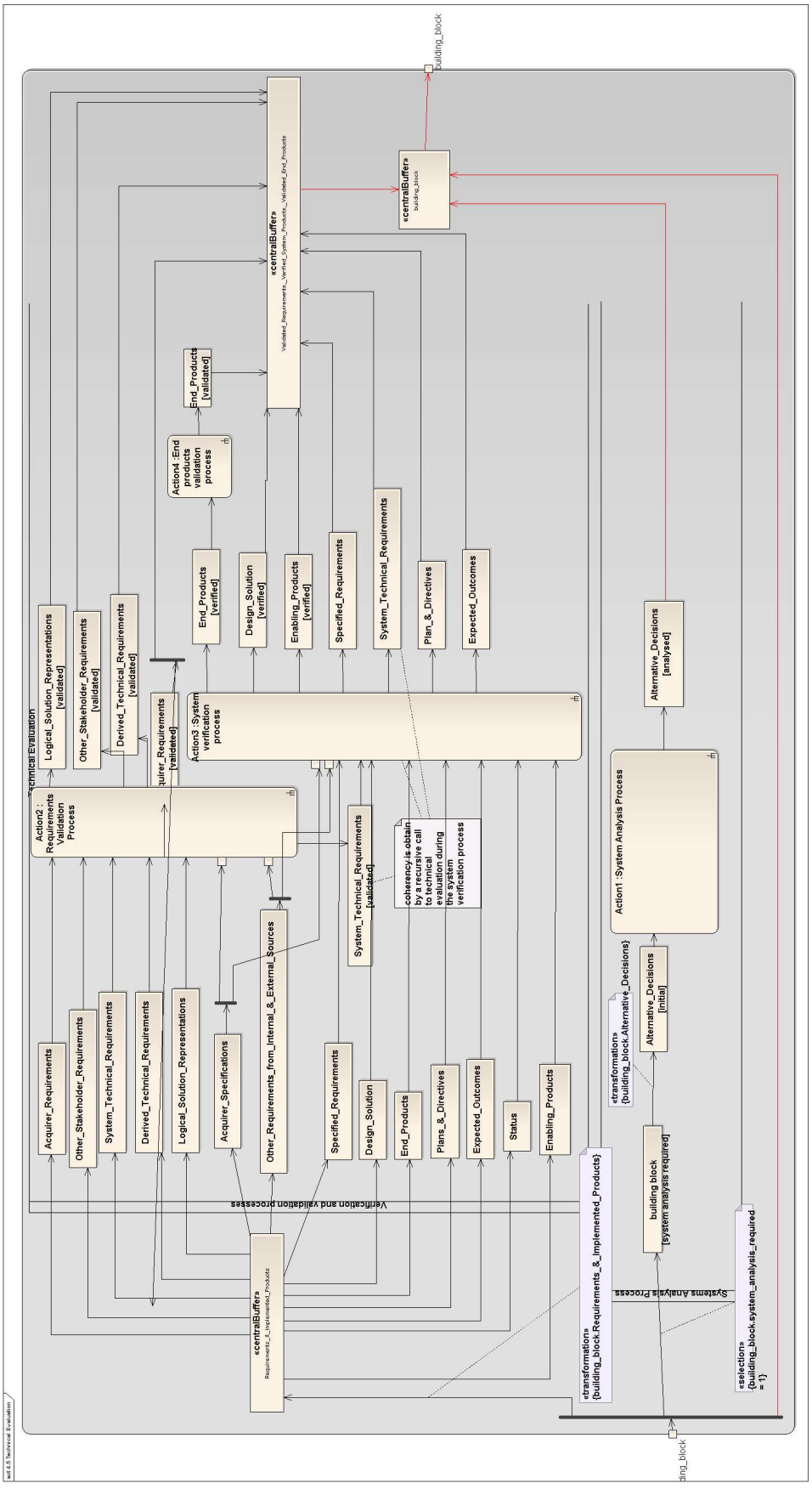


FIG. 2.13: *Groupe d'évaluation technique. Des vues détaillées de ce modèle sont données en annexes aux figures 8 et 9.*

Catégorie	Processus ou package	UML	Stéréotype SPEM	Rang
Phase	Pre-system definition	action composite	Phase	2
	System definition	action composite	Phase	2
	Subsystem design	action composite	Phase	2
	Detailed design	action composite	Phase	2
	End products physical integration, test and evaluation	action composite	Phase	2
Groupe	Acquisition & supply	package	–	–
	Technical Management	activité		3
	System design	activité		3
	Product realization	package	–	–
	Technical evaluation	activité		3
Processus de groupes de type « paquetage »	P1 – Supply process	activité	process	1
	P2 – Acquisition process	activité	process	3
	P8 – Implementation process	activité	process	3
	P9 – Transition to use process	activité	process	3

TAB. 2.9: *Tableau récapitulatif des principaux processus de l'EIA-632. Le rang représente l'ordre d'invocation. Supply process invoque les processus de rang 2 correspondant aux phases du cycle de vie qui vont invoquer à leur tour les processus de rang 3. Dans le cas de groupes de type paquetage, aucun processus n'est associé. Les appels à P1, P2, P8 et P9 se font directement.*

L'étude du cycle de vie à permis d'identifier le processus de fourniture comme étant la source des autres actions menées dans le projet.

L'ensemble des actions menées dans le cadre de l'EIA-632 doit être considéré comme la mise en œuvre des moyens permettant de satisfaire les exigences du client lorsque l'entreprise œuvre dans le rôle de fournisseur.

On distingue, dans les groupes de l'EIA-632, ceux qui jouent un rôle de processus de ceux qui jouent un rôle uniquement structurant (Acquisition & Supply et Product Realization). La représentation des premiers se fait par des diagrammes d'activité (figures 2.12, 2.7(c) et 2.13) alors que les seconds sont des paquetages.

Cette distinction se retrouve dans les processus. S'ils appartiennent à un groupe de type « processus », les éléments d'entrée et de sortie seront des éléments de bloc de construction. En revanche, s'ils appartiennent à un groupe de type « paquetage », ils travailleront sur les blocs de construction eux–mêmes.

2.4.4.4 Validité du flot de données de l'EIA-632

L'EIA-632 comporte de nombreuses étapes de vérification au cours desquelles des éléments précédemment définis peuvent être corrigés. On peut se poser la question de la cohérence du traitement de ces éléments dans le cas où une partie seulement d'éléments complémentaires est modifiée.

Prenons l'exemple du processus de vérification du système à la figure 2.14. On constate que la solution de conception, les spécifications et les exigences techniques, après avoir été définies par R30, peuvent être modifiées par R31 ou par R32. Se pose alors une question de cohérence dans la mesure où la solution de conception est associée aux spécifications. Dans le cas d'une modification des spécifications, comment s'assurer que la solution envisagée reste valide ?

Dans cet exemple, il faut descendre au niveau des processus associés aux exigences pour voir de quelle manière la cohérence est obtenue. L'exigence 31 de vérification de produit final (figure 2.15, page 99) considère deux cas :

1. Aucune correction n'est nécessaire. Dans ce cas, les éléments d'entrée ne sont pas modifiés et la cohérence est respectée.

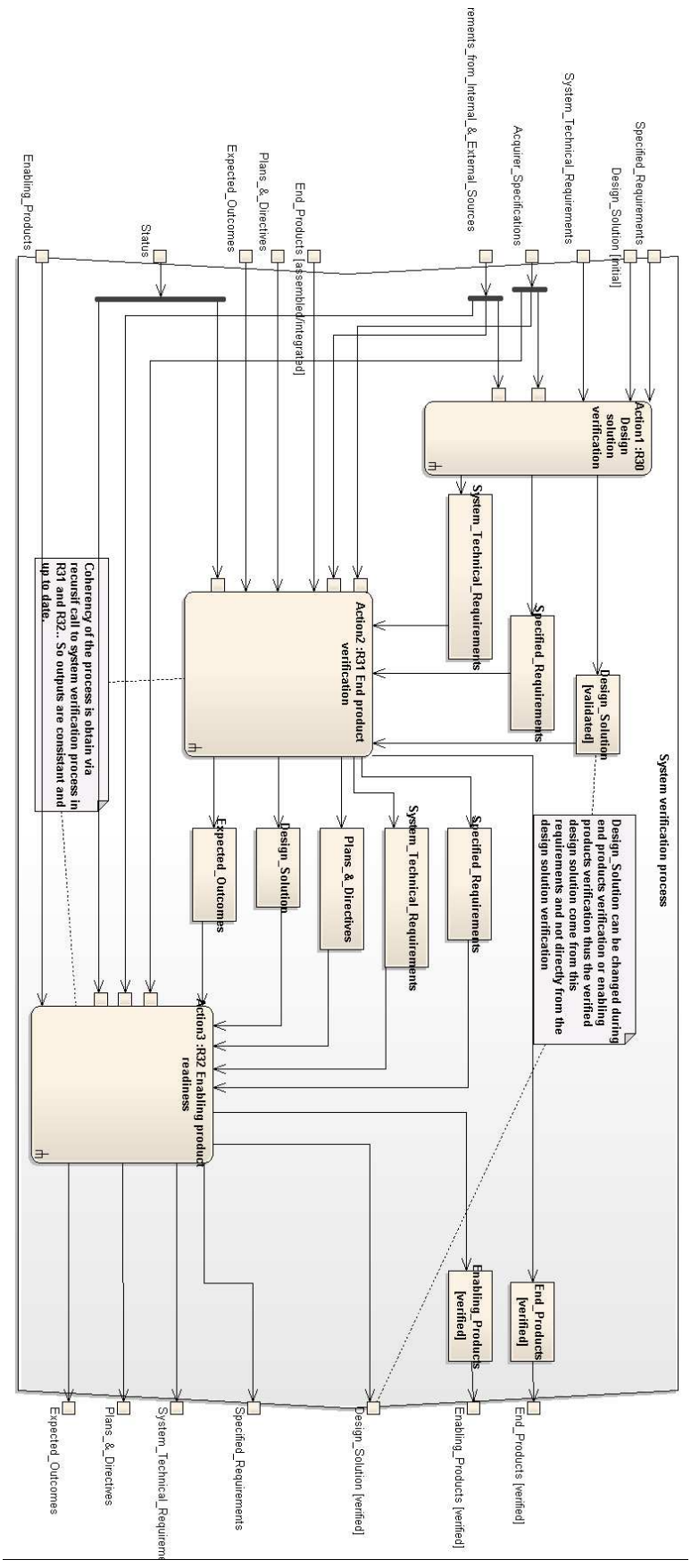


FIG. 2.14: *Processus de vérification du système. Des vues détaillées de ce modèle sont données en annexes aux figures 10 et 11.*

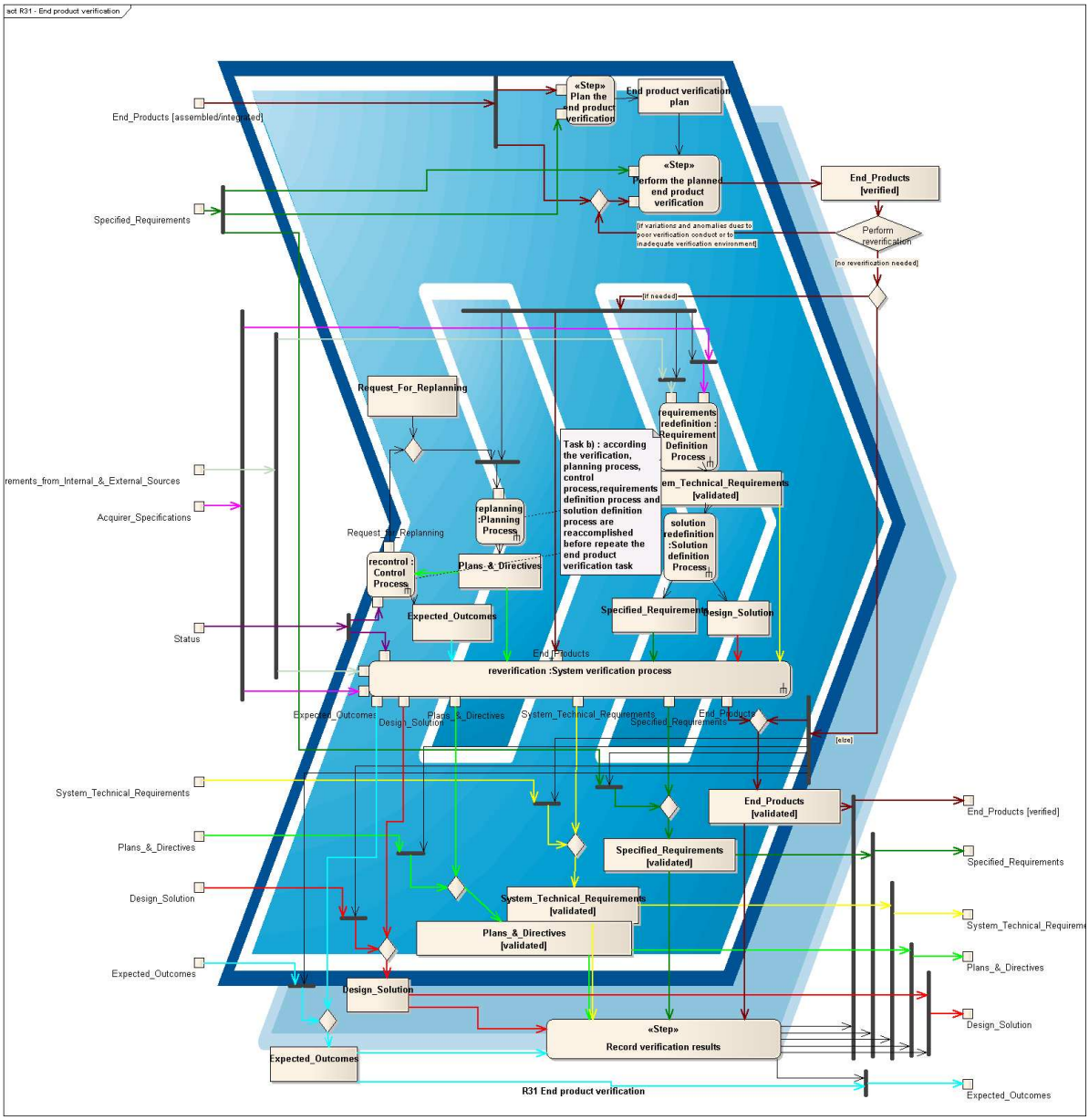


FIG. 2.15: *Exigence 31 – vérification du produit final. Des vues détaillées de ce modèle sont données en annexes aux figures 12 et 13.*

2. Une correction d'un ou plusieurs éléments précédemment définis est nécessaire. Dans ce cas, le processus de vérification du système est ré–invoqué. C'est la récursivité de ce processus qui va permettre de s'assurer de la cohérence.

La validité du flot de données est donc obtenue via la définition de processus récursifs.

L'utilisation de tels processus se retrouve souvent lors des étapes de validation au cours desquelles des éléments sont susceptibles d'être modifiés. Le tableau 2.8 permet de retracer les dépendances entre processus et de retrouver les processus récursifs.

Le flot de données de l'EIA-632 est donc valide vis–à–vis de la cohérence des données entre–elles. Il est dommage qu'un tel comportement ne soit pas mis en avant dans le standard. Le coté récursif des processus n'est décrit que de manière indirecte et le rôle spécifique de ce type de processus est complètement occulté.

2.5 Conclusion

Dans ce chapitre, nous nous sommes attachés à montrer que les concepts portés par les normes d'ingénierie système pouvaient être formalisés. Pour cela, nos deux principales contributions sont :

– d'avoir choisi un langage de représentation adapté aux particularités des normes d'ingénierie système,
– d'avoir réalisé un modèle complet de l'EIA-632 sur la base de ce langage.

L'utilisation d'un formalisme adapté permet de représenter à la fois les processus et les concepts de structure de système et donne les moyens de coordonner le tout autour du cycle de vie du système.

Le travail de modélisation de l'EIA-632 a mis en évidence des comportements jusque là restés implicites dans le standard. Il montre que le simple fait de formaliser les processus facilite leur compréhension et peut être un support de communication à la description de comportements complexes.

Dans le cadre de la mise en œuvre des processus d'ingénierie système, il est essentiel de disposer de modèles décrivant les interactions des processus avec le système qu'ils développent. De tels modèles peuvent non seulement servir aux industriels pour le suivi de leurs projets mais aussi, être à la base d'une vérification formelle des normes d'IS.

Chapitre 3

Spécialisations métier et projet des processus d'ingénierie système

« Quand on vous demande si vous êtes capable de faire un travail répondez : « bien sûr, je peux ! » Puis débrouillez–vous pour y arriver. »

Theodore Roosevelt

3.1 Introduction

L'originalité et l'intérêt de la démarche d'IS tient à l'ambition de donner un caractère universel aux recommandations qu'elle propose de mettre en œuvre. Ces normes IS décrivent des processus génériques pouvant être appliqués quel que soit le contexte. En pratique, il faudra donc adapter les méthodes, procédures et outils en fonction du projet, de l'entreprise et de son environnement. Il faudra prendre en compte l'environnement du projet (figure 1.2). Cette adaptation, nécessaire, est sans doute la difficulté principale à l'application des recommandations des normes d'IS.

Les processus décrits par les différentes normes d'ingénierie système ne sont pas accompagnés de méthodes et d'outils qui permettent de les mettre en place en s'assurant de la conformité des processus des projets par rapport aux standards.

Notre proposition ici est d'aider les acteurs de l'ingénierie système, en mettant à leur disposition des outils reposant sur la modélisation des processus d'ingénierie système permettant de mettre en place et de valider les processus des projets.

Dans le chapitre précédent, nous avons vu comment pouvaient être formalisées les normes d'IS. **Nous allons voir, dans ce chapitre, comment, sur la base du modèle de la norme, les adaptations nécessaires à l'application des recommandations dans un contexte donné peuvent être faites.**

Nous montrerons qu'un projet peut reposer sur un modèle respectant à la fois les recommandations générales d'ingénierie système et les contraintes ou recommandations de son métier (comme illustré à la figure 3.1).

Nous présenterons la démarche de modélisation qui servira de base à l'adaptation des normes d'IS à leur contexte d'application. Les éléments de ce processus seront ensuite présentés un à un : les spécificités liées au contexte d'application appelées spécificités métier, le modèle de ces spécificités ou modèle métier et, enfin, l'élément ultime de cette démarche, le modèle de projet.

3.2 Notre démarche de modélisation

3.2.1 Aperçu de la démarche

L'ingénierie système repose sur l'idée qu'il existe des processus, communs à toutes les domaines d'activité, qui permettent d'assurer le succès des projets. Ces processus ont été répertoriés et sont exprimés au travers des normes d'IS. En pratique, ils doivent cependant être adaptés d'une part au domaine d'activité, le métier, dans lequel ils seront employés et d'autre part à un projet particulier.

Notre recommandation est de respecter ces adaptations en utilisant un modèle des processus non pas sur la base des processus décrits dans les standards mais sur la base d'une adaptation de ces processus à un métier particulier et dans le cadre d'un projet. **Trois modèles sont ainsi définis : le modèle de standard évoqué au chapitre précédent, un modèle dit de métier et un modèle propre au projet.**

On envisage de construire successivement chaque modèle en utilisant des éléments du modèle précédent. Il serait cependant erroné de considérer que les modèles sont hiérarchiquement différents dans les couches du MDE. Les trois types de modèles, du standard, du métier

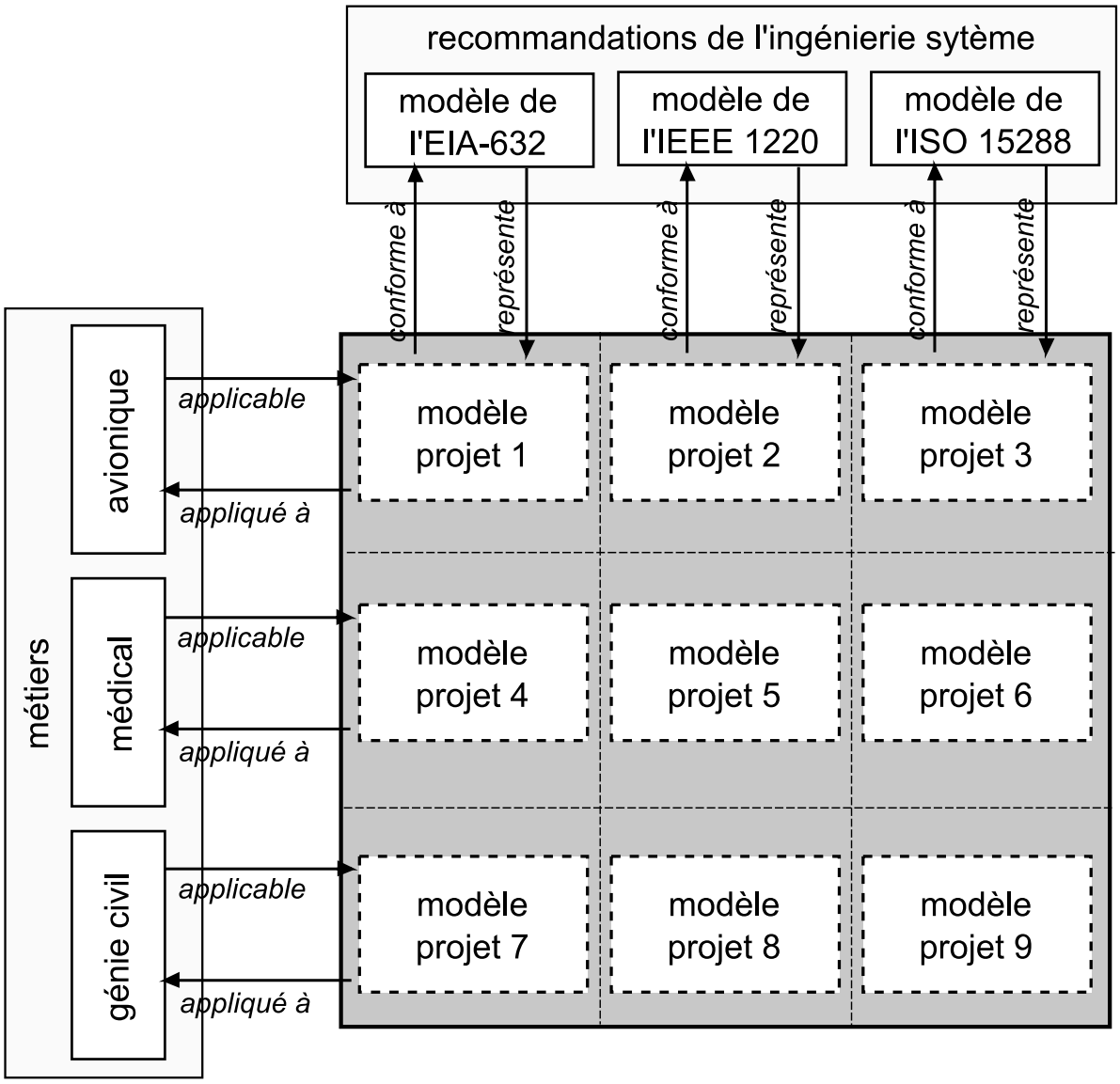


FIG. 3.1: *Conformité des modèles de projets aux processus d'IS et aux métiers. Les projets, et par extension leurs modèles, doivent être en conformité par rapport aux processus décrits par les normes d'ingénierie système tout en étant applicables à un métier particulier.*

ou du projet, se situent tous au niveau M1 de la pyramide (figure 3.2) et sont tous exprimés dans le même métamodèle. Les spécificités sont introduites progressivement dans ces modèles jusqu'à obtention du modèle de projet qui servira de référence pour la conduite et le suivi du projet réel.

3.2.2 Méthodes de construction

L'architecture retenue permet deux types de construction représentées à la figure 3.3.

La première est la construction intégrale des modèles. Sur la base du modèle de standard utilisé (EIA-632, IEEE 1220), le modèle métier est construit puis le modèle de projet.

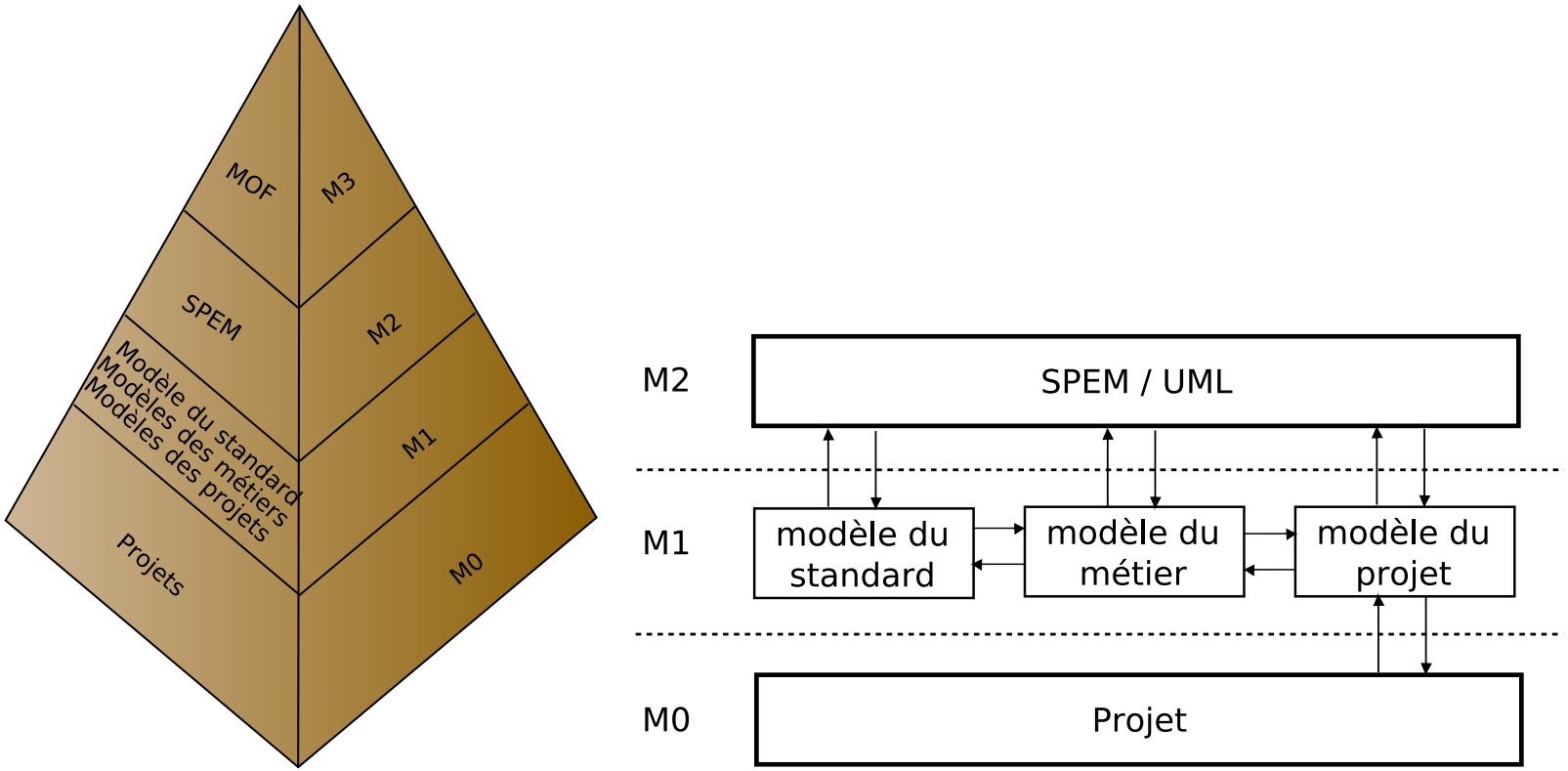
Le second type de construction repose sur la capitalisation de modèles déjà réalisés. Les recommandations des standards et l'environnement de l'entreprise (qui définit son métier) étant stables, il est possible de les réutiliser dans plusieurs projets. Ces deux modèles peuvent être choisis dans une base de données, ce qui limite le travail de modélisation à la création des modèles de projet.

3.2.3 Avantages de la démarche

D'un point de vue de l'utilisateur, cette démarche de modélisation introduit deux innovations majeures.

D'une part, les chefs de projet disposent d'une description de processus proche de leur préoccupations. Ils n'ont plus à adapter eux–mêmes les processus génériques des standards. Les processus sont plus facilement mis en application et les déviations ou adaptations des recommandations maîtrisées et formalisées.

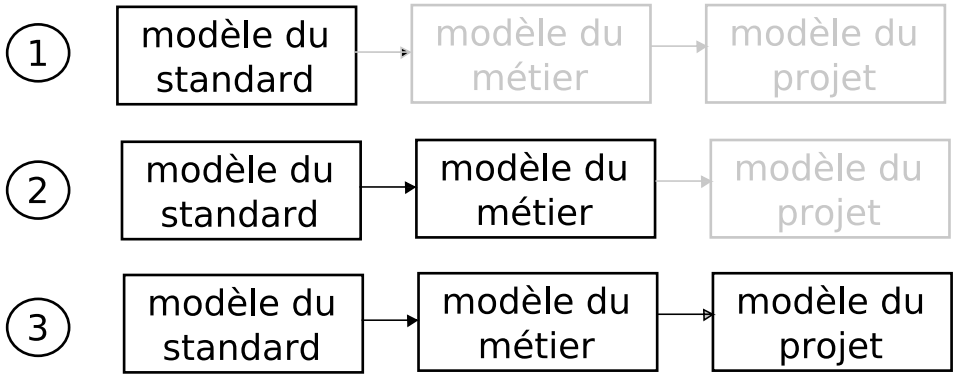
D'autre part, comme on le verra en 3.4.4, il devient possible de formaliser le fait que les adaptations des recommandations sont différentes pour chaque couche de développement. Chaque fournisseur d'un produit contributeur ou d'un sous–système est probablement d'un métier différent de celui qui développe le système principal. En permettant d'intégrer les spécificités de chacun dans un même référentiel, on facilite la coopération des différents acteurs du projet.



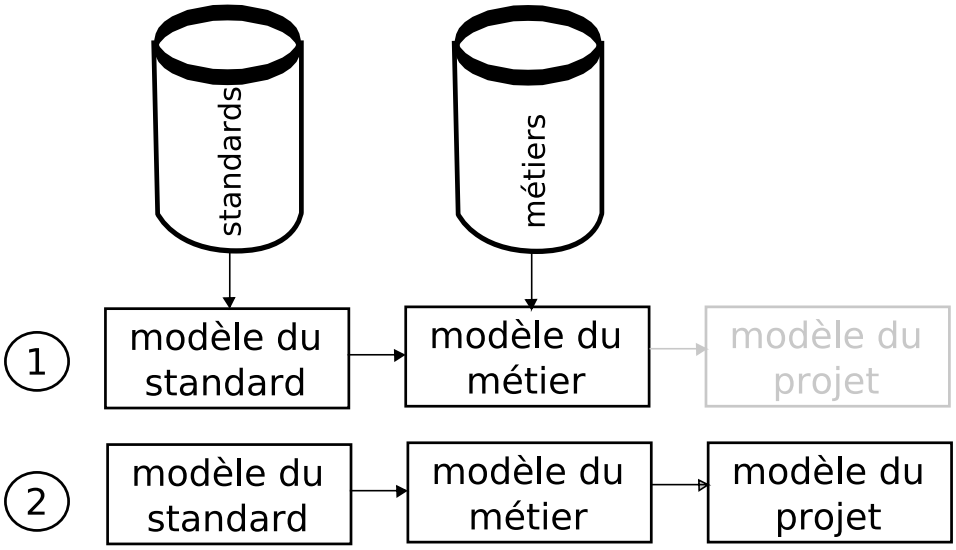
(a) Emplacement respectif des éléments de modélisation dans la pyramide du MDE

(b) Liens entre projet, modèle du standard, modèle du métier, modèle du projet et le méta-modèle SPEM

FIG. 3.2: *Situation des modèles de standard, métier et projet dans la pyramide du MDE. Les trois types de modèle sont situés au même niveau conceptuel dans la pyramide du MDE. Ils sont tous créés en utilisant le métamodèle présenté au chapitre 2. Chaque modèle est construit à partir des informations contenues dans le précédent par rapport auquel il peut être validé.*



(a) Construction intégrale des modèles standard, métier et projet



(b) Construction du modèle de projet à partir de modèles standard et métier existants

FIG. 3.3: *Modes de construction du modèle de projet. La construction du modèle de projet peut se faire soit par construction successive des modèles de standard et de métier (figure 3.3(a)), soit à partir de modèles de standard et de métier existants choisis dans une base de donnée (figure 3.3(b)).*

3.3 Les spécificités métier

Dans cette section, nous définirons le terme de métier puis nous étudierons un métier afin d'illustrer le type d'impact qu'il peut avoir sur les processus génériques. Enfin, nous présenterons les moyens de construction des modèles métier.

3.3.1 Définition d'un métier

Les normes d'ingénierie système demandent à être complétées par des éléments spécifiques sans pour autant donner de définition de « métier ». Elles restent floues quant à ce qui est couvert par le terme de métier et laissent son interprétation libre.

D'après [Aca], métier, du latin *ministerium* (fonction de serviteur) désigne :

1. Le travail dont une personne tire ses moyens d'existence et qui définit son état, sa condition ;
2. En un sens plus général, l'activité professionnelle qu'exerce une personne ;
3. Par analogie, se dit des activités qu'une personne accomplit dans l'exercice de sa charge, de son emploi, des fonctions qu'elle assume.

Avoir le même métier suppose pour deux entreprises de partager des activités communes, d'avoir le même domaine d'activité. Il serait cependant plus approprié de parler de production de produits ou de services similaires, les activités étant induites des objectifs de production. Dans le langage courant, on parle d'ailleurs des métiers de l'aéronautique, de l'automobile, de la téléphonie, *etc.* Partant du constat que c'est bien le produit (ou service) final qui va désigner le métier et qu'il induit des techniques, un savoir–faire ou des procédures spécifiques, nous définirons un métier comme :

Définition 14 : *Métier*

L'ensemble des spécificités techniques ou non liées à la réalisation d'un produit ou d'un service.

Une liste définitive de métiers ne peut être fixée à partir de cette définition. Les évolutions des besoins et des technologies font qu'en permanence, on voit apparaître de nouveaux

produits ou services alors que d'autres disparaissent. Il convient alors de s'adapter à ces modifications et de suivre les évolutions lorsqu'elles surviennent.

La définition 14 dépend de la vision que l'on a des produits ou des services. Dans ce cadre, doit–on considérer la production de deux voitures, l'une routière et l'autre sportive, comme relevant du même métier ou de métiers différents ? En d'autres termes, s'agit–il de produits distincts ou de deux produits similaires ? En fonction du degré d'abstraction choisi deux réponses sont acceptables :

1. Il s'agit bien du même type de produit à quelques options près et leurs réalisations relèvent du même métier ;
2. Les techniques mises en œuvre pour réaliser ces deux types de voiture sont sensiblement différentes. On ne peut les considérer comme des produits identiques. En conséquence il s'agit de métiers différents.

Pour répondre à ce problème nous avons choisi de nous référer à la classification des normes de l'AFNOR. Cette classification, donnée au tableau 3.1, nous servira de référence.
Nous avons choisi cette classification pour pouvoir valider notre démarche car elle a l'avantage de ne pas être trop spécifique sans pour autant être trop simpliste. De plus, elle permet de retrouver facilement quelles seront les spécificités relatives à chaque métier en fonction des normes qui lui seront associées. Elle respecte aussi les recommandations des standards qui préconisent de compléter leurs processus par les spécificités métier dont les normes métier font partie. En choisissant cette échelle de modélisation, on favorise l'identification des impacts du métier sur les processus généraux. Enfin, elle reste cohérente avec les visions de l'environnement du projet ou de l'entreprise données par les normes d'IS (figure 1.7) dans lesquelles les références aux normes ou aux standards sont très fortes.

Pour les besoins d'un projet particulier, il reste cependant possible de choisir une référence différente.

3.3.2 Un exemple de métier : les EPI (Équipements de Protection Individuelle)

Nous allons tenter d'identifier ce qui fait la spécificité des métiers. Pour cela, nous nous baserons sur les normes qui leur sont attachées. Bien entendu, il n'est pas suffisant d'intégrer les aspects normatifs dans les modèles de projets. D'autres éléments peuvent devoir y être

#	Intitulé	#	Intitulé
01	généralités. terminologie. normalisation. documentation	03	services. organisation de l'entreprise. gestion et qualité. administration. transport. sociologie
07	mathématiques. sciences naturelles	11	technologies de la santé
13	environnement et protection de la santé. sécurité	17	métrologie et mesurage. phénomènes physiques
19	essais	21	systèmes et composants mécaniques à usage général
23	fluidique et composants à usage général	25	techniques de fabrication
27	ingénierie de l'énergie et de la transmission de la chaleur	29	électrotechnique
31	électronique	33	télécommunications. techniques audio et vidéo
35	technologies de l'information. machines de bureau	37	technologie de l'image
39	mécanique de précision. bijouterie	43	véhicules routiers
45	chemins de fer	47	construction navale et structures maritimes
49	aéronautique et espace	53	matériel de manutention des matériaux
55	emballage et distribution des marchandises	59	industrie textile et technologie du cuir
61	industrie du vêtement	65	agriculture
67	technologie alimentaire	71	génie chimique
73	mines et minerais	75	industrie du pétrole et technologies associées
77	métallurgie	79	technologie du bois
81	industries du verre et de la céramique	83	industries des élastomères et des plastiques
85	technologie du papier	87	industries des peintures et des couleurs
91	bâtiment et matériaux de construction	93	génie civil
95	génie militaire	97	équipement ménager et commercial loisirs. sports
99	(sans titre)		

TAB. 3.1: *Classification des normes AFNOR.*

insérés comme des pratiques ou un vocabulaire spécifique. Cependant, notre objectif ici est de voir ce que peut être une caractéristique liée à un métier et comment elle peut interagir avec les processus standards plutôt que de faire un modèle précis des métiers étudiés.

Le rôle et la place des normes sont très diversifiés selon les métiers. Dans de nombreux domaines, comme l'automobile, les normes sont principalement des contraintes sur le produit. Pour d'autres, par contre, elles peuvent être indispensables à l'introduction d'une technologie. C'est le cas dans le domaine des communications ou de l'aéronautique.

Dans les deux cas, faire de sorte d'imposer des techniques ou des processus déjà maîtrisés par l'entreprise dans une norme est un enjeu stratégique, les concurrents doivent alors rattraper leur retard sous peine d'être coupés du marché.

Pour donner une idée pertinente de ce que peut être un métier, nous avons choisi d'analyser un métier de taille restreinte avec de fortes contraintes normatives : le métier des EPI (Équipements de Protection Individuelle).

3.3.2.1 Définition d'un EPI

Un équipement de protection individuelle est une protection concernant un individu contre un risque donné. Il peut s'agir d'un casque, de chaussures de sécurité, de vêtements protecteurs, de gants, de lunettes, de masque, de protections auditives contre le bruit, de protection respiratoire, de harnais, *etc.*

La conception, la fabrication et l'utilisation de ces équipements ont la particularité d'être particulièrement encadrée par la règlementation. Ce domaine d'activité étant relativement limité et fortement encadré, on peut estimer que l'analyse des normes existantes permet l'obtention d'une image précise du métier correspondant.

La définition de l'article R233-83-3 décret n°92765 du 29/05/92, un EPI est :

Définition 15 : *Équipement de Protection Individuel (EPI)*

Dispositif de protection individuelle préservant une personne d'un risque menaçant sa sécurité. Cette définition est étendue aux dispositifs associés de façon solidaire ainsi qu'aux composants interchangeables.

Il existe trois catégories d'EPI reportées au tableau 3.2. Selon la gravité du risque pris par l'utilisateur on considère les EPI de catégorie 1, dont l'utilisation entraîne des risques mineurs (petits chocs mécaniques, rayonnement solaire), les EPI de catégorie 2, dont l'utilisation entraîne des risques graves, et les EPI de catégorie 3, dont l'utilisation entraîne des risques majeurs ou mortels.

3.3.2.2 Quelles exigences pour les EPI ?

Les exigences s'appliquant aux EPI peuvent être classées de la façon suivante [PET] :

- exigences techniques ;
- exigences de marquage ;
- exigences d'informations de l'utilisateur ;
- obligation des distributeurs ;
- exigences de vérification et de maintenance.

Chacune de ces exigences sera détaillée ci–dessous.

Exigences techniques

L'EPI doit satisfaire aux exigences essentielles de sécurité et de santé. Il est conçu et fabriqué pour assurer le plus haut niveau de protection possible en respectant l'ergonomie et le confort de l'utilisateur. La conformité aux normes européennes est un moyen de vérifier que les équipements répondent à des exigences techniques définies. Ce n'est cependant pas le seul car :

- Il n'existe pas de normes pour tous les produits visés par la règlementation EPI,
- Le fabricant peut s'écarter de la norme s'il dispose d'une solution technique plus appropriée pour répondre à des situations particulières et aux exigences règlementaires.

L'application des normes européennes n'est pas obligatoire.

Exigences de contrôle

Les exigences associées à chaque type d'EPI sont reportées au tableau 3.2. Selon le type d'EPI, et donc selon les risques encourus par l'utilisateur, les contrôles seront plus ou moins restrictifs. Ils peuvent aller d'une auto–certification par le fabricant à un examen de type et une garantie qualité CE.

Lors d'un examen CE de type, un laboratoire agréé teste un ou plusieurs échantillons du produit pour vérifier sa conformité aux exigences techniques. Si l'examen est positif, le fabricant obtient une attestation d'examen CE de type.

Pour les EPI de catégorie 3, le fabricant est soumis à une procédure de Garantie Qualité CE, c'est–à–dire qu'il est contrôlé par un organisme notifié au cours de la fabrication et sur le produit final.

Catégorie de l'EPI	**Type de risque**	**Contrôles nécessaires**
1	risques mineurs	auto–certification du fabricant
2	risques graves	examen CE de type
3	risques majeurs ou mortels	Examen CE de type + Garantie Qualité CE

TAB. 3.2: *Types d'EPI et exigences de contrôle associées en fonction des risques entraînés.*

Exigences de marquage

Tous les EPI doivent porter le marquage CE, les 2 derniers chiffres de l'année de fabrication, le numéro de série et le nom du fabricant.
Le marquage CE indique qu'un produit répond aux exigences de la directive européenne 89/686/CEE. Pour les EPI de catégorie 3, le marquage du numéro d'identification du laboratoire qui assure le contrôle qualité est également obligatoire.
Ce marquage est à différencier du marquage propre à la société pour identifier ses produits et conserver une traçabilité.

Exigences d'informations de l'utilisateur

Tous les EPI doivent être accompagnés d'une notice technique du fabricant précisant en particulier :

- les instructions d'emploi, de stockage, de nettoyage, d'entretien et de révision,
- les performances et les examens techniques,
- les instructions de compatibilité avec d'autres produits,
- les limites d'utilisation,
- les dates et délais de péremption,

– une fiche descriptive de suivi (nom et adresse du fabricant, N° de série, année de fabrication, date d'achat, date de première mise en service, nom de l'utilisateur, espace commentaires).

Obligation des distributeurs

Le distributeur doit au minimum veiller à la conformité des EPI qu'il propose au consommateur : présence du marquage CE et de la notice technique qu'il ne doit en aucun cas désolidariser du produit.

Vérification et maintenance des EPI

L'arrêté du 19 mars 1993 (qui fait référence à la directive 89-656 relative à l'utilisation des EPI dans le cadre des vérifications périodiques) impose que tout EPI utilisé soit soumis à des vérification périodiques au moins tous les 12 mois.

Ces vérifications sont décrites par :

– Le décret N° 93-41 du 11/01/93 qui transpose la directive 89-656-CEE en droit français concernant les mesures d'organisation et les conditions de mise en œuvre et d'utilisation applicables aux équipements de travail et aux moyens de protection.
– L'arrêté du 19 mars 93 qui fixe la périodicité des vérifications et la liste des EPI objets de vérifications.

Le fabricant devra indiquer les critères et les limites du contrôle et peut fixer une périodicité de contrôle plus courte. Ces vérifications ont pour but de déceler, en temps utile, toutes détériorations susceptibles d'être à l'origine de situations dangereuses.

Les périodicités des contrôles selon le fabricant s'exercent :

– avant toute mise en service et attribution personnelle,
– avant et après toute utilisation,
– tous les trois mois (inspection approfondie).

Tout utilisateur doit ainsi informer le responsable d'entreprise des incidents rencontrés et des défauts constatés. Les vérifications peuvent être confiées à des personnes qualifiées ayant acquis cette compétence.

Les dates et les résultats des contrôles annuels sont consignés sur le registre de sécurité à tenir à disposition des services de contrôle et de prévention. Sur ce registre, seront mentionnés

par article contrôlé : le modèle, le numéro de série, l'année de fabrication, la date d'achat, la date de première utilisation, le nom de l'utilisateur si l'EPI est attribué nominativement.

3.3.3 Impact sur les projets

Dans cette section, nous allons tenter d'identifier **de quelles manières les spécificités d'un métier vont influer sur les processus génériques décrits par les normes d'ingénierie système.**

Nous ferons deux constatations. La première est qu'une même norme métier peut potentiellement impacter toutes les composantes du projet. La seconde est qu'un même produit est couvert par de nombreuses normes. Il en découle que non seulement les impacts peuvent être multiples mais qu'ils peuvent provenir de nombreuses sources.

Sont ainsi touchés dans le processus générique :

Le produit : Des exigences peuvent être imposées sur le produit par les normes ou les pratiques du métier. C'est le cas des spécifications techniques automobiles ou du marquage des EPI.

Les produits contributeurs : De la même manière que pour le produit, des exigences spécifiques peuvent être ajoutées aux produits contributeurs, ou, des produits contributeurs être imposés comme par exemple la notice liée aux EPI.

Les produits de travail : Le métier peut imposer des produits de travail particulier ou du moins imposer leur format. Par exemple, les produits issus de la phase de conception seront des dessins techniques en mécanique et des schémas électroniques dans le métier éponyme.

Les rôles : En plus d'un vocabulaire spécifique pour désigner des rôles génériques (mécanicien et électronicien désignent tous deux un concepteur), de nouveaux rôles peuvent être introduits par le métier. Généralement, ces rôles sont liés à de nouvelles tâches. C'est le cas par exemple de la personne en charge de la certification pour les EPI.

Le processus : Comme évoqué ci-dessus, de nouvelles tâches peuvent être introduites.

Les procédés ou les outils : Enfin, les moyens peuvent aussi être affectés.

L'identification et la modélisation d'un métier se fait donc en deux étapes. Dans un premier temps, il convient de lister les normes qui seront en vigueur. Cette première étape peut être

rendue difficile si le nombre de normes est important ou/et si certaines d'entre–elles sont contradictoires. La seconde est de représenter les impacts de ces normes sur le modèle des processus générique (figure 3.4).

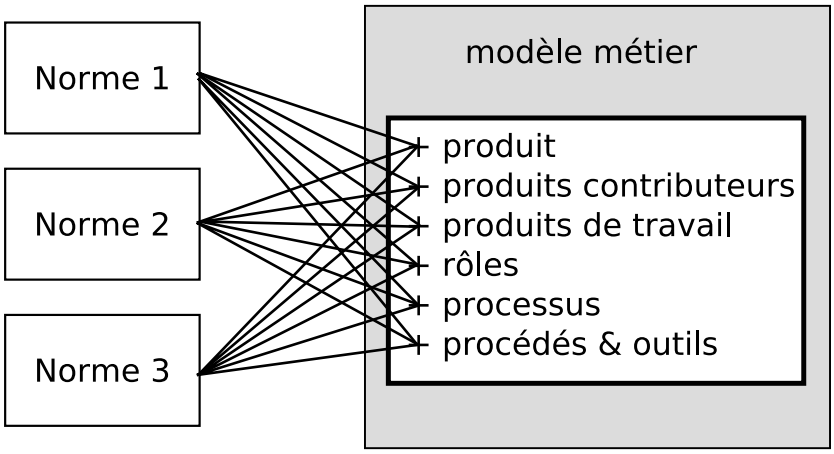


FIG. 3.4: *Illustration de l'impact des normes sur le modèle générique. La modélisation d'un métier demande d'intégrer des spécificités provenant de plusieurs normes distinctes. Chacune d'entre–elles peut potentiellement impacter tous les types d'éléments du modèle générique.*

En pratique, la première de ces étapes est faite naturellement par les industriels. Recenser et appliquer les contraintes légales ou règlementaires auxquelles ils sont soumis n'a rien d'inhabituel. On note même, dans certains cas, qu'un travail de formalisation des normes est effectué [Bla00]. Dans la section suivante, on supposera que la liste des normes à appliquer au processus est connue. On s'attardera plus particulièrement sur le second point concernant les méthodes et moyens de modélisation des impacts des normes.

3.4 La construction d'un modèle de processus métier

Un modèle métier représente les processus génériques décrits par les normes d'ingénierie système auxquelles sont ajoutées les spécificités du métier dans lequel elles vont être appliquées.

Comme représenté à la figure 3.5, il s'agit de trouver les moyens de réaliser la transformation du modèle de standard en modèle du métier en y intégrant les spécificités du métier.

On se trouve face à une double difficulté de modélisation :

– Comme on l'a vu ci–dessus, une même norme peut potentiellement impacter tous les éléments du processus générique ;

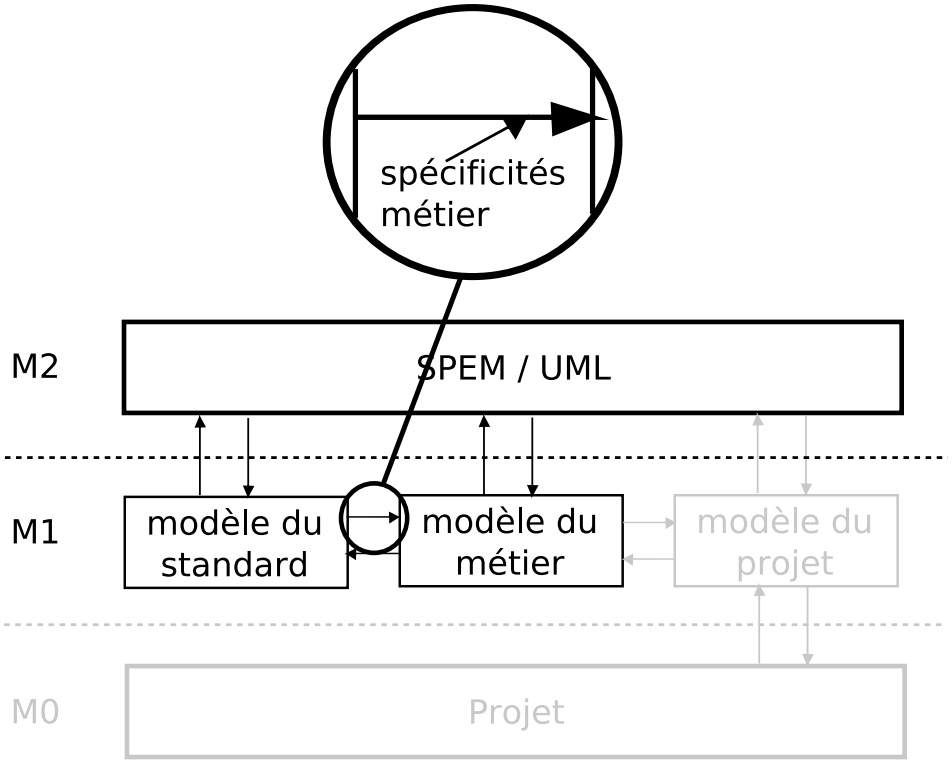


FIG. 3.5: *Passage du modèle de standard au modèle métier. Cette étape passe par la transformation du modèle de standard écrit en SPEM en modèle métier lui aussi rédigé en SPEM. La transformation permet d'ajouter les spécificités du métier au modèle générique de l'EIA–632.*

– Plusieurs normes se rapportent à un même produit ou projet.

Dans certains cas cependant, le domaine impacté est clairement identifié par la norme. Pour s'en convaincre, il suffit d'observer la cartographie des documents en support des normes ISO 9000 et 2000 donnée à la figure 3.6. Citons, pour exemples, les normes FD X50-176 [AFN05] « management des processus », FD X50-189 [AFN04] « lignes directrices pour l'intégration des systèmes de management », FD X50-128 [AFN03a] « lignes directrices pour le processus achat et approvisionnement », FD X50-127 [AFN02] « maîtrise du processus de conception et développement » ou FD X50-179 [AFN00] « guide pour l'identification des exigences des clients » qui sont toutes facilement assimilables à un des processus de l'EIA-632. La modélisation du métier est alors facilitée dans la mesure où les éléments à intégrer sont limités à un sous–ensemble du standard.

De la même manière, l'intégration des normes d'IS complémentaires vue à la section 1.4.4 est facilitée par une portée limitée de ces normes. Une exception existe toutefois : les normes spécifiques à un domaine pour lesquelles l'ensemble des processus sont touchés. Par contre, elles peuvent être considérées comme l'expression de spécificités métier déjà introduites dans une norme générale.

3.4.1 Exemple de modèle métier : le cas des EPI

La notion de modèle métier sera illustrée sur le cas des EPI.

La première étape de modélisation consiste à lister toutes les normes ou autres spécificités métier, se référant à ce type de produit. C'est un travail dont la synthèse a déjà été présentée à la section 3.3.2.2. En résumé et de manière simplifiée, la fabrication d'un EPI est soumise à :

– à une auto–certification par le fabricant ;
– à un test du produit par un laboratoire indépendant ;
– au contrôle annuel du produit et de la production ;
– à un marquage spécifique du produit ;
– à la production d'une notice spécifique.

Nous avons suffisamment d'éléments rappelés dans le paragraphe précédent pour illustrer comment ces éléments peuvent être intégrés dans le modèle de processus générique de l'EIA-632 présenté au chapitre 2 :

auto–certification par le fabricant : Cette spécificité peut être décomposée en deux partie :

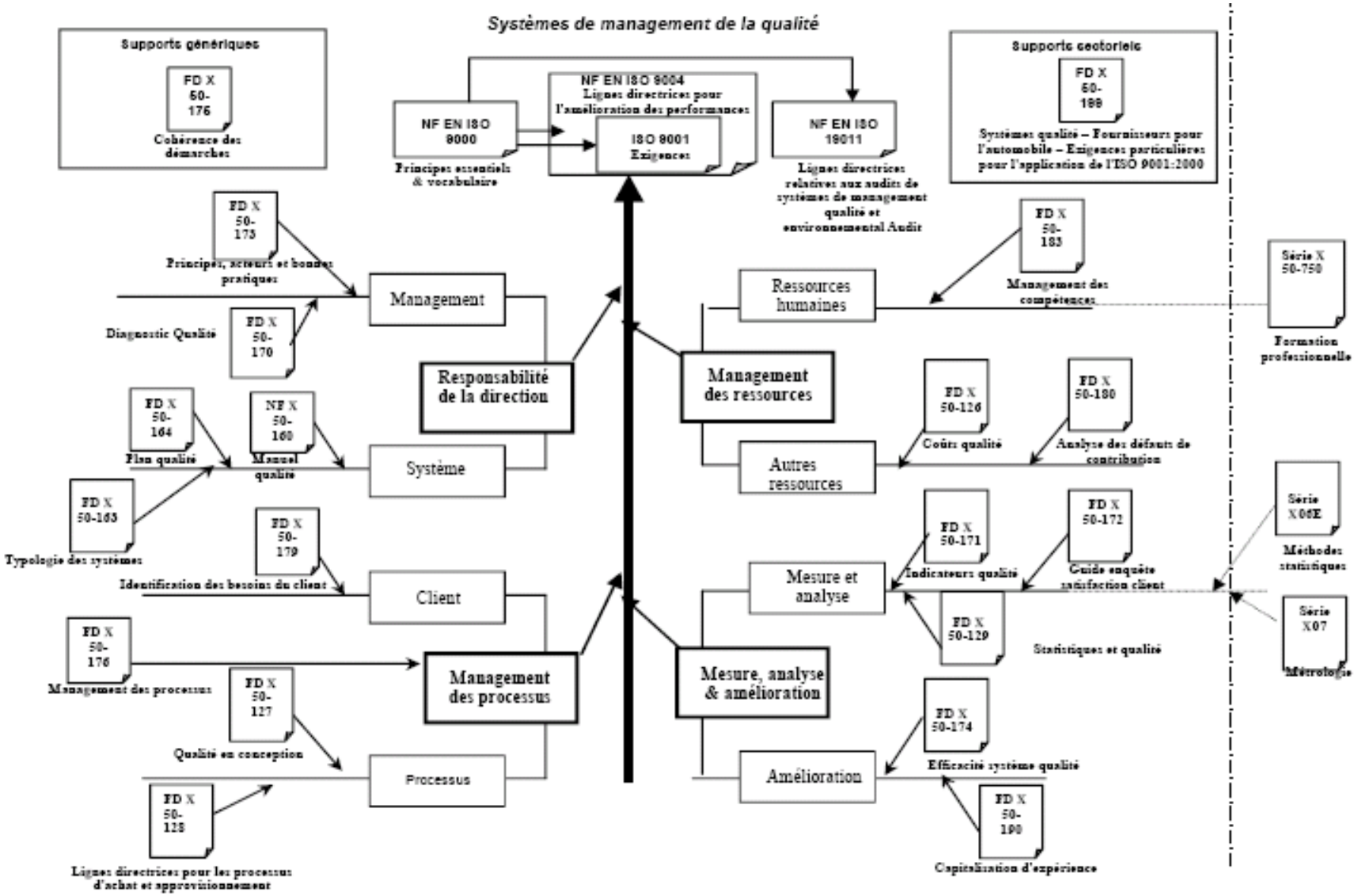


FIG. 3.6: *Cartographie des documents français en support des normes ISO 9000 :2000. Source AFNOR [AFN04, AFN03a].*

1. les exigences, essentiellement techniques, fixées par les normes, sont ajoutées de manière systématique aux exigences du produit par le biais d'une modification du « processus de définition des exigences ». Le tableau 9 en annexe donne un exemple de telles exigences ;
2. les processus sont modifiés pour intégrer l'auto–certification. On modifiera particulièrement le « Processus de vérification des exigences système » comprenant les processus de « vérification de la solution de conception » et de « vérification du produit final ». Les normes fixent en effet certains éléments du plan de vérification en imposant des points de vérification précis (exemple tableau 10 en annexe).

test du produit par un laboratoire indépendant : Une nouvelle tâche est ajoutée dans le « processus de vérification du produit final ». Elle sera sous la responsabilité d'un nouvel acteur, le laboratoire.

contrôle annuel du produit et de la production : Cette exigence demande d'introduire les points qui seront contrôlés annuellement comme exigences sur le processus lui–même. Le « processus de contrôle » est à modifier en conséquence.

marquage spécifique du produit : Encore une fois, il s'agit principalement de fixer des exigences sur le produit par le biais d'une modification du « processus de définition des exigences ».

production d'une notice spécifique : Cette dernière spécificité est plus délicate à modéliser. D'un point de vue du modèle métier, il s'agit de l'introduire comme une exigence du système. Cependant celle–ci entraînera systématiquement le déclenchement d'un processus de conception du produit contributeur « notice ». Elle sera alors représentée par un bloc de construction spécifique.

3.4.2 Construction incrémentale du modèle métier

Le choix du niveau de modélisation est laissé libre. Il est fixé par les besoins et la définition de métier propre à l'entreprise.

L'étape de modélisation métier n'est pas neutre. Elle peut refléter certains choix stratégiques, par exemple, le choix de suivre ou non des labels (par exemple les recommandations de l'UIAA pour les EPI) ou des chartes.

En pratique, comme illustré à la figure 3.7, nous proposons que le passage au modèle métier se fasse en plusieurs passes pour intégrer les éléments provenant de sources différentes.

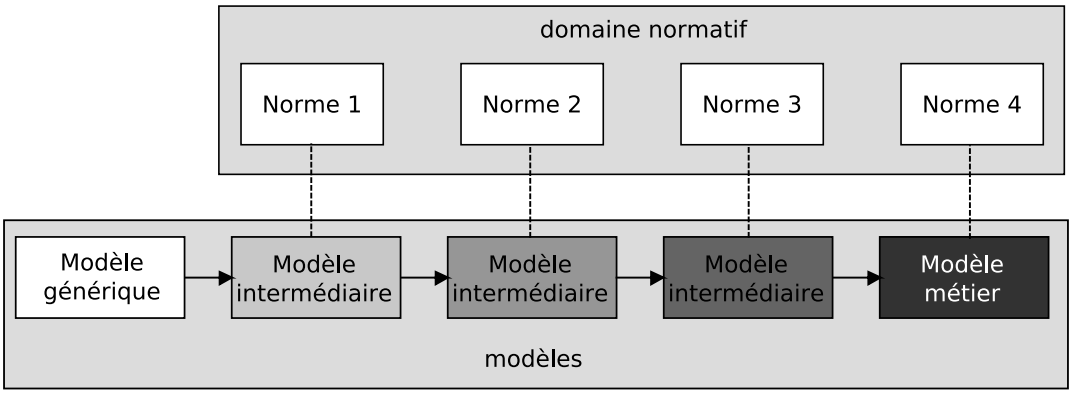


FIG. 3.7: *Construction incrémentale du modèle métier. Les spécificités issues des normes du métier sont insérées une à une à partir du modèle générique. Après chaque intégration, un modèle intermédiaire est obtenu contenant les spécificités d'une nouvelle norme et les spécificités des normes déjà intégrées. Le modèle métier est obtenu par la modélisation des spécificités de toutes les normes.*

Ce travail, pour l'immédiat, doit se faire manuellement. Cependant, il sera intéressant de lui associer des outils favorisant la traçabilité entre modèles et normes pour :

Conserver une traçabilité des spécificités au travers des modèles : Le mécanisme de modélisation doit permettre de lier chaque modification du processus générique à une norme source afin de permettre aux utilisateurs de disposer d'une vue d'ensemble de la norme considérée. Prises séparément, les modifications apportées au processus générique peuvent sembler incohérentes ou inutiles. Connaître les raisons de ces modifications permet de mieux les appréhender et les appliquer.

Permettre la mise en place d'une bibliothèque de modèles : Disposer d'un modèle métier n'est intéressant que dans la mesure où le travail de modélisation nécessaire doit être réalisé une seule fois. Une fois le modèle réalisé, l'avantage est de pouvoir le réutiliser dans différents projets. Avoir à disposition une liste de modèles métiers classés en fonction de leurs normes afférentes faciliterait cette réutilisation.

Favoriser les évolutions : Les normes ne sont pas figées et subissent des évolutions. Ne pas pouvoir tracer une norme au modèle demanderait de recommencer en grande partie le travail de modélisation lors de l'évolution d'une norme.

3.4.3 Opérations de modélisation complémentaires à l'élaboration du modèle métier

Pour réaliser ces modélisations successives du métier, nous proposons d'appliquer les opérations suivantes :

opération d'ajout : De nouveaux rôles, activités, produits de travail ou notes (incluant les contraintes OCL) peuvent être introduits dans le modèle.

opération de spécialisation : Cette opération permet de remplacer des éléments génériques par des éléments de même type avec une responsabilité restreinte. Par exemple, le rôle générique de développeur responsable de toutes les activités de l'EIA-632 sera remplacé par des rôles de responsable produit, électronicien ou thermodynamicien selon le métier et les activités considérés. La spécialisation peut être limitée à une évolution sémantique permettant l'utilisation du vocabulaire métier. Dans le cas d'un produit de travail, l'opération de spécialisation peut permettre d'associer un format aux produits de travail (dessin technique, schéma électronique, *etc.*).

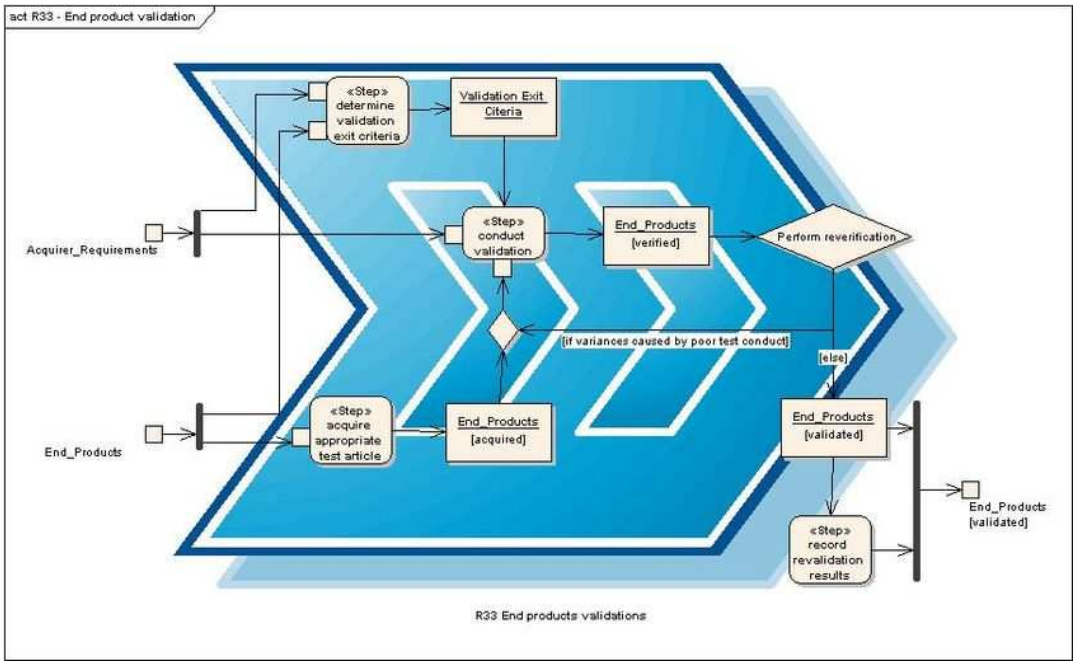
opération d'affinage : La connaissance du métier entraîne souvent une meilleure connaissance du processus laissée jusqu'ici à un niveau d'abstraction élevé. Ils peuvent être affinés par une modélisation fine des processus métier (description de protocoles métiers, association d'outils aux tâches, *etc.*).

Ces différentes opérations concernent potentiellement tous les éléments du modèle générique.

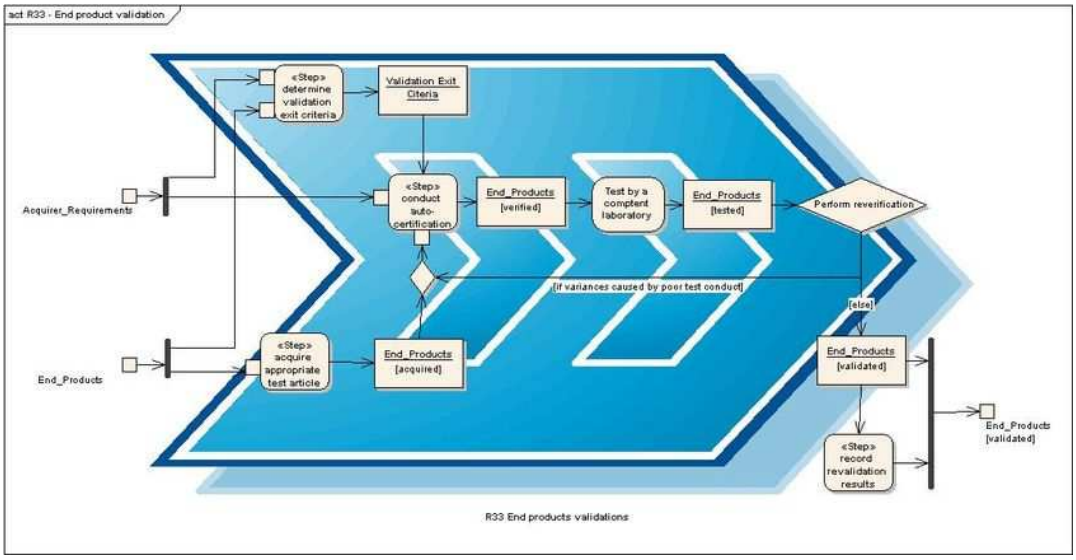
La figure 3.8 illustre la réalisation de ces opérations. Elle représente l'exigence 33 de l'EIA-632 : « validation du produit final » dans le modèle du standard et dans le modèle métier des EPI. On peut constater, entres autres, l'*ajout* d'une nouvelle activité de « test par un laboratoire habilité » et d'un nouveau produit de travail ainsi qu'une *spécialisation* de l'activité de « conduite de validation » en « conduite de l'auto–certification ».

La connaissance des normes relatives aux essais de validation (voir l'exemple du tableau 10) permet d'*affiner* l'activité d'auto–certification. On peut lui ajouter des diagrammes (non représentés ici) la décrivant plus finement.

L'opération de suppression est délibérément exclue de la liste des opérations. Nous considérons que les processus génériques décrits par les normes d'ingénierie système doivent être pris comme référence et qu'ils sont supports d'une bonne gestion de projet. Permettre la sup-



(a) Requirement 33, End Product Validation dans le modèle de standard.



(b) Requirement 33, End Product Validation dans le modèle de métier des EPI.

FIG. 3.8: *Illustration des opérations de passage du modèle de standard au modèle métier. Exemple de la validation du produit final.*

pression de ces éléments reviendrait à modifier cette référence et empêcherait tout travail de validation des processus de projet par rapport aux processus de référence.

La suppression peut aussi concerner des éléments issus de normes et intégrés par la suite dans le modèle générique. Là encore, cette opération n'est pas permise car :

– elle peut être source d'erreur,

– elle permettrait d'intégrer dans un même modèle des normes contradictoires.

Les éléments modélisés ont été intégrés pour répondre à une certaine norme métier. Permettre de les retirer lors de la modélisation d'une autre norme reviendrait soit à les enlever par erreur soit à les effacer pour cause de conflit avec la norme en cours de modélisation. Dans ce dernier cas, il serait erroné de croire disposer d'un modèle intégrant les deux normes. En cas de conflit, un choix doit être fait de manière à pouvoir modéliser et surtout suivre une seule des normes contradictoires.

Bien entendu, lors de l'édition des spécificités d'une norme, il est possible de revenir sur des erreurs de modélisation. Les suppressions sont gelées seulement une fois l'édition de la norme terminée.

3.4.4 Modélisation multi-métier

Jusqu'à présent, nous avons considéré qu'un projet était constitué d'un seul métier. Or, on voit par exemple que la production d'une notice spécifique ne fait pas partie en elle–même du métier des EPI. C'est pourquoi il est nécessaire d'élargir le nombre de métiers associés à un projet.

Il est rare que la conception d'un système ne concerne qu'un métier unique. D'après l'EIA-632, un système est la composition de produits finaux et de produits contributeurs. C'est d'ailleurs la force de l'ingénierie système que de permettre l'assemblage d'éléments distincts pour créer un tout cohérent et conforme aux attentes du client et des parties intéressées.

Les systèmes actuels font intervenir une multitude de métiers pour leur conception et leur réalisation. Dans l'automobile par exemple, l'architecture du système est proche des métiers utilisés : moteur, pneumatiques ou éclairages sont autant de produits finaux développés par des spécialistes. Cette répartition des métiers peut aussi se faire par fonction. Dans le logiciel par exemple, les fonctions de communication, de cryptage ou d'interface utilisateurs font intervenir des compétences spécifiques. De la même manière, les services sont une composition de produits finaux et de produits contributeurs.

Dans ce cadre, on peut se demander comment intégrer les aspects métier alors que chaque produit constituant le système peut être lié à un métier spécifique.

Une solution est d'employer le modèle métier au niveau des blocs de construction. **En associant un métier à un bloc de construction on dispose d'un moyen de modélisation simple et en cohérence avec les recommandations de l'EIA-632.** On rappelle qu'un bloc de construction est défini comme : « *La représentation du cadre conceptuel d'un système utilisé pour organiser les exigences, le travail, et les autres informations associées à l'ingénierie système. Un élément dans la décomposition structurelle du système* ». Il est donc logique de lui associer les éléments issus du métier. D'autre part, étant liée à la décomposition structurelle du système, cette association reste cohérente avec notre définition des métiers qui sont fortement liés à un produit.

Reste à traiter le cas de la répercussion des spécificités métiers sur des sous–systèmes. C'est le cas lorsque des sous–systèmes dépendent d'éléments liés à un système particulier. Par exemple, l'assurance qualité en automobile va fixer des contraintes spécifiques sur les composants qui ne seraient pas présentes s'ils n'étaient pas intégrés dans un véhicule.

Dans ce cas, notre recommandation est de maintenir un lien simple entre le bloc de construction de l'élément pris comme système à part entière et le métier qui lui est lié. Les éléments issus du niveau supérieur peuvent être transmis via les spécifications du système pour ses sous–systèmes.

Ces choix de représentation permettent de conserver des liens simples entre produits et processus. Ils ne sont, en un sens, que l'application des recommandations de l'EIA-632 puisque pour employer les modèles métiers nous les rattachons aux concepts définis dans le standard. Notre démarche étant de permettre une spécialisation des processus décrits tout en conservant et en respectant la cohérence des processus généraux.

Un autre avantage de ce choix de modélisation est de respecter le rythme du projet. Il serait irréaliste de penser connaître l'ensemble des métiers nécessaires en début de projet. Ceux–ci, issus des options et des choix de conception, ne seront découverts qu'au fur et à mesure de son déroulement. En associant les métiers au blocs de construction, disposer d'un modèle métier ne devient nécessaire qu'au moment où le choix d'un produit ou d'un service en tant que sous–système est effectué.

Le déroulement global d'un projet se compose donc d'une multitude de spécialisations des processus en fonction des métiers correspondant aux systèmes ou sous–systèmes en cours de développement. Comme on le voit sur la figure 3.9, les processus d'ingénierie sont spécifiques

à chaque système tout en conservant un comportement cohérent par rapport aux recommandations génériques.

3.5 La construction d'un modèle projet

Le modèle de projet est l'élément final du processus reposant sur la démarche de formalisation des normes d'ingénierie système que nous proposons. Il doit d'une part s'intégrer pleinement dans ce processus et d'autre part être suffisamment mature et concret pour servir aux intervenants du projet. Selon que l'on se place dans l'un ou l'autre de ces points de vue, nous allons voir ci–dessous quels sont les rôles et les contraintes de ce modèle de projet.

3.5.1 Rôle du modèle de projet dans la formalisation des processus d'ingénierie système

Notre proposition est de permettre une formalisation des processus des projets. Cette formalisation doit permettre de disposer :

- d'une méthode d'application des processus d'ingénierie système décrits dans des normes telles que l'EIA-632,
- d'une formalisation des pratiques et des règles spécifiques au domaine d'activité (modèle métier),
- de représenter les processus constituants d'un projet spécifique,
- d'un moyen de certifier la conformité des processus du projet par rapport aux grands standards de l'ingénierie système ainsi qu'à ceux issus de la spécialisation du métier,
- de réutiliser certains éléments sur des projets similaires.

En pratique, il s'agit, pour nous, de mettre à disposition des outils qui permettront de :

1. représenter les processus du projet en cours ;
2. valider ces processus par rapport
   - aux objectifs du projet,
   - aux contraintes spécifiques du projet,
   - aux règles de bonne conduite du projet identifiées au travers des modèles de standards et de métier,

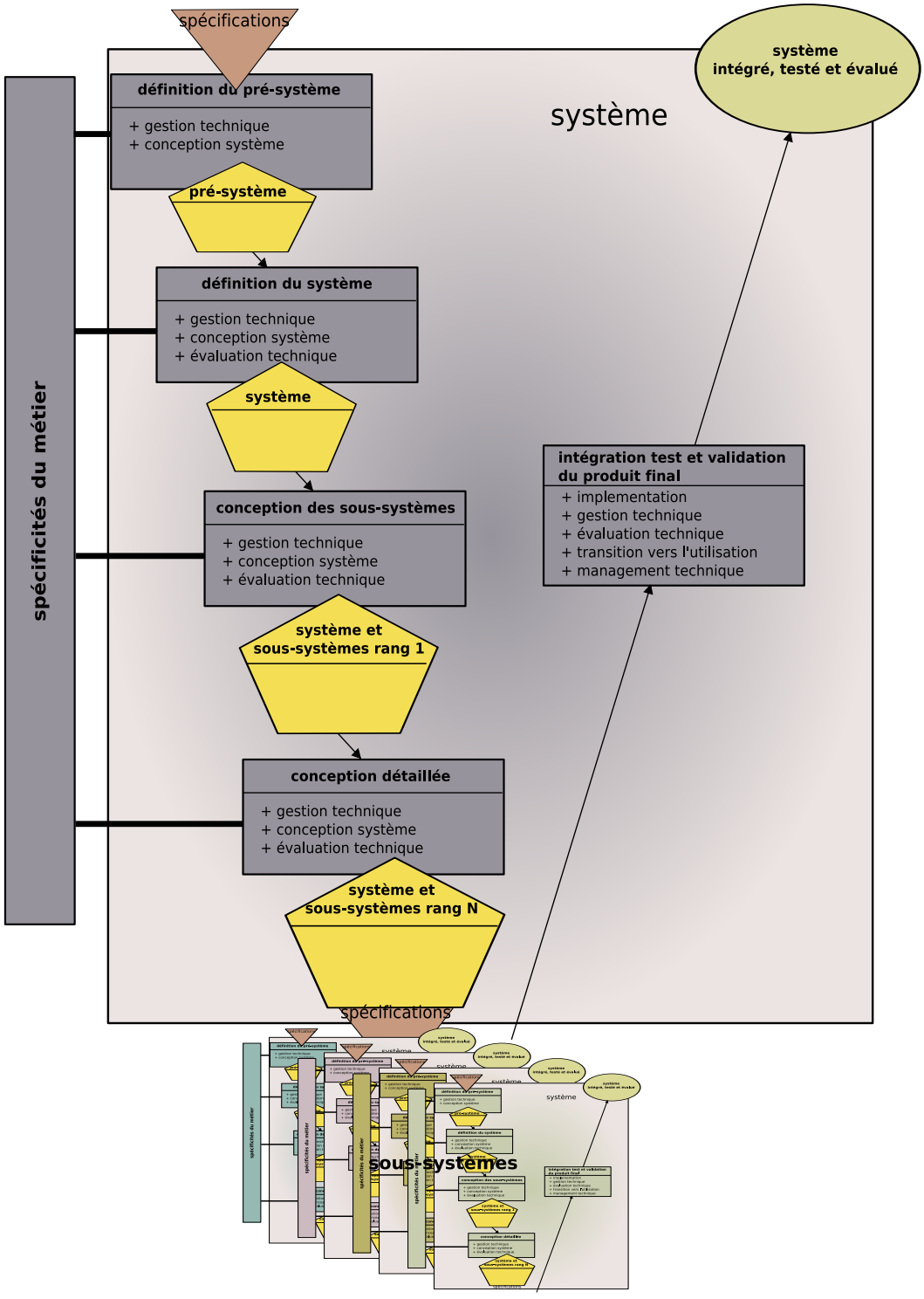


FIG. 3.9: *Impact des spécificités sur le cycle de vie. D'un point de vue système, le déroulement global du cycle de vie du système est conservé. En revanche, chaque processus employé est spécifique au métier considéré. Sur l'ensemble du projet, le comportement global est guidé par les recommandation d'IS avec des adaptations pour chaque système ou sous–système.*

et ce, tout au long du cycle de vie du projet.

Notre proposition repose sur des étapes de modélisation successives des processus constitutifs du projet. Des trois modèles nécessaires (standard, métier et projet), le modèle de projet est celui qui est le plus proche du projet tel qu'il sera mené. Il intègre les spécificités du projet et sert de référence à son déroulement. Il faut cependant garder à l'esprit qu'il s'agit d'un modèle de projet et non du projet lui–même. Il représente un déroulement prédictif du projet dans lequel sont pris en compte les règles expertes issues des modèles de standard et de métier.

Le projet proprement dit doit être considéré comme une instance du modèle de projet. Au fur et à mesure du déroulement du projet, celle–ci sera complétée jusqu'à représenter, à sa clôture, le déroulement intégral du projet (figure 3.10).

Les instances du projet intègrent les réalisations effectives du projet. Entre chaque version, de nouveaux éléments sont ajoutés et des ajustements sont faits. Chacune des instances est réalisée en conformité avec les contraintes et les règles de bonne conduite identifiées précédemment.

Le rôle du modèle de projet est double. D'une part, il doit permettre d'instancier les éléments représentés *i.e.* de conduire le projet. D'autre part, il doit permettre une validation du projet, c'est–à–dire, vérifier que les règles de bonne conduite et les contraintes identifiées précédemment sont bien respectées.

3.5.2 Rôle du modèle de projet dans la conduite d'un projet

Le modèle de projet doit posséder un caractère opérationnel qui n'est pas présent dans les modèles présentés précédemment. Dans la pratique, la complexité provient de l'interconnexion des différents métiers de l'entreprise nécessaire au déroulement du projet. Une fonction de l'entreprise peut attendre des informations de la part d'une seconde ou deux fonctions peuvent avoir des objectifs contradictoires. Par exemple, l'industrialisation est fortement dépendante de la fonction de conception car elle doit connaître les caractéristiques du produit. Or, le contour du produit va évoluer durant le projet, ce qui modifie les exigences d'industrialisation. De la même manière, l'industrialisation peut amener la conception à revenir sur des choix techniques valides vis–à–vis des exigences fonctionnelles mais difficilement compatibles avec les contraintes d'industrialisation du produit.

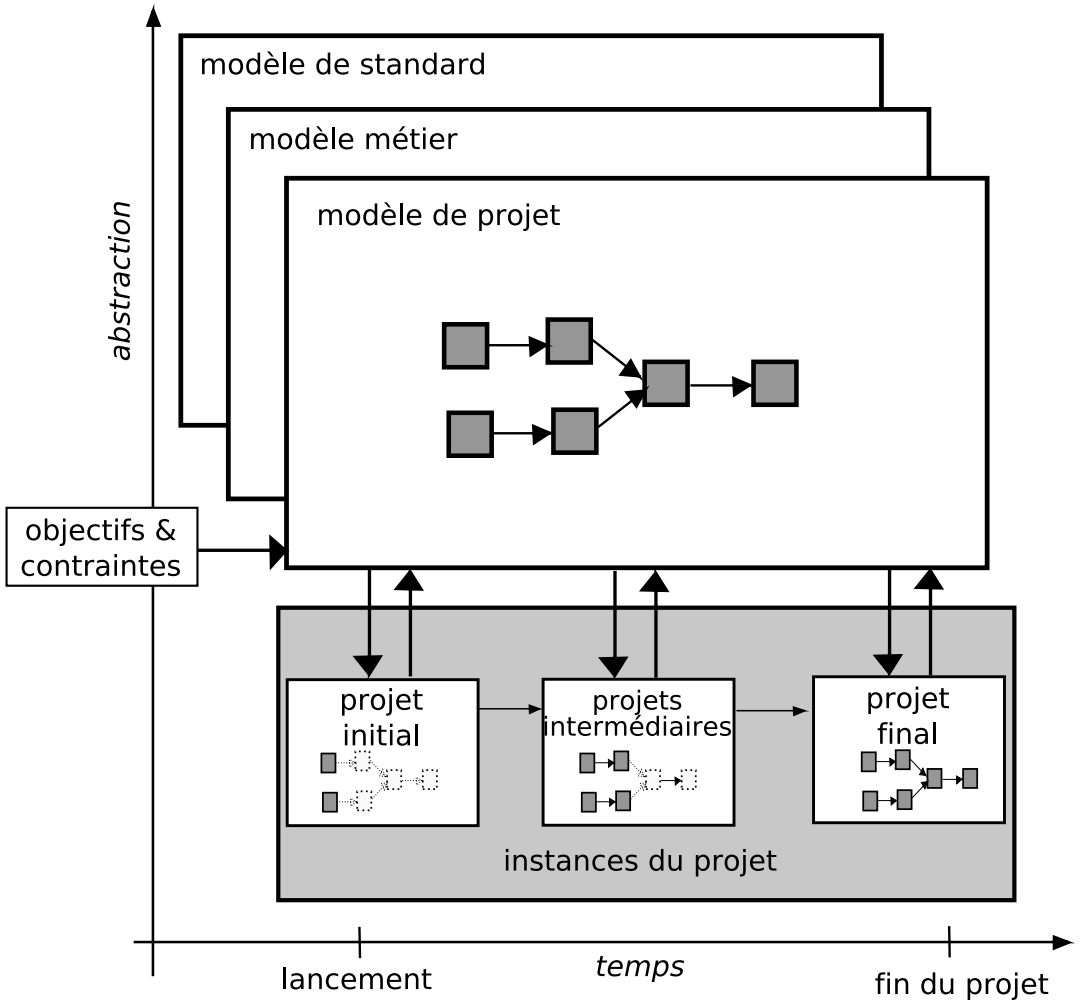


FIG. 3.10: *Évolution des instances du modèle de projet. Le modèle de projet sert de référence à la conduite du projet. Il intègre les éléments issus des modèles de standard et de métier ainsi que les objectifs et contraintes spécifiques au projet. En fonction de l'avancement du projet, les instances du projet évoluent jusqu'à représenter tout l'historique du projet de son lancement à sa clôture.*

Dans la pratique, on constate que les évolutions du projet se font à travers une représentation du produit commune aux fonctions de l'entreprise. Chacune des fonctions (conception, méthodes, qualité, *etc.*) a son propre point de vue sur le produit mais il est essentiel que tous les partenaires partagent une vision commune du produit. Cette représentation du produit peut être informelle ou, dans le cas de projets disposant de plus de moyens, être formalisée. Dans le cas d'une formalisation, le modèle du produit peut être organisé selon une description fonctionnelle et/ou organique. Cette représentation du produit commune aux différentes fonctions de l'entreprise va évoluer au cours du projet et devenir de plus en plus proche du produit qui sera finalement délivré. Il est de la responsabilité de chaque fonction de l'entreprise de s'adapter à ces évolutions pour fournir un service adapté à la vision actuelle du produit.

L'utilisation d'un modèle commun (de produit et/ou de processus) permet de planifier régulièrement des revues dans lesquelles sont impliqués l'ensemble des acteurs du projet. Au cours de ces revues, les évolutions prévues (et donc à terme le produit ou ses processus) sont présentées et soumises à (d'âpres) discussions. L'utilisation de revues pour le suivi de projet n'est pas nouveau. L'aspect novateur réside plutôt dans la formalisation des éléments abordés et des décisions prises au cours de la revue. Clairement, l'utilisation d'un modèle de projet ne se substitue pas à la concertation et à une communication directe entre les acteurs du projet, qui restent indispensables. **L'utilisation du modèle de projet doit être considérée comme une aide dans la gestion du projet qui permet de s'assurer d'une certaine conformité face aux recommandations des standards de l'ingénierie système ainsi que du respect de règles expertes issues du métier.**

3.5.3 Création du modèle de projet

Les techniques de construction du modèle de projet sont similaires à celles du modèle métier. On retrouvera le principe de construction incrémentale du modèle et les mêmes opérations de construction.

La principale différence avec le passage au modèle métier vient de la nature des spécificités introduites. Dans les transformations précédentes, il s'agissait d'intégrer les particularités liées au domaine d'activité considéré. Dans celle–ci, il s'agit de représenter les procédures spécifiques à un service de l'entreprise. Si deux services similaires de deux entreprises différentes partagent le même objectif, on constate généralement que ces services travaillent avec des règles de fonctionnement différentes. Tout comme les spécificités métier sont synthétisées dans des normes, les spécificités du projet sont répertoriées dans des notes de service et des

règles de fonctionnement internes et propres à l'entreprise. Ce sont ces pratiques que l'on cherche à intégrer dans le modèle de projet.

Cette décomposition en modèle métier et modèle projet permet, à des acteurs précis, de représenter *leurs* pratiques dans le contexte, plus général, du projet. On peut alors, sur la base du modèle de projet, confirmer ou remettre en cause ces pratiques en considérant leur intégration globale.

Dans un premier temps, les spécificités du projet sont donc progressivement introduites dans des modèles intermédiaires jusqu'à obtention du modèle de projet (figure 3.11).

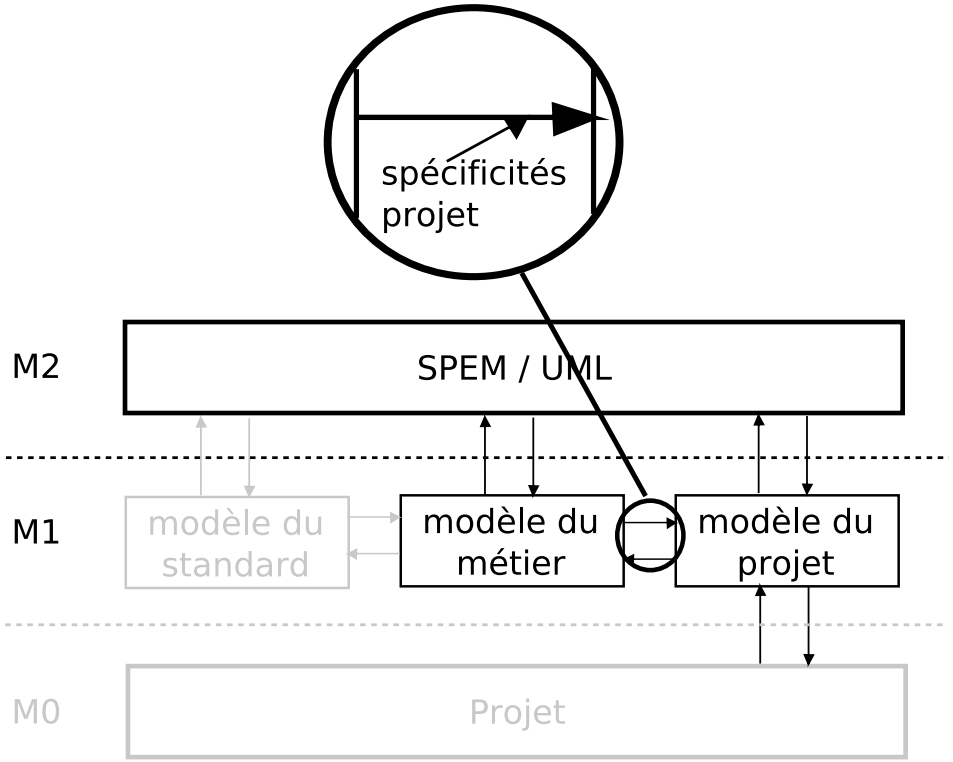


FIG. 3.11: *Passage du modèle de métier au modèle de projet. Comme lors du passage vers le modèle métier, cette étape passe par la transformation d'un modèle de référence, ici le modèle métier, vers un modèle spécialisé, ici le modèle de projet. Lors de l'opération de transformation, les spécificités du projet sont introduites dans le modèle. Le passage au modèle de projet a aussi pour objectif de préparer le suivi effectif du projet.*

Le modèle de projet diffère aussi du modèle métier par sa nature applicative. C'est sur la base du modèle de projet que se fera le suivi du projet et c'est pourquoi, au lancement d'un projet, les intervenants doivent disposer :

– des objectifs et contraintes du projet en termes d'exigences,

– d'un modèle métier dépendant de la solution logique envisagée pour le système.

La première ébauche du modèle de projet est donc logiquement constituée de l'agrégation des exigences formalisées avec des éléments issus du modèle métier.

On voit ici qu'il est nécessaire de disposer d'exigences formalisées avant de commencer la modélisation. Un premier traitement des exigences a donc été réalisé avant la modélisation du projet.

Le début effectif du projet, qui va comprendre entres autres des étapes de définition de la cible et des moyens alloués au projet, est donc dissocié de la modélisation du projet dans ses toutes premières phases. On ne peut modéliser un projet que sous réserve d'avoir au préalable formalisé les concepts qui le constitueront, c'est pourquoi la modélisation des projets ne peut se faire de manière précoce et doit résulter d'un travail de compréhension du projet, de ses objectifs et de ses contraintes.

Les exigences se décomposent généralement en deux catégories distinctes :

les exigences fonctionnelles : elles décrivent *ce que* le système *doit faire* et sont accompagnées d'attributs de qualité décrivant *à quel point* et *avec quelle qualité* il doit le faire. Parmi les attributs de qualité, on trouve :

- l'efficacité,
- l'efficience,
- la consommation de ressources,
- la performance,
- la disponibilité,
- la fiabilité,
- la sûreté,
- l'intégrité,
- la confidentialité,
- la sûreté de fonctionnement,
- la sécurité,
- la tolérance aux fautes,
- la convivialité,
- la maintenabilité,
- la flexibilité,
- la portabilité.

les exigences du projet : ce sont des exigences non–fonctionnelles qui décrivent plus le processus de développement que le système lui–même. Par exemple :

- les délais,
- le coût,
- certains produits contributeurs (par exemple les manuels).

Ces exigences, une fois identifiées, seront introduites dans le modèle comme les exigences initiales du projet et liées au bloc de construction de la couche supérieure. L'application des

processus d'IS consistera alors à transformer ces exigences initiales en solutions logiques, solutions physiques puis, enfin, système.

3.5.4 Problèmes ouverts

L'utilisation pratique d'un modèle de projet est suspendue à la résolution de divers problèmes. Outre les problèmes de réorganisation de l'activité même de l'entreprise autour de ce nouvel outil se posent des problèmes techniques.

Une des questions principales posées par l'utilisation des modèles de projet est celle de leur validité et leur vérification. La section 4.2 sera d'ailleurs entièrement consacrée à cette question.

À ces problèmes de validité et de vérification, s'ajoutent des problèmes de répartition de responsabilité dans la gestion des modèles de projet. La figure 3.12 donne un exemple de projet dans lequel se posent de tels problèmes.

L'organisation en « cascade » des produits est difficilement synthétisable au niveau de la représentation des processus du projet. D'une part, les produits constituants peuvent être liés à un métier différent sans rapport avec le métier du système et nécessitent alors des processus de développement spécifiques. D'autre part, des contraintes sur les produits constituants et leurs processus peuvent être héritées du produit principal et de son processus. Dans ce cas, il peut être nécessaire de lier les deux processus entre eux. La réponse la plus simple à ce problème est de considérer chaque produit contributeur comme un projet indépendant. Chaque projet est représenté et suivi via son propre modèle. Cette solution a l'avantage de permettre de représenter le développement des produits contributeurs sous un point de vue de haut niveau dans le modèle du projet principal et de disposer d'autre part d'une description fine du processus du produit contributeur. C'est la modélisation multi–métier présentée au § 3.4.4.

Cette séparation des processus permet de ne pas complexifier à outrance le modèle de projet. Cependant, il est nécessaire d'inclure les modèles des projets de produits contributeurs dans le modèle de projet dans le cas d'un recouvrement des contraintes entre processus produit et processus de produit contributeur. Le modèle de projet du produit contributeur devient alors un sous–ensemble du modèle de projet.

Cette solution, réalisable d'un point de vue technique, pose des problèmes de répartition des responsabilités dans le cas où des équipes différentes sont à la tête des projets. Il serait logique que chaque équipe soit responsable de son modèle. Le modèle du projet du produit

contributeur (sous la responsabilité de l'équipe 2) étant inclus dans celui du produit principal (équipe 1) il est difficile d'établir des responsabilités simples vis–à–vis des modèles. Si l'équipe 1 est nommée responsable du modèle de projet du produit contributeur, elle va empiéter sur le domaine de l'équipe 2. Inversement si l'équipe 2 est responsable, elle va modifier indirectement le modèle du produit principal sous la responsabilité de l'équipe 1.

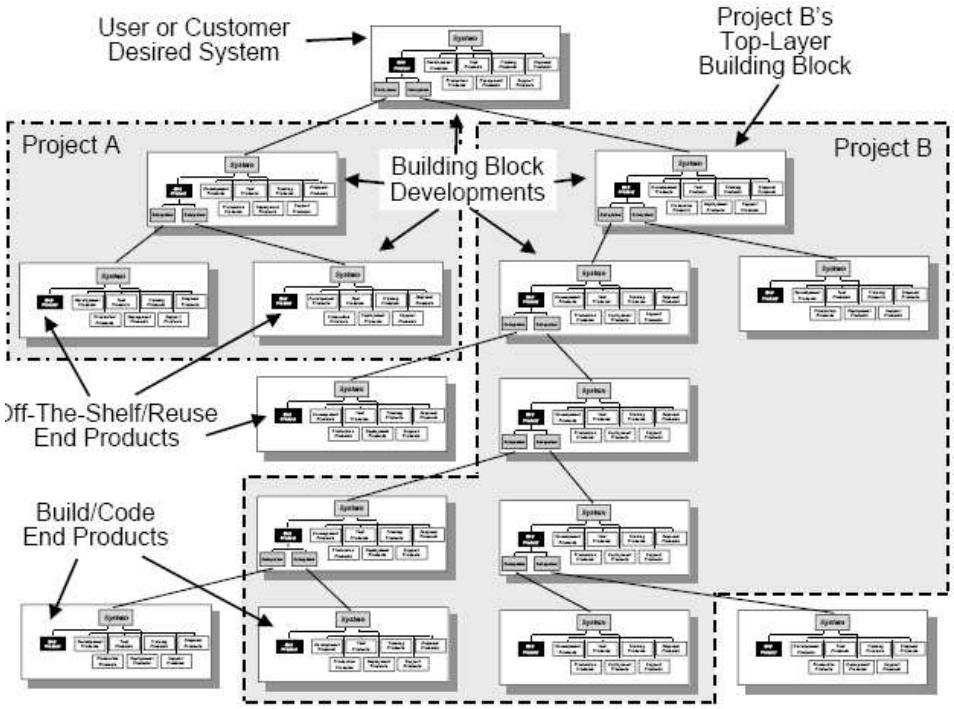


FIG. 3.12: *Illustration des problèmes de représentation et de responsabilité. Si l'on prend comme exemple l'illustration du concept de bloc de construction donné par l'EIA-632 on voit clairement apparaître ces problèmes.*

Problème de représentation car la solution proposée nous permet de représenter uniquement le développement d'un produit par projet. On constate cependant en pratique le regroupement du développement de plusieurs produits dans le même projet. Notre modélisation ne permet pas de représenter de telles choses. Les liens entre développement de produit et développement de produits contributeurs sont alors délicats à intégrer et à valider en cours de projet.

Problème de responsabilité sur cet exemple car le développement des deux produits contributeurs sont sous la responsabilité des équipes A et B. On peut intégrer leurs modèles de projets respectifs dans le modèle de projet global. Dans ce cas, l'évolution du modèle est–elle de la responsabilité de A, de B ou de l'équipe du produit principal ? Comment vont vivre ces modèles ?

3.6 Conclusion

Les propositions de ce chapitre explicitent la démarche de spécialisation du modèle générique vers le modèle projet. Cette démarche est une démarche d'enrichissement pas à pas du modèle générique à partir des normes métiers puis des choix projets. Nous avons aussi, mis en évidence les opérations d'ajout, de spécialisation et d'affinage valables pour les deux étapes de la spécialisation : métier et projet.

Pour résumer, notre conception du problème repose sur les éléments suivants :

- maîtriser les processus décrits par les normes d'ingénierie système permet de répondre à de nombreux problèmes actuels,
- pour maîtriser ces processus, il est nécessaire de modéliser les projets de l'entreprise,
- cette modélisation est possible grâce à des modèles formalisés spécifiques au métier du projet,
- ces modèles spécifiques peuvent être construits à partir du modèle générique d'une norme d'IS qui permet de s'assurer de la conformité au standard et ainsi de contrôler l'évolution du projet.

En définissant une démarche de construction du modèle de projet nous fournissons :

- une méthode d'application pratique des normes d'IS,
- un moyen de les adapter aux différents niveaux du projet,
- un moyen de faciliter la certification de la conformité des processus du projet par rapport au standard,
- la possibilité de définir des éléments de modélisation réutilisables sur des projets similaires.

Nous rendons ainsi opérables les processus décrits dans les grandes normes d'ingénierie système. Nous espérons par ce biais donner les moyens aux entreprises de mieux suivre et organiser leurs projets en diminuant les risques d'erreurs liées aux interconnexions entre processus.

Chapitre 4

Compléments à la modélisation des activités en ingénierie système

« Ce n'est pas assez d'avoir l'esprit bon, mais le principal est de l'appliquer bien. »

René Descartes

4.1 Introduction

Dans les chapitres précédents, nous avons montré :

- que la formalisation des normes d'ingénierie système est possible,
- que les processus qu'elles décrivent peuvent être adaptés à leur contexte d'application par une spécialisation des modèles.

En faisant cela, nous avons défini une démarche d'ingénierie originale qui répond aux manques de formalisation actuels dans l'organisation des projets.

Il s'agit maintenant d'organiser des réflexions conjointes à cette démarche pour la rendre pleinement opérationnelle. Pour qu'elle soit, un jour, appliquée avec succès dans le monde industriel, il est nécessaire de venir la compléter par des outils et par des réflexions qui favoriseront sa mise en œuvre.

Dans cette optique, nous avons organisé nos réflexions autour de trois questions majeures :

- la question de la validité des modèles : comment s'assurer de la cohérence de ce que l'on modélise ?
- la question de la mise en application : qu'arrive–t–il lors du passage d'un modèle abstrait de processus à sa mise en œuvre concrète ?
- la question des améliorations possibles : peut–on faire évoluer la méthode vers une réelle ingénierie système dirigée par les modèles ?

Chacun de ces axes sera abordé dans ce chapitre. La première section sera consacrée à la validité des modèles : validité des modèles en tant qu'entités propres et validité des relations inter–modèles. La seconde section illustrera l'application de cette méthodologie dans le cadre d'un projet universitaire. Il illustrera comment peut être mise en œuvre l'activité d'ingénierie système et les avantages de cette méthodologie sur le suivi de projet. Enfin, la dernière section donnera des pistes d'amélioration de la méthodologie pour la faire tendre vers une ingénierie système guidée par les modèles. Nous montrerons que cette méthodologie peut être employée non seulement en vue d'un suivi de projet mais qu'elle peut aussi être utilisée dans les phases préliminaires des projets. Elle peut servir de base à une identification précoce des besoins en langages de modélisation, en transformations de langages et être à la base du développement d'outils support du projet.

4.2 Validation et vérification des modèles de processus

L'approche que nous proposons dans cette thèse est centrée sur l'utilisation de modèles. Dans ce cadre, s'assurer de leur validité est essentiel.

La validation et la vérification sont des domaines très actifs de la recherche actuelle. Les questions qu'elles soulèvent dans le cadre de l'ingénierie des modèles dépassent largement le cadre de cette thèse. En fait, chaque modèle, méta–modèle ou relation du processus de modélisation présenté au chapitre précédent pourrait être sujet à des études. C'est le cas par exemple de la validité des méta–modèles UML [SV06] et SPEM [CCCC06b] qui sont tous deux imparfaits et pour lesquels on voit régulièrement apparaître des contributions visant à les améliorer.

Dans le cadre de nos travaux, nous nous sommes limités à l'étude de la validation de modèles avec pour objectifs :

1. de démontrer que le processus proposé est cohérent,
2. d'illustrer les possibilités offertes par la formalisation.

Dans ce chapitre, on considèrera deux types de validations distinctes :

la validation d'un modèle : le modèle est considéré indépendamment de son environnement et l'on cherche à s'assurer de sa bonne formation et d'une cohérence intrinsèque.

la validation des relations entre modèles : le modèle est mis en relation avec le modèle à partir duquel il est obtenu : on cherche à s'assurer de la conservation de propriétés après transformation.

Cette section présentera successivement ces deux types de validation : la validation d'un modèle dans la section 4.2.1 et la validation inter–modèles dans la section 4.2.2. Enfin, la section 4.2.3 présentera les moyens permettant la vérification de modèles.

4.2.1 Validation d'un modèle

Dans cette section, on s'intéresse à un modèle en tant qu'entité propre. L'étude des relations entre modèle de standard, modèle métier et modèle de projet se fera dans la section suivante.

La construction d'un modèle est un processus délicat : des erreurs ou des inconsistances peuvent être introduites lors de sa construction et éventuellement se reporter dans le modèle suivant. L'introduction d'une erreur est d'autant plus facile que les éditeurs actuels ne permettent ni de s'assurer de la sémantique du modèle, ni même de sa conformité au méta–modèle.

Il est tout à fait possible de construire un modèle ne respectant pas son méta–modèle. Cette conformité au méta–modèle est un premier point de la validation. Or, même conforme à son méta–modèle, un modèle peut être incohérent et donner représentation incomplète ou absurde. Dans le cas de SPEM, le modèle d'une seule activité utilisant et produisant le même produit de travail est incohérent. Soit le produit de travail existe déjà et il n'est pas nécessaire de réaliser l'activité. Soit l'activité ne pourra jamais être réalisée en l'absence du produit de travail. En revanche, un tel modèle modèle est bien conforme au SPEM !

On complète donc la validation de la conformité au métamodèle par une vérification de propriétés qui assureront la cohérence du modèle.

4.2.1.1 Conformité au méta–modèle

On a vu dans la section 2.2.2 que l'IDM repose sur la relation de conformité d'un modèle à son méta–modèle.

En pratique cependant, garantir cette conformité n'a rien d'évident. Les éditeurs actuels n'incluent généralement pas une vérification exhaustive de la conformité des modèles créés. C'est d'autant plus vrai dans le cas de SPEM dont l'OMG a défini la syntaxe abstraite mais dont personne n'a défini de syntaxe concrète. Ce méta–modèle est couvert par peu d'éditeurs et, lorsqu'il l'est, est considéré en tant que profil UML et non comme méta–modèle à part entière.

Déduire la conformité d'un modèle à son méta-modèle se fait en vérifiant les propriétés énoncées dans le méta–modèle. Celles–ci peuvent être déduites du méta–modèle par analyse des diagrammes et des contraintes OCL qui le composent ou ajoutées aux méta–modèles, comme dans [SV06] pour UML et [Com05, CCCC06b] pour SPEM, pour améliorer la cohérence des modèles exprimés.

La figure 4.1 illustre comment des règles peuvent être déduites des diagrammes du méta–modèle. On peut identifier, par exemple, la règle suivante : « un processus ne peut être gouverné que par un cycle de vie maximum ».

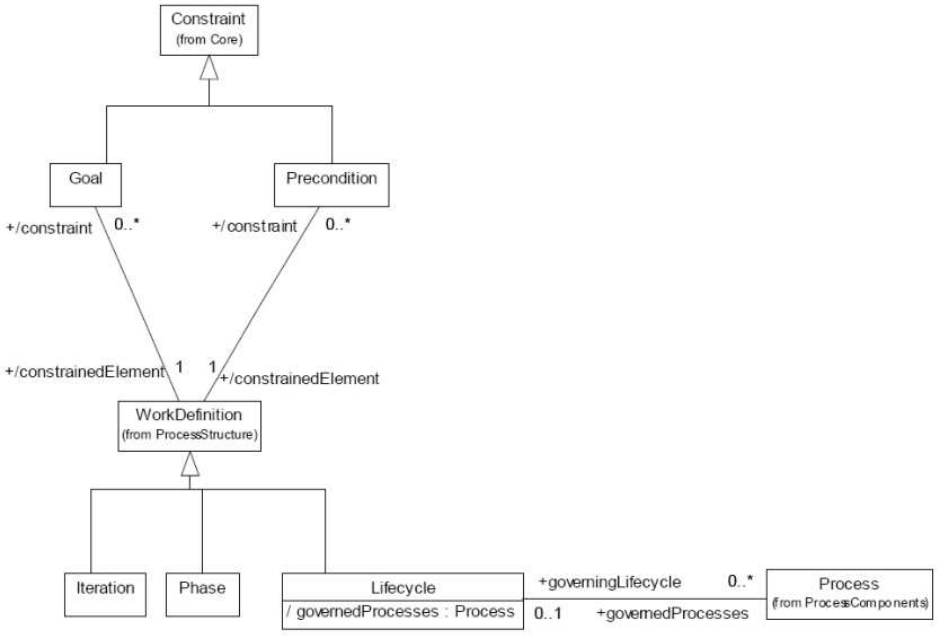


FIG. 4.1: *Package cycle de vie d'un processus de SPEM.*

Ces règles sont complétées par des contraintes OCL comme, par exemple, la contrainte ci–dessous qui exprime qu'un cycle de vie contient uniquement des phases.

```
context Lifecycle inv:
self.subWork->forall(ph | ph.oclIsKindOf(Phase))
```

4.2.1.2 Cohérence du modèle

La conformité au méta–modèle vérifie qu'un modèle est correctement écrit sans s'intéresser à la cohérence de ce qu'il exprime. En général, la cohérence d'un modèle est vérifiée en s'assurant de la présence de propriétés dans le modèle. Ces propriétés sont exprimées sous forme de règles de cohérence.

Propriétés des processus

Combemale [CCCC06a] identifie quatre types de propriétés pouvant être définies pour des processus :

propriétés structurelles : elles permettent de structurer sémantiquement le procédé.
Exemple : « *La réalisation d'une activité ne peut pas être assistée par le rôle qui en a la responsabilité* » ;

propriétés de causalité : elles expriment la temporalité et donc la dynamique sans toutefois quantifier le temps.
Exemple : « *Pour chaque itération, les activités liées aux exigences doivent être réalisées avant celles liées à la conception* » ;

propriétés temps réel : elles offrent une temporalité quantifiée au sein du procédé.
Exemple : « *Le temps compris entre la fin de rédaction du cahier des charges et la livraison du premier prototype ne peut excéder 3 mois* » ;

propriétés opérationnelles : elles permettent l'expression de la sémantique opérationnelle au sein du modèle de procédé.
Exemple : « *Au démarrage d'une phase, chacune des (instances des) activités doit être affectée à (une instance d') un rôle* ».

Limitations

Lors de la validation d'un modèle de processus, on doit prendre en compte les limitations suivantes :

1. on ne peut pas changer le point de vue pris par le méta–modèle, *i.e.* le modifier pour faire apparaître de nouvelles propriétés ;
2. on ne peut valider que le modèle du processus et non pas le processus qui sera effectivement réalisé.

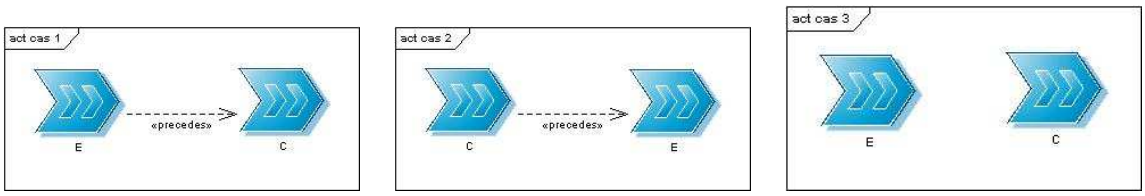
Dans le cas des processus, la validation est plus délicate que dans le cas des logiciels. Pour ces derniers, les modèles sont très proches du code et l'essentiel des informations nécessaires sont présentes dans un modèle. Dans le cas des processus, les instances sont plus éloignées du modèle. Le piège serait de vouloir descendre au niveau de la planification des tâches pour laquelle le méta–modèle n'est pas adapté. Il serait nécessaire d'ajouter de nouvelle propriétés telles que dates de début, dates de fin, ressources consommées, *etc.* Il faut donc se contenter de la vision haut niveau du méta–modèle.

Une autre particularité des processus est que l'on ne peut vérifier leurs propriétés qu'une fois qu'ils sont achevés. Par exemple, comment certifier *a priori* que les propriétés temps réel seront bien respectées ? Les contraintes exprimées dans le modèle de processus ne sont que des indications quant à la manière de mener le projet à bien. Un projet étant soumis à des aléas, rien ne peut certifier que les contraintes ne seront pas violées et que le déroulement du projet sera en tout point identique à celui prévu.

Il est important de distinguer la validation du modèle de processus et la validation du processus lui–même. Prenons l'exemple de la propriété de causalité exprimée ci–dessus. La figure 4.2 donne trois modèles pour lesquels cette propriété doit être vérifiée. Dans le cas de la figure 4.2(a), la propriété est vérifiée sur le modèle du fait de la relation de précédence entre les actions. La relation devrait être vérifiée dans le processus correspondant.

Sur la figure 4.2(b), la relation de précédence est inversée : la propriété n'est donc pas vérifiée pour le modèle et ne devrait pas non plus l'être dans le processus. Dans le cas de la figure 4.2(c), aucun ordre de réalisation n'est précisé : la propriété n'est donc pas vérifiée dans le modèle car il n'exprime pas l'obligation pour les activités liées aux exigences d'être réalisées avant celles liées à la conception. En revanche, on ne peut se prononcer sur la présence ou non de cette propriété dans le processus car l'ordre de réalisation est *a priori* indéterminé.

Une validation exhaustive d'un modèle de processus demanderait de pouvoir intégrer dans les règles des éléments provenant des trois couches suivantes : méta–modèle, modèle et instances. Or les instances sont justement indéterminées avant exécution du processus. On se



(a) Modèle 1 (b) Modèle 2 (c) Modèle 3

FIG. 4.2: *Exemples de modèles pour la vérification de propriété de causalité. Les activités notées* E *sont liées aux exigences et les activités notées* C *sont liées à la conception.*

contentera donc de travailler avec des règles construites à partir d'éléments du modèle et du méta–modèle.

Propriétés pour l'ingénierie système

Même en se limitant à la validation des modèles, sans pousser jusqu'à la validation des processus eux–mêmes, une démarche de validation est riche.

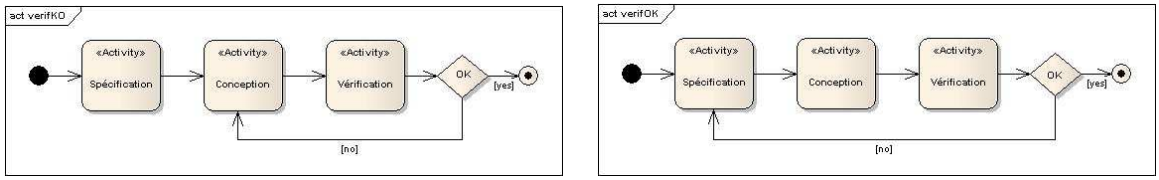
Contrairement à la conception de systèmes, pour laquelle l'aspect temporel est essentiel, on ne s'intéresse guère aux dates effectives de réalisation des activités. La philosophie de l'IS est plus orientée autour de l'obtention d'une cohérence des processus. Ce sont les propriétés garantissant cette cohérence et le bon déroulement d'un projet que nous rechercherons dans les modèles.

Prenons par exemple, le concept de vérification qui est au centre de l'IS. Dans les processus d'IS, des étapes de vérification sont régulièrement introduites et permettent de revenir, si besoin est, sur des étapes du processus. Une bonne propriété d'un modèle de processus d'IS pourrait être : « *il doit être possible de revenir sur chacune des étapes du processus* ». Notons que cette propriété est à l'opposée de ce qui peut être exprimé pour un système car elle demande d'introduire des cycles.

Sur l'exemple 4.3(a) de la figure 4.3, cette propriété n'est pas vérifiée. Une fois passée l'étape de spécification il n'est plus possible d'y revenir. En revanche, sur le modèle 4.3(b) il est possible de remettre en cause des spécifications avant de recommencer la conception.

4.2.1.3 Conclusion de la cohérence d'un modèle

On peut valider un modèle de processus selon deux axes :



(a) Propriété d'accessibilité pour la validation non vérifiée

(b) Propriété d'accessibilité pour la validation vérifiée

FIG. 4.3: *Exemples de vérification de propriété sur un modèle de processus d'IS.*

– D'une part en vérifiant qu'il correspond bien à son méta–modèle, *i.e.* qu'il est bien exprimé.
– D'autre part en vérifiant que des propriétés définies par l'utilisateur sont bien présentes dans le modèle, *i.e.* qu'il exprime bien ce que l'on désire.

L'utilisation de propriétés dans les processus, qu'il s'agisse des processus d'IS ou de processus d'entreprise, n'a pas été explorée. Il serait intéressant de pousser cette démarche en l'appliquant notamment à la correction ou à la rédaction des normes d'IS ou des grands processus d'entreprise.

4.2.2 Validation de relations inter–modèles

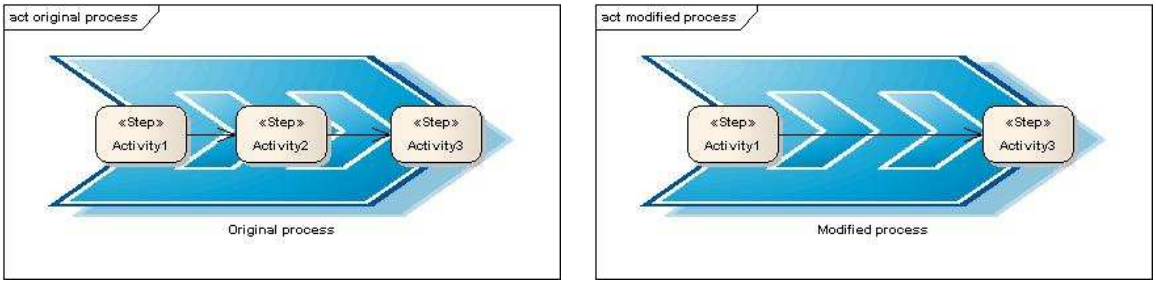
Nous avons vu comment vérifier un modèle. Cependant, notre proposition repose sur l'hypothèse de la conservation des propriétés de l'ingénierie système tout au long de la spécialisation des modèles de processus. Il est donc indispensable de s'assurer de la transmission de ces propriétés lors des opérations de transformations de modèles. On s'intéressera donc, dans cette section, à la validation de propriétés entre modèles.

Là encore, il convient de se distinguer des approches logicielles. Dans ce domaine, les validations entre modèles telles que les travaux de [MVDS05] concernent des versions d'un même modèle qui évoluent dans le temps. Le programme décrit reste identique mais sa représentation va évoluer. Dans notre cas, il s'agit de représenter trois éléments distincts : les processus recommandés par les normes d'ingénierie système, ces processus appliqués dans un contexte particulier fixé par le métier et, enfin, les processus spécifiques à un projet donné.

On a vu, dans la section 3.4.3, que le nombre d'opérations permises pour passer d'un modèle à l'autre était limité, cette limitation est due à la nécessité de conserver une cohérence entre les modèles. Les propriétés inter–modèles vont s'assurer du respect de ces règles élémentaires. Leur rôle ne s'arrête cependant pas là ; elles permettent aussi de vérifier des

propriétés de construction non–exprimées jusqu'alors.

La figure 4.4 illustre une propriété simple entre modèles : « *Toutes les activités du modèle original doivent être présentes dans le modèle modifié* ». Dans cet exemple, on observe que l'activité deux du modèle initial n'est plus présente dans le modèle modifié et donc que le modèle modifié n'est pas conforme à nos attentes.



(a) Processus original (b) Processus dérivé

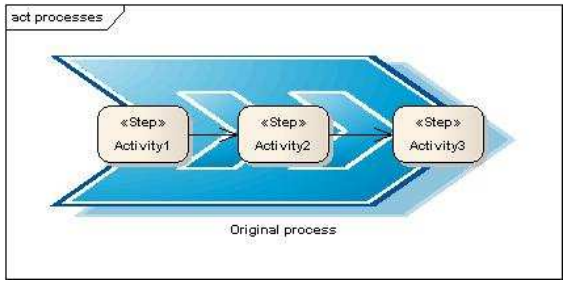
FIG. 4.4: *Exemple simple de validation inter–modèles.*

La figure 4.5 illustre une propriété plus complexe. Sur cette figure, le diagramme 4.5(a) représente les processus avant transformation et les diagrammes 4.5(b) et 4.5(d) l'état de ce processus après transformation. Ils sont identiques dans les deux cas de figure présentés. Dans chacun d'entre eux, les opérations élémentaires ont été remplacées par des activités. L'ordre de réalisation reste inchangé sur ce diagramme. On constate cependant que le comportement global peut être affecté par les activités qui ont été affinées. Dans le premier cas, l'activité 1 sur le diagramme 4.5(c) se compose uniquement d'opérations élémentaires. Le comportement global est donc inchangé. En revanche, sur le diagramme 4.5(e), on constate que des appels aux activités 2 et 3 sont effectués. En conséquence, non seulement ces activités se dérouleront en parallèle au lieu de se succéder, mais elles seront effectuées deux fois car elles font encore partie du processus principal.

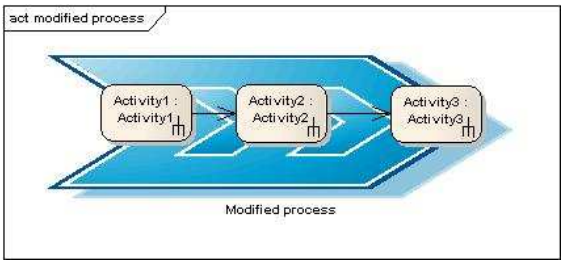
Une règle empêchant un tel comportement peut être : « *les CallBehavior ne peuvent être faits que pour les subWork du parentWork du WorkDefinition considéré* ». En d'autres termes, on limite les appels aux processus de même niveau ou de niveaux inférieurs.

4.2.3 Moyens d'aide à la validation

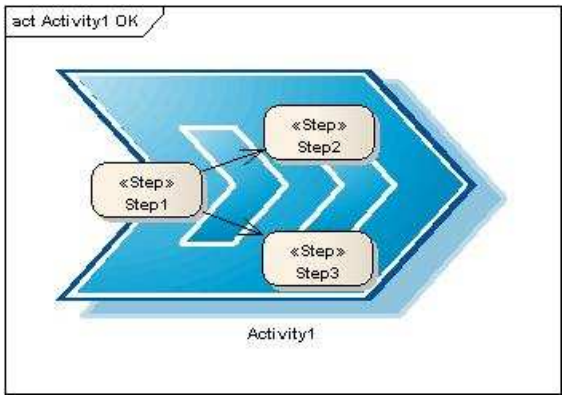
Jusqu'à présent, nous avons présenté les règles de cohérence sous leur forme en langage naturel. La vérification des modèles doit cependant être la plus automatisée possible. Il est



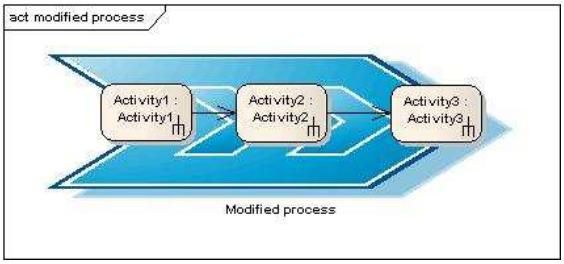
(a) Processus original



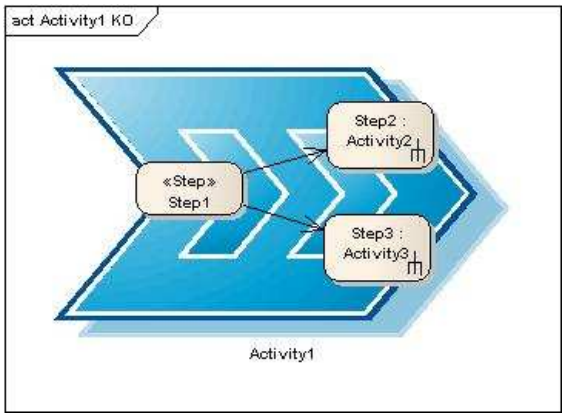
(b) Processus dérivé (cas 1)



(c) Activité correctement affinée (cas 1)



(d) Processus dérivé (cas 2)



(e) Activité affinée de manière incorrecte (cas 2)

FIG. 4.5: *Exemple de modification du comportement global par affinage des activités. On observe deux cas de processus issus du même processus original. Dans le premier cas, l'activité 1 se compose uniquement de « step » et ne vient pas modifier le déroulement prévu. Dans le second cas, elle appelle les activités 2 et 3 qui ne sont plus réalisées en série mais en parallèle puis ré–appliquées une fois l'activité 1 terminée.*

en effet difficile de réaliser cette activité manuellement étant donné le nombre de règles et la complexité des modèles manipulés. Dans cette section, nous montrerons comment formaliser ces règles de cohérence et les vérifier de manière automatique.

Les activités orientées validation de modèles sont essentiellement liées à UML. C'est autour de ce langage que se focalise l'essentiel des activités de recherche dans ce domaine. Nous présenterons, dans un premier temps, le principe de la vérification des modèles UML puis nous montrerons comment nous avons adapté l'une de ces techniques à la vérification de modèles de processus.

4.2.3.1 Principes de la vérification des modèles

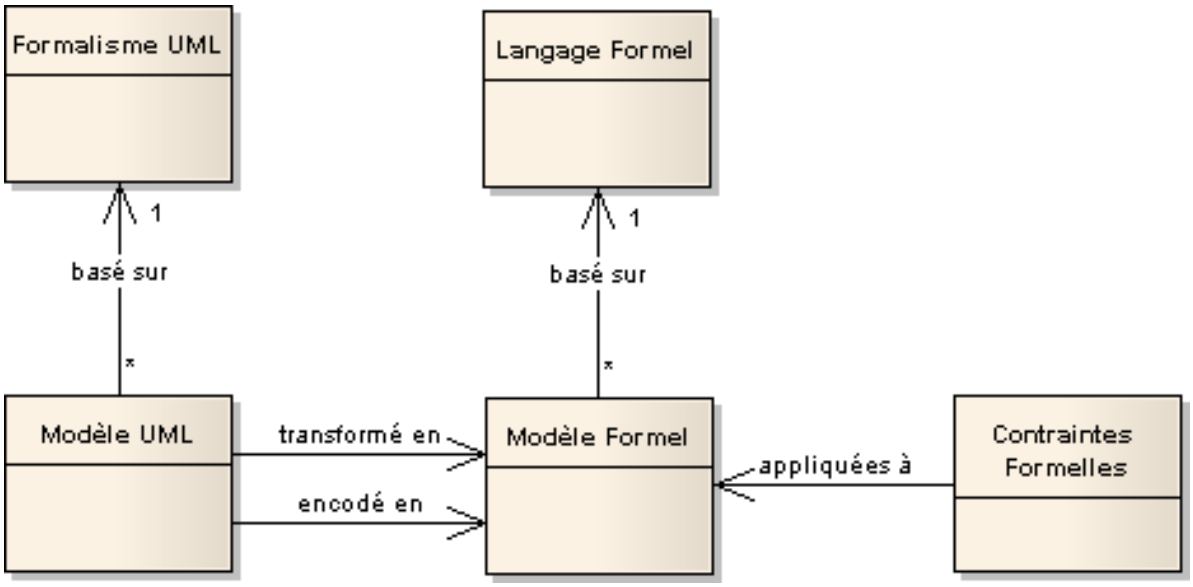
Les diagrammes de la figure 4.6 illustrent les principes de la vérification des modèles UML. Le formalisme UML étant graphique, il est nécessaire de convertir les modèles UML en modèles formels basés sur un langage formel. A ce stade, on distingue l'encodage des modèles et la transformation de modèles (diagramme 4.6(a)). Lors d'un encodage, l'intégralité des informations sont transmises au modèle formel alors qu'une transformation peut être source d'une perte d'information.

Une fois formalisé, il est possible d'appliquer des contraintes formelles sur le modèle. Ces contraintes formelles sont une formalisation des règles exprimées dans les sections précédentes. Comme on l'a vu, on distingue deux types de règles de cohérences : les règles de cohérence induites par le méta–modèle et les règles de cohérence exprimées par le langage formel. Cette distinction est représentée figure 4.6(b).

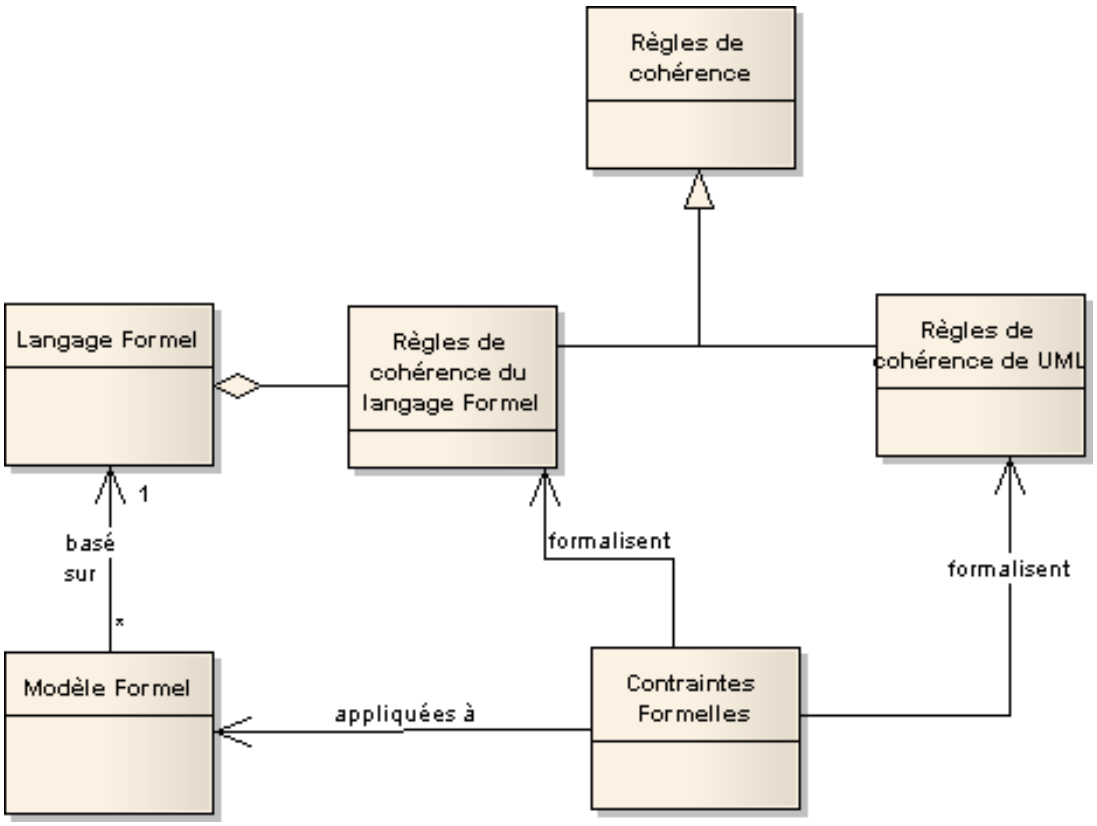
4.2.3.2 Vérification de modèles de processus

Si de nombreuse approches ont été développées et testées pour les modèles UML (checking, monitoring, construction ou analyse [HS04]) celles destinées à la vérification des processus sont beaucoup plus réduites. Si le processus est pris en compte, c'est généralement pour mieux valider le modèle UML qu'il génère ; la représentation du processus de développement guide la validation [BBLC+03, LMO05].

En ce qui concerne la validation des modèles de processus même, il faut se rapporter aux travaux de Benoît Combemale [Com05, CCCC06a, CCCC06b] qui définit certains types de propriétés des processus et explore quelques pistes de validation.



(a) Formalisation des modèles UML



(b) Règles de cohérence

FIG. 4.6: *Vérification de la cohérence des modèles UML.*

En ce qui nous concerne, nous avons choisi d'adapter une technique de vérification de modèles UML à la vérification de modèles de processus. Nous nous sommes inspirés des travaux de Hugues Malgouyres sur la vérification de la cohérence [Mal06, MM06].

Ses travaux reposent sur l'utilisation de la programmation logique comme langage formel utilisé lors de la validation. L'originalité de son approche est de transformer le méta–modèle UML en faits et règles logiques avant d'inclure dans la base de faits les faits et règles correspondants au modèle à vérifier. L'intérêt de son approche est que son principe peut être étendu à d'autres méta–modèles. En introduisant le méta–modèle SPEM, ou plus précisément le méta–modèle UML avec son profile SPEM, en lieu et place du méta–modèle UML on dispose d'un outil de validation logique de modèles de processus. La documentation utilisateur de cet outil est donnée dans les annexes de ce mémoire.

4.2.3.3 Principe de la vérification des modèles de processus

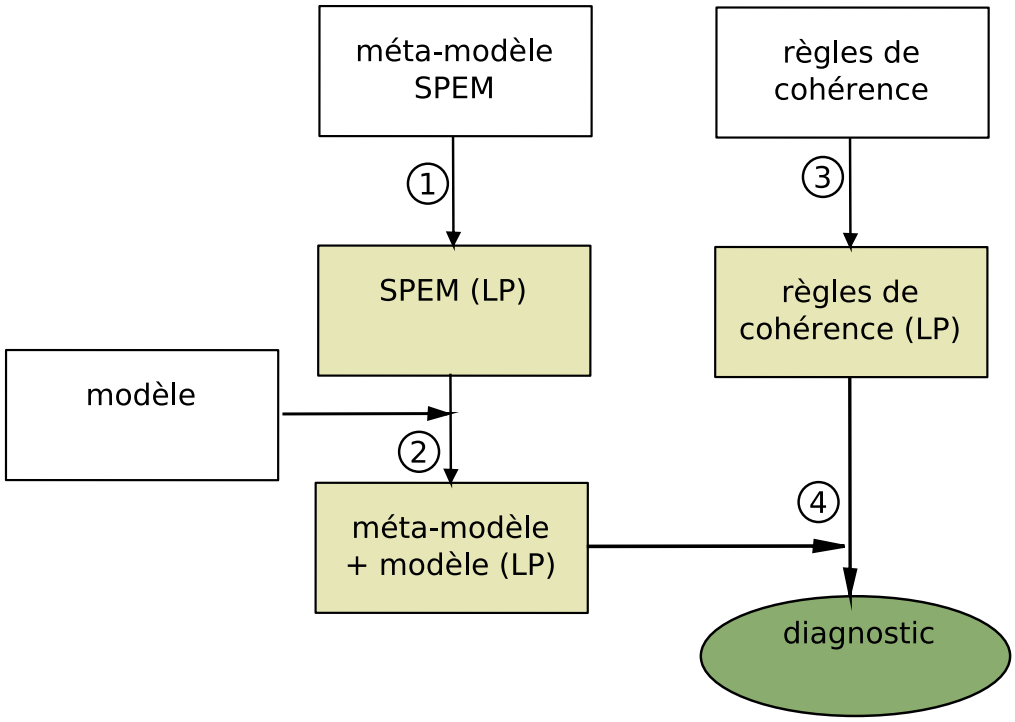
Le principe retenu pour la validation des modèles de processus reprend l'architecture de celle proposée par Malgouyres du moins dans le cas de la vérification d'un modèle unique. Le principe est identique. La différence principale vient du fait que l'on ne travaille plus à partir du méta–modèle d'UML mais du méta–modèle de SPEM.

La figure 4.7 illustre les principes de la vérification d'un modèle de processus et d'une vérification inter–processus. Dans le cas de la vérification d'un processus, représentée figure 4.7(a), on distingue quatre étapes :

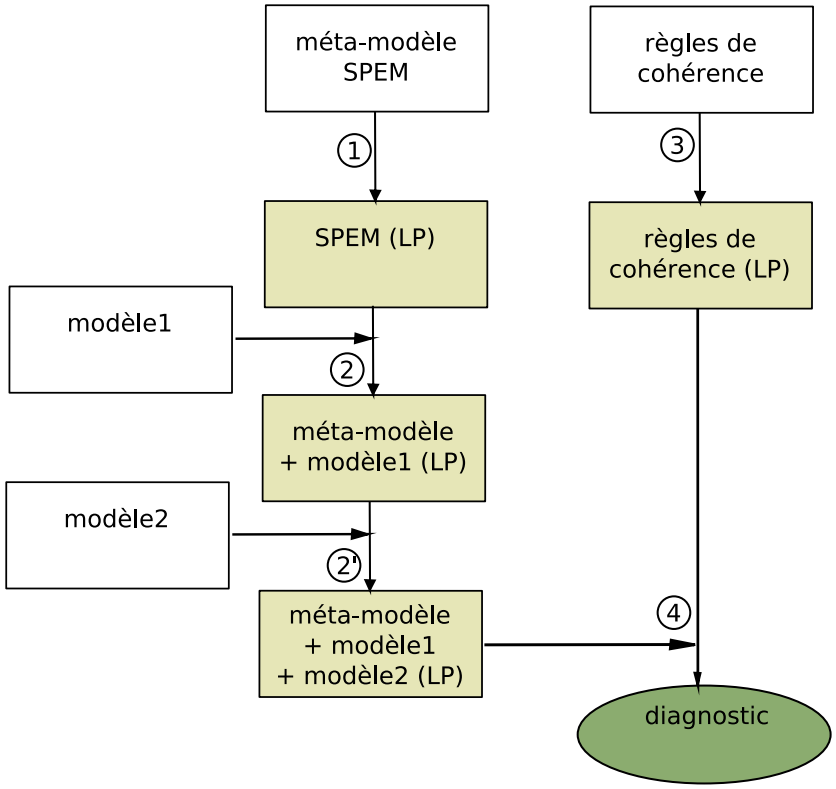
1. transformation du méta–modèle SPEM et création de la base de faits ;
2. ajout à la base des faits du modèle à vérifier ;
3. rédaction des règles de cohérence en programmation logique ;
4. lancement du moteur d'inférence pour le diagnostic.

Toutes ces étapes sont automatisées à l'exception de la troisième qui nécessite l'intervention de l'utilisateur. La création de la base de faits à partir du modèle et du méta–modèle se fait en fournissant les fichiers correspondants à un programme dédié et le diagnostic est obtenu en invoquant le moteur XSB [XSB07, SSW+06a, SSW+06b].

Dans le cas d'une vérification inter–processus, la structure générale est conservée mais l'étape $2'$ est introduite (figure 4.7(b)). En fait, dans ce cas de figure, les deux modèles sont introduits successivement dans la base de faits en précisant pour chaque élément s'il provient



(a) Vérification de propriétés sur un modèle



(b) Vérification de propriétés entre deux modèles

FIG. 4.7: *Principes de la vérification d'un modèle et de la vérification entre deux modèles.*

du modèle 1 ou du modèle 2. Comme dans le cas précédent, l'utilisateur va rédiger des règles de cohérence qui porteront, cette fois, sur les relations existant entre les modèles concernés.

4.2.3.4 Formalisation des règles de cohérence

La formalisation des règles de cohérence se fait en exprimant en XSB Prolog les règles en langage naturel identifiées jusqu'alors. D'un point de vue pratique, il est souvent plus intéressant de détecter la présence d'incohérences dans les modèles plutôt que de lister tous les éléments valides.

Si l'on prend comme exemples les règles de cohérence exprimées précédemment.

1. « *Il doit être possible de revenir sur chacune des étapes du processus.* »
2. « *Toutes les activités du modèle original doivent être présentes dans le modèle modifié.* »
3. « *Les CallBehavior ne peuvent être faits que pour les subWork du parentWork du WorkDefinition considéré.* »

On peut en déduire des règles d'incohérence à partir desquelles seront identifiés les éléments non–conformes. Dans notre cas, on obtient les règles ci–dessous.

1. « *Une activité A est non–vérifiable si elle peut atteindre une activité B alors que cette activité B ne peut l'atteindre.* »
2. « *Une activité est manquante si elle est présente dans le modèle original et n'est pas présente dans le modèle modifié.* »
3. « *Un travail réalise un CallBehavior incorrect si le travail appelé n'est pas un de ses sous–travaux.* »

Le tableau 4.1 donne les codes correspondants à ces deux règles. Dans ces exemples, tous les prédicats utilisés ont été générés automatiquement à l'exception des prédicats « inModel » et « reach ». Ils ont été déduits de l'analyse du méta–modèle lors de l'étape 1 (figure 4.7) de la vérification de propriétés et permettent de retrouver les éléments correspondants aux faits donnés par les modèles (étapes 2 et 2′). Les prédicats « inModel » et « reach »quant à eux sont utilisés pour simplifier l'écriture du code et correspondent à des prédicats complémentaires écrits sur la base des prédicats générés automatiquement. Dans ces exemples, « inModel » est vrai si l'élément est inclus dans le modèle et « reach » est vrai s'il existe un chemin entre les deux activités.

Règle en langage naturel	Règle codée
Une activité A est non–vérifiable si elle peut atteindre une activité B alors que cette activité B ne peut l'atteindre.	noVerifiable(IdAct1):– isActivity(IdAct1,_), isActivity(IdAct2,_), IdAct1\=IdAct2, reach(IdAct1,IdAct2), **not**(reach(IdAct2,IdAct1)).
Une activité est manquante si elle est présente dans le modèle original et n'est pas présente dans le modèle modifié.	missing(IdAct):– isActivity(IdAct,_), getName(IdMdOriginal,modeloriginal), getName(IdMdDerive,modelderive), inModel(IdAct,IdMdOriginal), **not**(inModel(IdAct,IdMdDerive)).
Un travail réalise un CallBehavior incorrect si le travail appelé n'est pas un de ses sous–travaux.	badCallBehavior(Work):– isWorkDefinition(Work,_), isParentWork(PrtWork,Work), isCallBehavior(Work,WorkCalled), **not**(isSubWork(WorkCalled,PrtWork)).

TAB. 4.1: *Exemples de formalisation des règles de cohérence.*

4.2.4 Synthèse des recommandations pour la validation des processus

Nous avons, dans cette section, proposé une méthode et un outil qui permettent la validation des modèles de processus. Nous avons vu que cette validation peut se faire sur un modèle comme sur deux modèles. Dans le premier cas, il s'agit de s'assurer que le modèle répond à des règles de cohérence. Dans le second que des relations de conformité existent entre les deux modèles.

Cependant, il faut préciser que notre outil n'est pas encore pleinement opérationnel. S'il est facile, pour l'utilisateur, de rédiger des règles de validation de la sémantique statique des modèles à partir des prédicats générés automatiquement, il lui est beaucoup plus difficile de rédiger des règles de validation de la sémantique dynamique. Il serait donc intéressant de poursuivre nos efforts de validation de modèles en associant à notre outil une bibliothèque de fonctions dédiées à la vérification dynamique.

4.3 Application à la conception d'un modèle réduit de voilier

Pour valider notre démarche, nous l'avons appliquée à un projet concret.

Un projet d'enseignement didactique a été retenu. L'application de la démarche sur un projet industriel sera une prochaine étape. Dans un premier temps, il s'est avéré nécessaire de consolider les concepts que nous proposions. Dans le cadre de ce test, les enjeux sont considérablement réduits par rapport à ceux rencontrés dans l'industrie. L'échelle de temps, plus courte, permet de disposer de retours plus fréquents et de résultats plus rapides. Enfin, ce type de projet laisse une grande liberté d'action.

Les objectifs de cette application sont d'illustrer comment la méthode que nous proposons peut être mise en place et quels sont ses apports. Nous ne chercherons pas à mettre en avant les transformations nécessaires à l'adaptation aux métiers et aux projets. Ces adaptations ainsi que leurs intégration dans les modèles ayant déjà été présentées dans ce mémoire. L'objectif ici est plutôt de montrer comment l'application des processus va se répercuter dans un projet.

Cette étude s'est déroulée dans le cadre d'un module d'ouverture de l'INSA de Toulouse intitulé « Conception d'un modèle réduit en matériaux composites ».

Un module d'ouverture est un enseignement ouvert aux étudiants de toutes spécialités qui souhaitent découvrir une matière sans pour autant se spécialiser dans celle–ci. Ces modules

ont pour finalité de consolider la culture générale des étudiants et de leur permettre d'acquérir des compétences tout en restant accessibles à toutes les filières.

Le module dans lequel nous intervenons propose à 24 étudiants, divisés en 4 groupes, de construire en 40 heures un modèle réduit de voilier télécommandé de classe 1 m. L'objectif pédagogique de cet enseignement est de donner une initiation par l'exemple de l'ingénierie système.

On attend des étudiants qu'ils livrent à la fin du module 4 bateaux complets constitués chacun d'une coque, d'appendice et de gréement. Des rapports techniques et des plans sont demandés tout au long du projet.

Pour arriver à ce résultat, les étudiants sont amenés à résoudre des problèmes d'analyse du besoin, d'analyse fonctionnelle, d'intégration multi-technologie, de collaborations dans un contexte de partenaires multiples, *etc.* Ils sont aidés par des introductions à la navigation et aux règles de course à la voile et conseillés au cours de leur projet.

La modélisation et le suivi des processus tels que définis dans cette thèse ne sont pas faits par les étudiants mais par les intervenants. Le temps de travail des étudiants ne leur permettrait pas de mener cette étude de bout en bout. De nombreuses tâches sont préparées ou faites par les intervenants pour permettre la réalisation des bateaux dans la limite de 40 h de travail par étudiant. Les techniques décrites dans cette thèse sont donc employées par les intervenants pour les aider dans leur encadrement des équipes d'étudiants.

4.3.1 Le voilier modèle 1m

Le modèle 1 m est la plus récente des classes de voilier télécommandé reconnue par la fédération internationale de voile (ISAF, International SAiling Federation). Les règles de course sont les mêmes que pour celles des voiliers dits « grandeurs » hormis que l'équipage se compose d'une personne sur la berge munie d'une télécommande. Les voiliers doivent respecter des règles de classe internationales [ISA07]. Les dimensions des mâts et des voiles sont définies strictement. En revanche, une certaine liberté est donnée en ce qui concerne la forme des carènes. Les principales règles de la classe sont :

- longueur : 1m maximum ;
- poids : 4kg minimum ;
- 3 gréements monotypes répondant à des contraintes précises de taille et d'accastillage ;
- 2 voies seulement pour la radiocommande ;
- matériaux exotiques (carbone, Kevlar) interdits pour la fabrication des coques.

Le principal intérêt de la classe 1m est de permettre la réalisation de bateaux compétitifs à moindre coût. Les performances de ce type de voilier sont souvent très proches, même pour des carènes très différentes. Les résultats en course demandent plus une pratique régulière et de l'habileté qu'un investissement matériel.

4.3.2 Modélisation et suivi du projet

4.3.2.1 Modélisation du projet

L'objectif pédagogique du module est de donner une illustration par l'exemple de l'IS. Pour cela, nous avons choisi de nous appuyer sur les recommandations de l'EIA-632. Pour illustrer ses concepts tout en intégrant les spécificités du projet, une spécialisation des processus est faite au travers des modèles de standard, de métier et de projet.

Modèle de standard

L'EIA-632 a été choisie comme référentiel pour la conduite de la conception et de la réalisation de modèles réduits. Le modèle de standard est donc identique à celui présenté à la section 2.4.

Modèle métier

La construction de voilier de classe 1 m est contrainte principalement par des limitations techniques fixées par la jauge : les dimensions, le poids ou les matériaux des composants du voilier sont fortement réglementés. En revanche, les constructeurs sont libres de concevoir et de réaliser leur modèle réduit comme ils l'entendent sous réserve que le bateau, une fois construit, respecte les contraintes techniques.

Des contraintes de procédures existent mais sont liées à la certification du bateau nécessaire à sa participation à une régate. Elles dépassent donc le cadre de ce module d'ouverture dont l'objectif consiste uniquement à réaliser les voiliers.

En l'absence de contraintes sur les processus même d'ingénierie, les adaptations nécessaires à notre métier (conception et réalisation d'un modèle 1m) sont très limitées. Le modèle métier est pratiquement similaire à celui du standard mis à part de légères modifications non significatives comme le passage de la jauge au lieu de la vérification du système principal.

Modèle de projet

On considère que les spécificités du projet par rapport au métier viennent du fait :

– que l'on cherche à réaliser quatre bateaux en parallèle ;
– que l'on se place dans le cadre d'un module d'ouverture.

Les contraintes d'enseignement font qu'il est nécessaire de prendre en compte des contraintes de temps (40h par étudiant), de ressources ou d'approvisionnement spécifiques. Ces contraintes entraînent des changements dans le séquencement de la construction du voilier. Pour la construction d'un seul bateau sans contraintes de temps, comme c'est généralement le cas pour les modélistes, les éléments du bateau sont réalisés un à un et les matériaux acquis au fur et à mesure sans contraintes d'approvisionnement. Dans le cadre d'un module d'ouverture, il est nécessaire d'acheter les matériaux à l'avance pour terminer la construction dans les temps. Le partage des ressources entre les différentes équipes oblige aussi à modifier l'ordre de réalisation. Les éléments sont construits en fonction de la disponibilité des ressources et non plus selon un ordre d'assemblage dicté par l'architecture du système.

Cependant, du point de vue du modèle de projet, ces contraintes sont vues comme des exigences et n'entraînent pas de changements significatifs dans le modèle de projet. En revanche, le déroulement du projet est affecté par des contraintes différentes. Si deux projets avec des contraintes distinctes ont des modèles identiques, leurs instances seront, par contre, différentes.

Comme pour le modèle de métier, les adaptations sont limitées et le modèle de projet reste proche du modèle initial.

4.3.2.2 Suivi du projet

Les adaptations des modèles de métier et de projet par rapport au modèle de standard sont limitées. Cependant, dans la mesure où l'on cherche à voir comment les processus d'IS s'appliquent, il est intéressant de travailler avec des processus proches des recommandations initiales.

L'application de ces processus à un projet a permis :

1. de suivre l'avancement du projet sur la base d'une formalisation de ses processus ;
2. de mettre en avant le fonctionnement des processus de l'EIA-632.

Suivi du projet

L'ingénierie du système est facilitée par l'utilisation d'un modèle référent. La formalisation des processus aide dans le suivi du projet et sa structuration.

Pour chaque phase du cycle de vie, un rapport résumant les solutions choisies et les vérifications effectuées est demandé aux étudiants. On voit, au travers de l'évolution de la représentation du système, apparaître les différentes couches des blocs de construction. Cette évolution est présentée à la figure 4.8. Elle montre l'état de la structure d'un voilier aux phases de pré–définition du système, définition du système, conception des sous–systèmes et conception détaillée.

Pour simplifier la représentation, on considère que les quatre bateaux sont identiques. Les schémas représentent une structure du système étendue puisqu'elle inclut les produits contributeurs en plus des produits dédiés aux fonctions opérationnelles du système. On peut y trouver des produits contributeurs de réalisation (moules, planches à gabarit, machine à laize), de mise à disposition (ber ou caisse à voile), de test (bac à jauge) et de formation (règles de course).

On constate que la structure définie lors de la définition du pré–système est identique à celle de la définition du système. Si la structure est identique, la connaissance du système s'est affinée entre les deux phases et passe du concept à un début de définition technique. Par la suite, on voit apparaître les éléments correspondants aux blocs de construction de la seconde couche puis de la troisième couche ou inférieure. La figure 4.9 montre la décomposition obtenue avant intégration, test et validation des produits finaux.

L'application de la démarche modélisée dans le modèle de projet permet de structurer la conception. En systématisant la vérification technique, la détection des incohérences de conception est facilitée. Du fait de la récursivité des processus, les spécifications des éléments concernés sont alors modifiés de manière à corriger les erreurs détectées puis re–vérifiées.

On trouve ci–dessous deux exemples de défauts détectés lors du projet :

1. l'exemple du passe–pont qui illustre la détection d'une erreur de spécification lors de la conception ;
2. l'exemple du puits de mât qui illustre la détection d'une erreur en phase d'intégration, test et évaluation.

Exemple du passe–pont Cet exemple illustre le cas dans lequel la définition d'un sous–système va amener à revoir non seulement les spécifications du système dont il dépend mais

(a) PBS du voilier après définition du pré–système

(b) PBS du voilier après définition du système

(c) PBS du voilier après conception des sous–systèmes

(d) PBS du voilier après conception détaillée

FIG. 4.8: *Évolution du PBS du voilier au cours du cycle de vie d'ingénierie. Les PBS représentés sont étendus et incluent les composants du système aussi bien que les produits contributeurs. Dans les deux cas, on distingue les éléments sur lesquels seront ré–appliqués les processus d'ingénierie des éléments unitaires implémentés, existants ou acquis.*

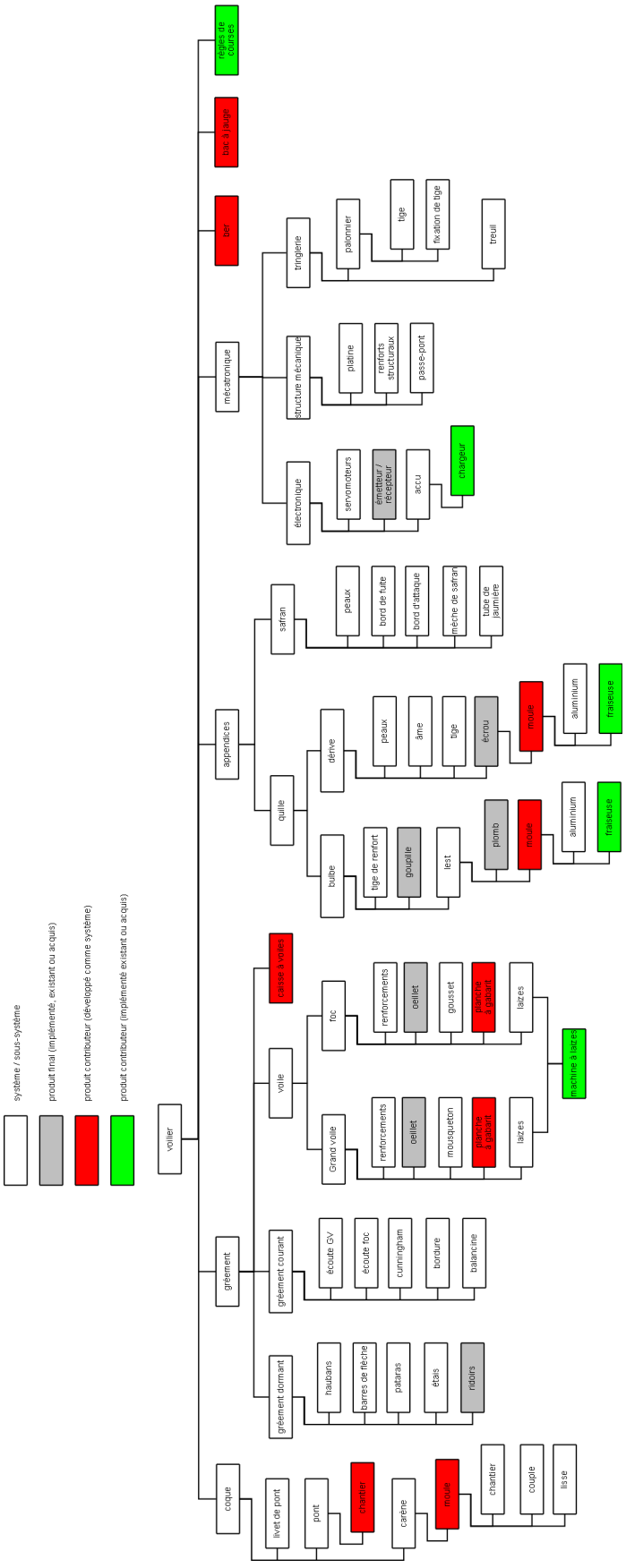


FIG. 4.9: *PBS du voilier après conception détaillée. Les produits finaux courants (bois, époxy ou colle) et les produits contributeurs courants (petit outillage) ne sont pas représentés*

aussi les spécifications des systèmes environnants.

La définition du système conduit à décomposer le voilier en coque, gréement, appendice et mécatronique (figure 4.8(b)). A ce stade, les spécifications d'interfaces entre mécatronique et coque se limitent à la fixation mécanique des éléments à la coque. Lors de la conception du sous–système mécatronique, la tringlerie, qui a pour fonction de transmettre les mouvements des servomoteurs au gréement, est introduite (figure 4.8(c)).

L'évaluation technique de la tringlerie va conduire à deux modifications majeures des spécifications :

- D'une part, les spécifications d'interface entre mécatronique et tringlerie peuvent maintenant être rédigées de manière technique.
- D'autre part, les concepteurs sont amenés à modifier la spécification du pont de manière à permettre le passage des câbles de la tringlerie. L'application du processus d'évaluation technique permet de reporter ce besoin sur le pont en modifiant ses spécifications. La tringlerie va donc avoir un impact sur un élément de la coque alors qu'ils appartiennent à des blocs de construction différents.

Exemple du puits de mât L'exemple du puits de mât illustre le cas de la modification d'un composant lié à un défaut détecté lors de la phase d'intégration.

Il peut arriver que le moule de puits de mât réalisé par les étudiants soit trop court. Dans ce cas, le pont, qui vient effleurer le puits de mât, va être abaissé par rapport à ce qui était prévu. Pour éviter un écart trop important entre le pont et la baume, la position de la baume doit être abaissée. La hauteur de voile étant fixée par la jauge, la seule solution pour compenser la différence de hauteur est de scier le mât.

En résumé, le processus d'intégration, test et évaluation permet de modifier les spécifications d'un sous–système pour rattraper les défaillances d'un autre sous–système.

Ce fonctionnement n'est bien entendu pas systématique et la plupart du temps l'élément défaillant est remplacé. Cependant, dans certains cas, il est plus intéressant de modifier des spécifications que de modifier ou reconstruire un élément.

Application de l'EIA-632

L'EIA-632 s'oriente principalement autour de la conception du système et de sa vérification. De ces points de vue, la démarche est très structurée : découpage de la conception en

quatre phases, décomposition du système en blocs de construction, test et validation systématique des éléments acquis ou fabriqués *etc.* En revanche, le processus décrit ne permet pas de prendre facilement en compte les contraintes liées à l'acquisition et à l'implémentation des composants.

En pratique, les étudiants ont dû anticiper sur la construction de certains éléments sans attendre la spécification complète du voilier. Les limites de temps et les contraintes liées à la répartition des ressources entre les groupes d'étudiants ont rendu nécessaire des déviations par rapport au comportement strictement descendant/ascendant.

Ces déviations ont de grandes chances de se retrouver dans les projets industriels pour lesquels les contraintes de temps et de ressources sont très fortes. C'est toute la différence entre théorie et pratique. S'il est inévitable de dévier par rapport aux recommandations, il est en revanche nécessaire de limiter ces déviations pour profiter au maximum du rôle structurant des processus d'IS. L'avantage de la formalisation des processus du projet est qu'elle permet de connaître avec précision quelles ont été les déviations par rapport aux recommandations.

Ce comportement correspond bien à l'analyse qui a été faite de l'EIA-632 et a déjà été identifiée à la section 2.4.4. En revanche, l'application des processus à un projet a permis d'identifier un défaut dans le concept de système proposé par l'EIA-632.

On peut remarquer sur la structure de système de la figure 4.9 que deux types de produits contributeurs sont employés :

produit contributeur classique : attaché à un élément du système ;

produit contributeur à usage multiple : attaché à plusieurs éléments du système.

L'association simple d'un produit contributeur à un élément du système correspond tout à fait à la structure de système décrite par l'EIA-632. En revanche, l'association de plusieurs éléments à un produit contributeur n'est pas couvert par les recommandations. Dans notre exemple, la machine à laize doit être développée de manière à pourvoir produire aussi bien des laizes de grande voile que des laizes de foc. En limitant les liens d'un système à ses sous–systèmes de niveau 1 et en forçant les sous–systèmes à être liés à un et un seul système, le processus d'ingénierie ne permet pas de partager un sous–système ou un produit contributeur entre plusieurs systèmes.

La conception d'un produit standard, contributeur ou non, utilisable tel quel sur plusieurs systèmes est pourtant une activité essentielle communément rencontrée en industrie. Pour profiter d'une mise en commun des efforts de conception et limiter la redéfinition de composants,

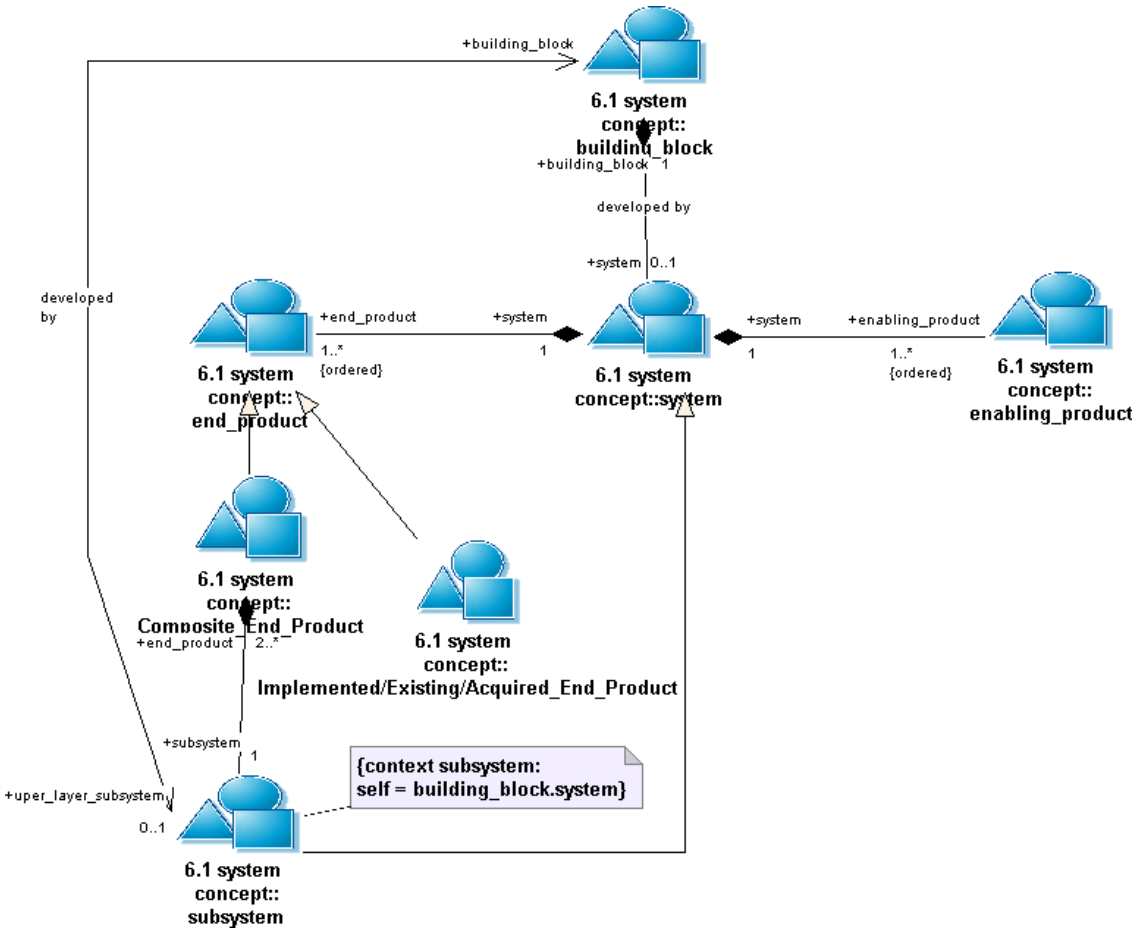


FIG. 4.10: *Concept de structure de système pour systèmes à usages multiples. La multiplicité des relations entre produit final et sous–systèmes passe à* $2..*$, $1..*$ *(originalement à* $2..*$, 1 *dans la figure 2.8) pour permettre l'ingénierie de produits destinés à plusieurs produits finaux.*

il faut permettre l'association de plusieurs produits finaux à un sous–système. Formellement, notre recommandation revient à remplacer la multiplicité « $2..*$, 1 » entre produit final et sous–système et entre produit final et produit contributeur par une multiplicité « $2..*, 1..*$ » (figures 4.10 et 4.11).

4.3.3 Synthèse pour l'application des processus à la conception d'un modèle réduit

Cette section a donné un aperçu de l'application de notre démarche. On a pu constater que l'utilisation d'un modèle formalisé peut aider au suivi du projet. En structurant la démarche de conception et de réalisation, il guide les acteurs du projet en favorisant la détection précoce d'incohérences de conception.

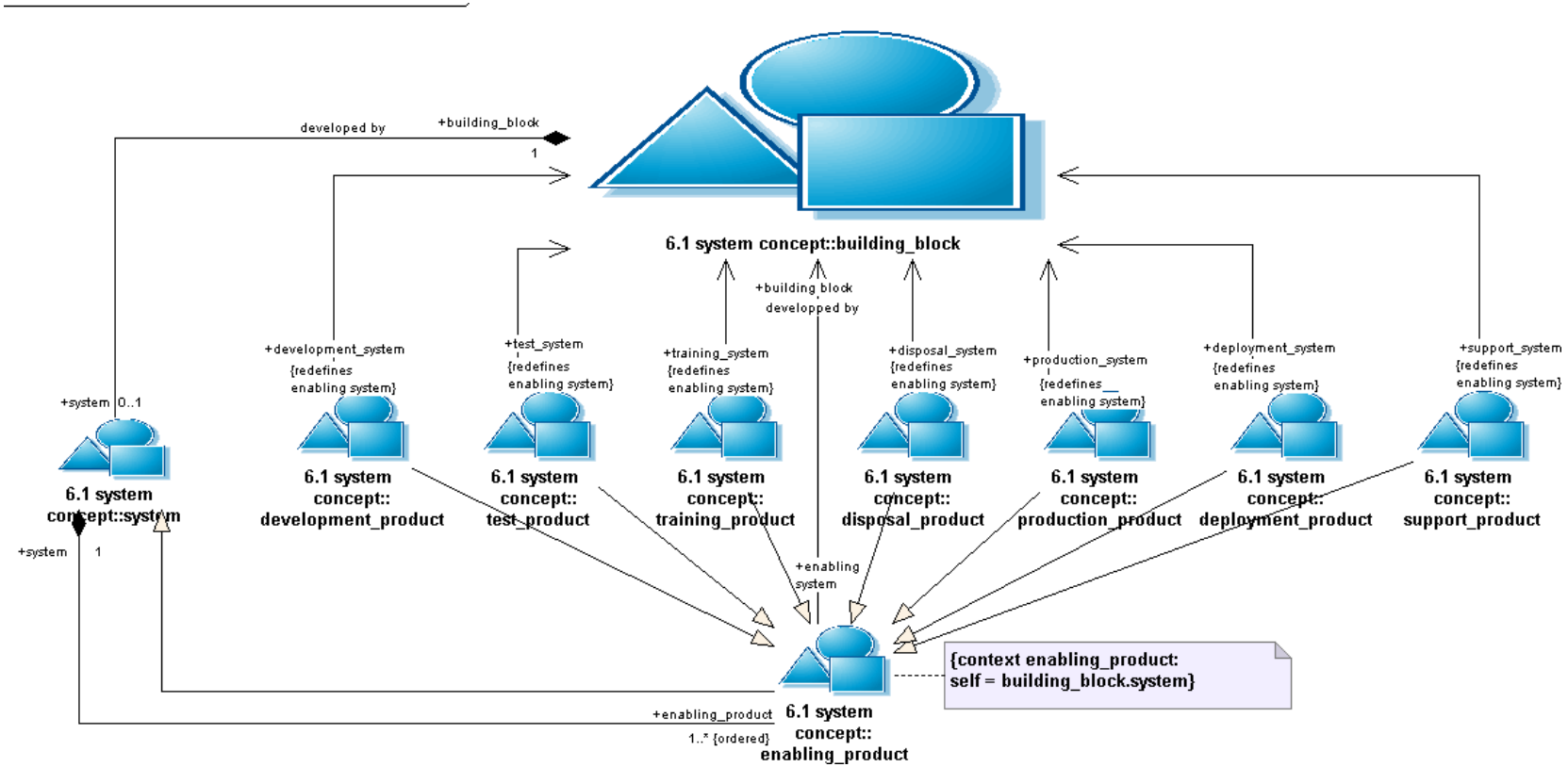


FIG. 4.11: *Concept de structure de système pour les produits contributeurs à usages multiples. La multiplicité des relations entre produit final et produits contributeurs passe à* $2..*$, $1..*$ *(originalement à* $2..*$, 1 *dans la figure 2.8) pour permettre l'ingénierie de produits destinés à plusieurs produits finaux.*

La mise en œuvre de la démarche a aussi été l'occasion d'expérimenter l'application des processus de l'EIA-632. On a pu constater les avantages fournis par cette méthodologie tout en identifiant deux limites majeures :

- les déviations par rapport au processus descendant et ascendant pourtant nécessaires au respect des contraintes du projet et
- une structure de système qui ne permet pas la mise en commun de ressources.

4.4 Extension : identification précoce des besoins pour une conception système orientée modèles

La conception d'un système complexe passe par les modélisations successives du système, de son environnement, du projet de conception... La difficulté consiste à faire évoluer ces modèles tout en conservant une cohérence jusqu'à satisfaction des besoins du client. Jusqu'à présent chaque domaine de l'entreprise travaillait avec des langages propres sans liens formels avec ceux utilisés par les autres.

A l'origine réservés au logiciel, on commence à voir apparaître des environnements de conception orientés modèles pour les systèmes. Une conception système qui s'articule entièrement autour de modèles permettrait une maîtrise de bout en bout de la conception. La transposition des concepts de l'IDM aux systèmes n'est cependant pas immédiate et de nombreuses questions tant techniques que méthodologiques restent à résoudre.

Nous montrerons que la formalisation des processus de projet peut permettre une identification précoce des langages qui seront manipulés. Ce travail d'identification servira à guider les efforts de développement en limitant au strict nécessaire les inclusions de langages et leurs transformations de modèles dans l'environnement.

4.4.1 Contexte

Les techniques d'ingénierie de modèles, traditionnellement liées au seul domaine informatique, sont maintenant étendues au domaine des systèmes. Les industriels ont compris que le développement de systèmes, qui nécessite l'utilisation de langages de modélisation multiples doit être encadré. Il est nécessaire de lier les modèles de manière à conserver leur cohérence et permettre leur validation. Des processus de développement doivent alors reposer sur des opérations de transformation de modèles guidées par les processus de conception.

Le projet TOPCASED [FG05, CLCF+05, Gau06] (Toolkit in OPensource for Critical Application & SystEms Development) est le projet le plus actif actuellement. Il regroupe de nombreux industriels dont Airbus, Astrium, le CNES, Siemens VDO, Thales ou Freescale avec des PME et des laboratoires de la région toulousaine. Son objectif est le développement d'un ensemble d'outils open source de génie système et logiciel. Ce projet doit permettre de créer, de vérifier et de transformer l'ensemble des modèles intervenant dans le développement d'un système.

Les langages envisagés s'adressent aux processus de développement du système, à son architecture ou à son fonctionnement. On peut citer par exemple :

- Scade/LUSTRE ;
- MatLab/Simulink ;
- ESTEREL ;
- SDL ;
- l'analyse structurée ;
- des architectures temps réel (ARINC653, bare hardware) ;
- machines de Mealy ;
- HOOD ;
- UML ;
- SysML ;
- AADL ;
- EAST-ADL ;
- langages expérimentaux ;
- autres.

L'utilisation de techniques de méta–modélisation est exploitée pour la gestion des modèles des langages généralistes (General Purpose Languages) comme des langages dédiés (Domain Specific Languages ou DSL).

A terme, ce projet doit donner lieu à un environnement de développement complet sous Eclipse.

Pour notre part, nous nous contenterons de mettre en avant le rôle des processus de projet sur les modèles manipulés. **Nous montrerons notamment qu'un modèle des processus de projet peut aider à l'identification des modèles requis pour le développement d'un système ainsi qu'à l'identification des transformations entre ces modèles.**

4.4.2 Problématique

L'environnement de développement TOPCASED doit permettre la manipulation de modèles associés à des éléments du système tout au long des étapes de son développement.

L'utilisation pratique d'un tel environnement demande :

- de connaître les éléments constitutifs du système ;

– de leur associer un ou plusieurs langages de description selon les phases de développement considérées ;
– de connaître les transformations de modèles nécessaires à la conception du système.

En supposant qu'elles soient disponibles, ces informations peuvent être reportées sur un tableau équivalent au tableau 4.2.

Disposer de ces informations permet de cibler les besoins de manipulations et de transformations de modèles. Elles permettent de :
– limiter le nombre de langages intégrés à TOPCASED à ceux effectivement employés dans un projet ;
– identifier les besoins de transformations entre langages dans ce même projet.

Il ne serait pas possible de créer toutes les transformations ou même d'intégrer tous les langages existants. D'autre part, il est inutile de proposer des opérations de transformations de modèles si celles–ci n'ont aucun sens. Par exemple, dans le cas du développement d'un composant électronique un passage de SysML/UML vers VHDL puis vers FPGA est cohérent. En revanche, il est difficile d'envisager un passage direct de SysML/UML en FPGA.

Les langages et les transformations traités par l'environnement de développement doivent donc être étudiés soigneusement. En effet, chaque intégration de langage ou de transformation demande un effort de développement. En outre, l'absence d'un langage ou d'une transformation dans l'environnement remet en cause la cohérence du développement. La chaine de modèles serait brisée.

Identifier les langages qui seront manipulés et les transformations de modèles nécessaires pose les difficultés suivantes :
– Comment identifier *a priori* les composants du système alors qu'ils sont déterminés au cours du projet ?
– Comment en déduire les langages adaptés ?
– Quelles sont les opérations de transformation nécessaires ?

Par la suite, nous verrons comment la formalisation des processus du projet peut aider à identifier de manière précoce les langages et les transformations nécessaires.

4.4.3 Identification des langages et des transformations de modèles

Jusqu'à présent nous nous sommes focalisés sur la mise en œuvre des processus d'ingénierie système. Nous avons décrit comment l'utilisation et la déclinaison de modèles permettent une application cohérente des processus d'ingénierie système tout en permettant de les adapter à chaque partenaire.

Nous verrons maintenant que le processus de modélisation proposé peut aussi être utilisé pour faciliter la gestion des langages de modélisation associés au système.

La problématique revient à estimer au mieux et de manière préventive les langages qui seront utilisés au cours du projet.

On s'intéressera, dans un premier temps, au modèle d'association des langages aux processus. Nous verrons ensuite comment identifier de manière précoce les langages utilisés sur la base du modèle décrit.

4.4.3.1 Intégration des concepts de langages et de modèles

Au cours du développement d'un système, plusieurs langages peuvent être utilisés pour décrire les modèles du système. Ces modèles sont utilisés pour décrire le système et ses exigences vis–à–vis de ses produits contributeurs ou sous–systèmes. Chaque modèle décrit un point de vue particulier du système.

A chacun de ces stades, le système est représenté par plusieurs modèles qui vont se compléter pour couvrir entièrement le besoin de représentation du système. [Gui07] identifie les axes suivants dans le recouvrement des modèles :

multi–domaines : les modèles sont différents selon le domaine considéré (électronique, mécanique, achats...) ;

multi–échelles : des langages sont adaptés à la modélisation d'éléments dans une échelle donnée de temps, de dimension, *etc.* ;

descriptif/prédictifs : modèles dédiés à la communication ou à la simulation ;

multi–abstractions : l'expression de points de vues spécifiques sur le système ou mise en relief de certaines propriétés demande des langages spécifiques.

Un même système est donc représenté par plusieurs modèles utilisés en fonction de la phase du cycle de vie et des besoins de modélisation en un instant donné.

Ils formalisent le système à différents stades de sa conception. En se basant sur l'EIA-632, le système passe par une représentation de ses exigences, à sa solution logique, à sa solution physique et enfin à sa solution de conception.

Le modèle de la figure 4.12 représente l'intégration de ces notions dans le modèle de la structure de système. Il reprend les concepts de représentation de la solution logique, de représentation de la solution physique et de solution de conception et leur associe des modèles spécifiques.

En plus des modèles utilisés pour la représentation du système, le dernier type de modèle concerne les exigences. Dans le cas d'exigences complexes ou formalisées, les exigences peuvent être associées à un ou plusieurs modèles. D'autre part, comme on le verra par la suite, ces modèles d'exigences peuvent servir de liens avec les spécifications du bloc de construction de niveau supérieur.

Les modèles associés aux descriptions du système peuvent évoluer selon la phase du cycle de vie. Par exemple, on peut utiliser un langage pour décrire la solution logique en phase de pré–conception, puis utiliser un second langage pour décrire plus en détail cette même solution dans la phase de conception. L'ensemble des modèles associé à une description évolue donc avec le cycle de vie du système.

L'environnement dans lequel s'effectue le développement doit permettre la manipulation des langages identifiés mais aussi la transformation des langages en accord avec l'ordre donné par les phases de développement.

Un modèle tel que celui de la figure 4.12 permet :

– d'identifier les langages utilisés lors du développement ;
– d'identifier les liens inter–modèles nécessaires.

La liste des langages est déduite des modèles utilisés lors du développement. Les transformations entre langages sont déduites par l'étude des relations entre modèles. Ces relations sont dictées par l'enchaînement des activités de conception représentées dans le modèle de processus. D'une part, chaque modèle est lié à une représentation du système ou à une exigence (figure 4.12). D'autre part, les relations entre représentation système, représentation logique, représentation de solution physique, représentation de solution de conception et exigences sont connues (figure 2.9). Les besoins de transformation de modèle peuvent être obtenus par l'étude des relations entre ces éléments.

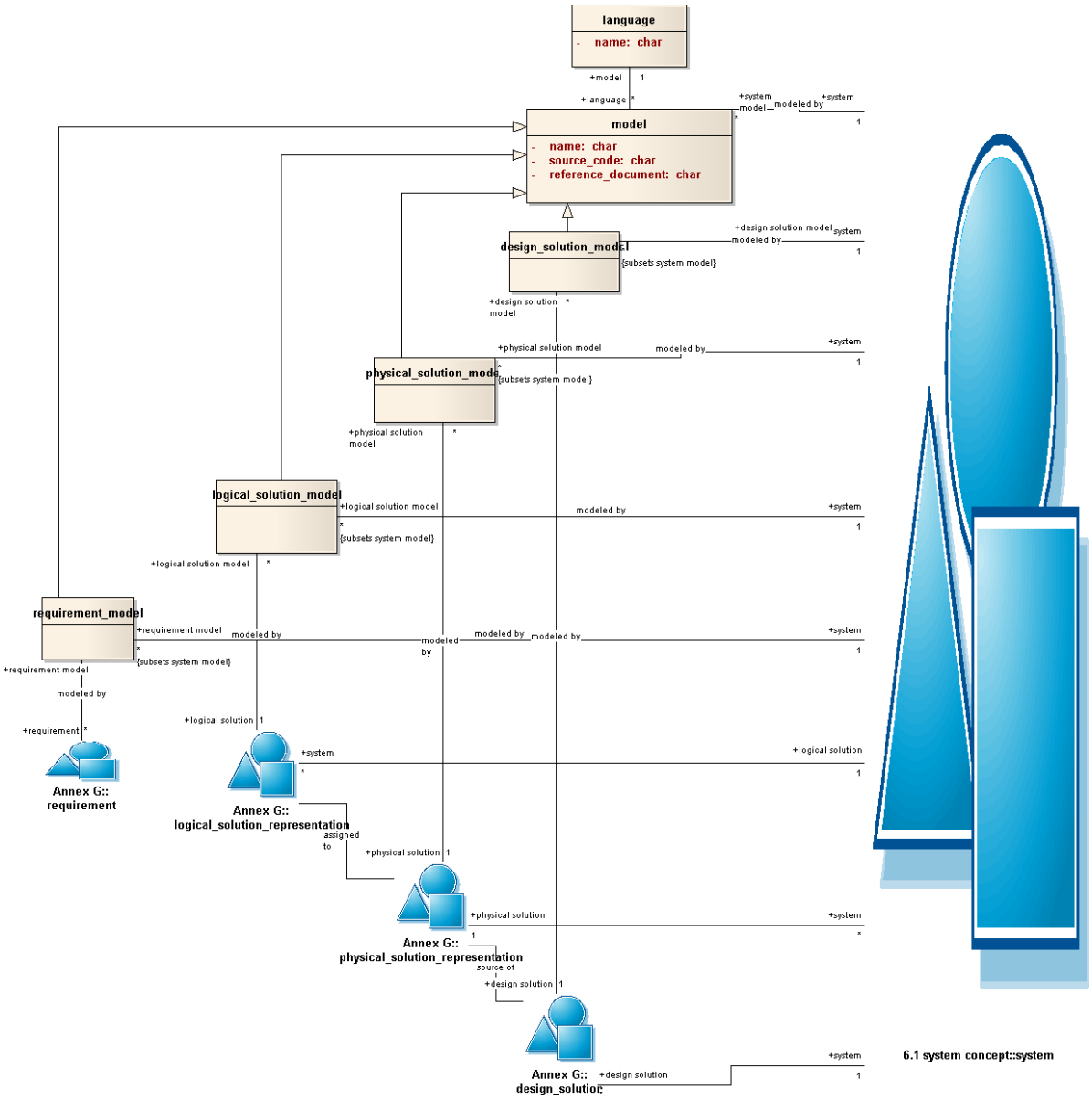


FIG. 4.12: *Langages et système. Au cours de son développement, un système passe par plusieurs représentations : solution logique, solution physique et solution de conception. Un ou plusieurs modèles spécifiques sont associés à chacune de ces représentations ainsi qu'aux exigences du système. Ces modèles sont exprimés dans des langages dédiés.*

On sait, par exemple, que la solution physique est source de la solution de conception. L'environnement de développement doit donc permettre la transformation des langages de description de la solution physique en langages de description de la solution de conception.

Les propriétés du développement en couches sont conservées. Dans le cas de systèmes complexes composés de plusieurs blocs de construction, il devient possible d'utiliser des modèles comme éléments de spécification. Une spécification est rédigée au niveau du bloc de construction principal puis associée à un modèle comportemental du sous–système vis–à–vis du système principal. Les spécifications étant considérées comme des exigences du client dans le bloc de construction de niveau inférieur, le sous–système sera développé sur la base de cette description. En utilisant ce mécanisme, notre modèle permet de représenter le passage de spécifications à un sous–traitant ou une équipe de développement sous forme de modèle du sous–système à développer. L'analyse des relations entre modèles et des besoins en transformation peut alors se propager dans toute l'architecture du système (figure 4.13).

Les langages utilisés aux différents stades du développement peuvent être identiques. On peut utiliser, par exemple, le ou les mêmes langages pour la description de la solution physique et pour la solution de conception mais à des niveaux de détails différents.

Dans le cas d'une spécification par modèle, les modèles de spécification (bloc de construction supérieur) et d'exigence client (bloc de construction inférieur) représentent le même objet. Il s'agit donc du même langage. En revanche, il est possible que l'équipe de développement du sous–système utilise d'autres formalismes lors de son traitement des exigences (exigences client dans un formalisme et exigences techniques dans un autre).

Le modèle de spécification d'un sous–système peut être un sous–ensemble du modèle de solution de conception.

Jusqu'à présent, les processus de transformation de modèles n'avaient fait l'objet que d'études limitées à un métier particulier (électronique [Gui07], systèmes embarqués temps réel [SV07], *etc.*) et pour des cycles de développement figés. La formalisation des processus d'IS, couplée à la formalisation de la structure du système permet de lier fortement modèles et processus de développement.

Plus que cela, dans la mesure où notre proposition est d'adapter les représentations aux besoins métier et projet, le choix des processus de développement et des langages est laissé libre. C'est un premier pas vers une réflexion globale sur les relations et interconnexions entre systèmes et processus. De plus, les adaptations étant locales aux blocs de construction,

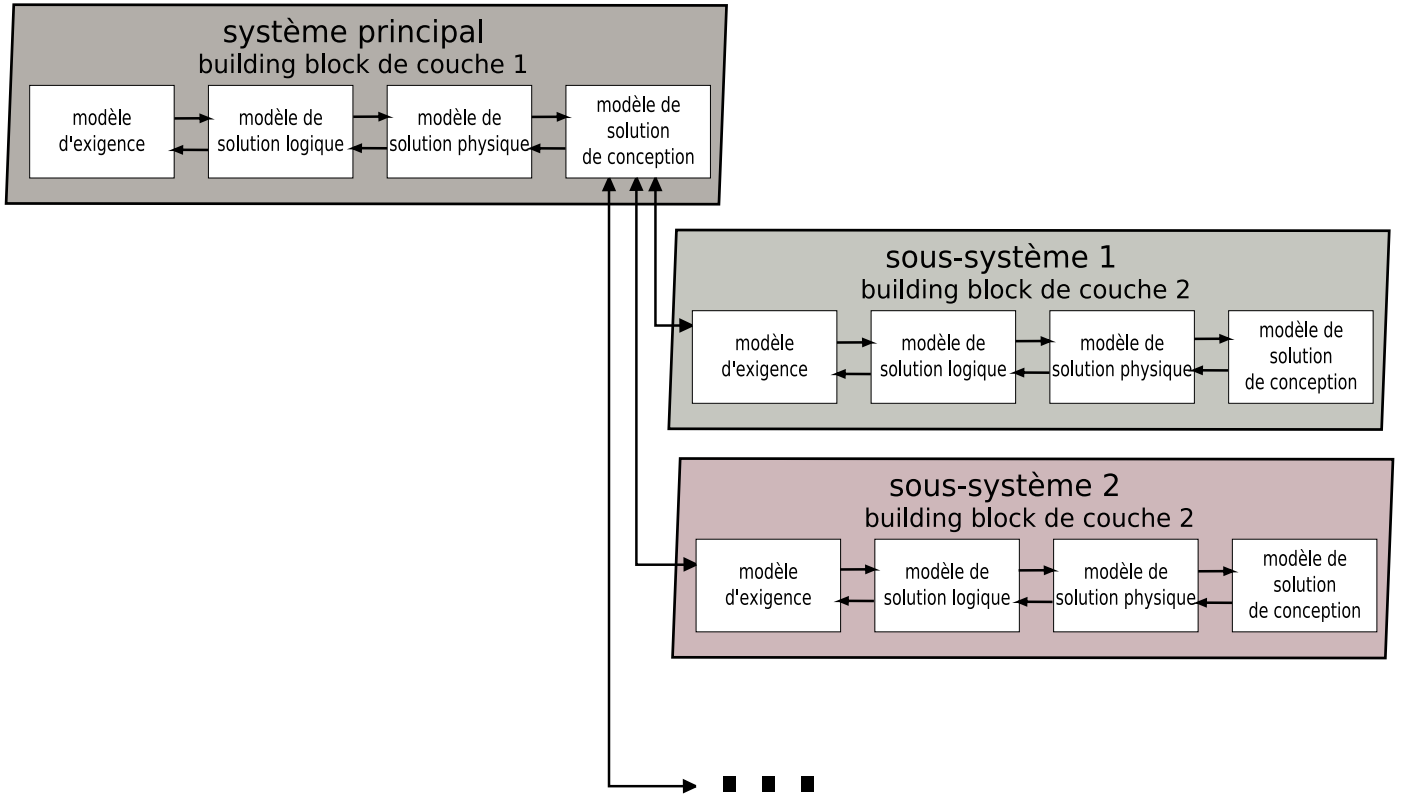


FIG. 4.13: *Modèles associés à un système dans un bloc de construction. Un système doit être modélisé successivement par ses exigences, dans une solution logique, une solution physique puis une solution de conception. L'environnement de développement du projet doit permettre la transformation entre langages d'expression d'une représentation à l'autre.*

Les spécifications d'un système étant considérées comme des exigences client dans les blocs de construction de rang inférieur, les modèles de spécification deviennent modèles d'exigences dans la couche inférieure. Des transformations entre blocs de construction de couches différentes sont identifiées par ce lien.

Pour un déroulement de projet optimal, des opérations de transformations de modèles doivent être possibles pour chacune des flèches représentées.

l'identification des langages est adaptée selon les systèmes ou les sous–systèmes considérés (figure 4.14). A terme, on devrait pouvoir assurer d'une cohérence globale des processus et des systèmes.

4.4.3.2 Identification précoce des langages et des opérations de transformation

L'identification des langages utilisés et des transformations peut se faire à partir d'une modélisation du projet. Malheureusement disposer d'un modèle complet du projet lors des phases amont n'est pas toujours possible. Nous verrons dans cette section comment modéliser au plus tôt les informations des projets de manière à anticiper au maximum sur le développement des opérations de transformations de modèles.

Notre hypothèse ici est qu'il faut adapter la procédure d'identification en fonction du type de conception rencontré par le projet. D'après [Var05] , on peut considérer trois types de conception qui diffèrent selon les sources de connaissance disponibles (tableau 4.3). On recense ainsi la conception routinière, la conception innovante et la conception créative.

La conception routinière se définit par la disponibilité des trois sources de connaissances à savoir l'objectif, le domaine et la démarche. Nous pouvons parler de conception routinière dès lors que les concepteurs ont une certaine habitude et expérience sur la démarche de conception d'une même famille d'objets. Ce type de conception concerne généralement un objet de complément de gamme, de remplacement ou des améliorations incrémentales.

Dans la conception innovante, seules deux sources de connaissances sont disponibles : l'objectif et le domaine. C'est-à-dire que l'expression du besoin et les technologies à employer sont le plus souvent identifiées mais les stratégies de conception restent à définir. Tel est le cas pour la mise sur le marché d'un objet correspondant à un besoin exprimé par les clients, mais non encore satisfait.

La conception créative est celle qui dispose du moins de connaissances. L'objet à concevoir est en général nouveau sur le marché, ce qui implique que seul l'objectif est défini. Les technologies à mettre en œuvre sont à identifier et à s'approprier. Les concepteurs n'ayant jamais eu à se confronter à ce type de problème, n'ont aucune expérience sur la démarche de conception à déployer.

L'identification des langages et des transformations de langages nécessaires au projet dépend en grande partie de la connaissance préalable que l'on a de celui–ci.

Éléments du système	Langages			
	Langage 1	**Langage 2**	**Langage 3**	**Langage 4**
Élément 1	✘	✘ ➠ L3		
Élément 2			✘	
Élément 3	✘ ➠ L2			
Élément 4		✘ ➠ L4		
Élément 5		✘		
Élément 6				✘

TAB. 4.2: *Tableau illustratif du fonctionnement d'un atelier de manipulation de modèles. Chaque élément d'un système peut être représenté dans un ou plusieurs langages. Le symbole ✘ montre que l'élément ou un point de vue sur cet élément peut être exprimé par le langage correspondant. Dans le cas où des opérations de transformations de langages sont nécessaires, le symbole ➠ X est utilisé. Il indique qu'un modèle dans le langage correspondant doit pouvoir être transformé en un modèle du langage X. Dans cet exemple, les éléments 1 et 3 vont être associés à des modèles exprimés en langage 1. Des transformations de modèles du langage 2 vers les langages 3 et 4 seront utilisées.*

Type de conception	Connaissance		
	de l'objectif	**du domaine**	**de la démarche**
Routinière	oui	oui	oui
Innovante	oui	oui	non
Créative	oui	non	non

TAB. 4.3: *Connaissances disponibles suivant le type de conception.*

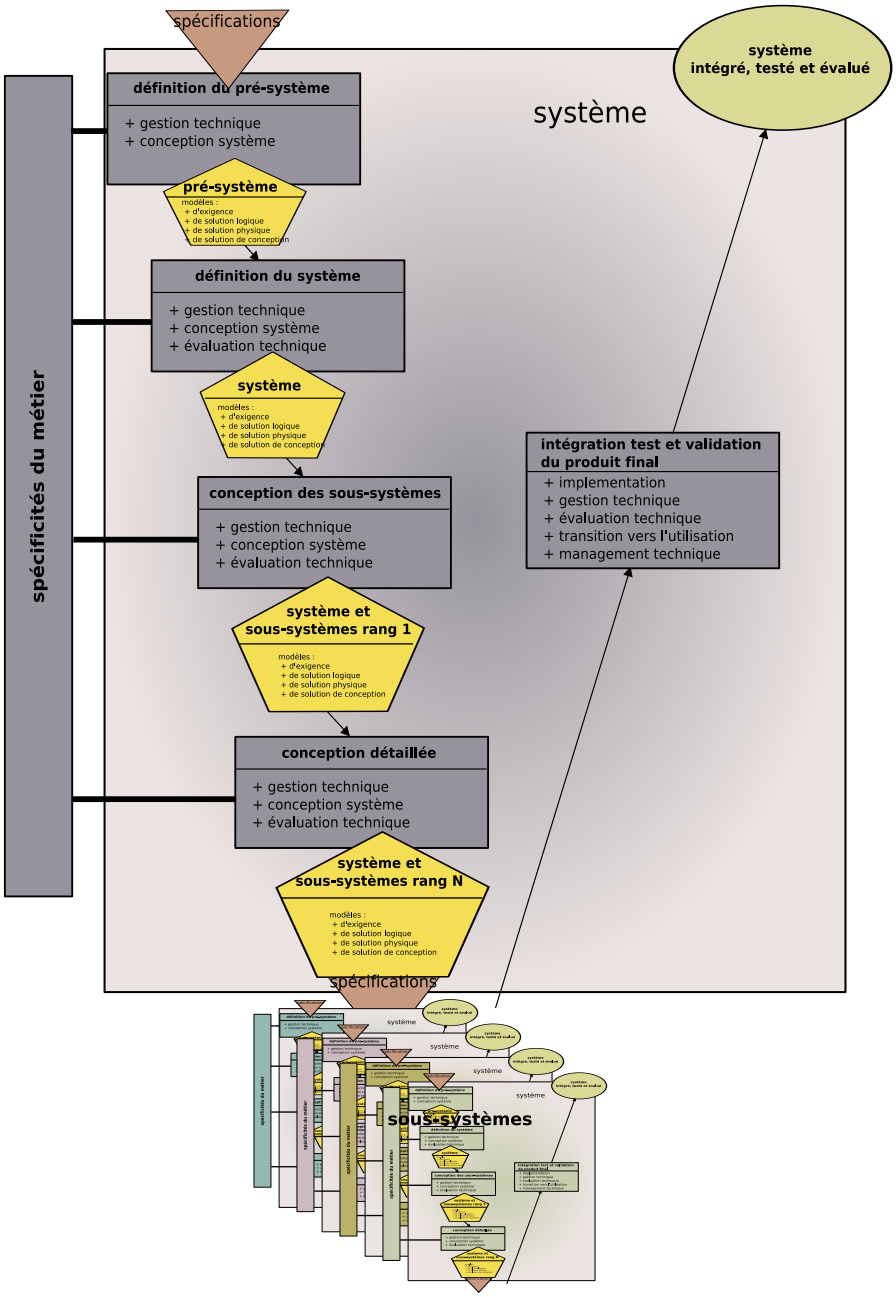


FIG. 4.14: *Langages et cycle de vie du système. L'identification des langages et des opérations de transformations de langages se déduit de l'étude du cycle de vie du système. L'étude des modèles de représentation du système permet de dresser la liste des langages à utiliser. Les interactions entre langages sont déduites de l'enchainement des phases du cycle de vie et du séquencement des processus dans les phases du cycle de vie. Cette identification couvre le bloc de construction du plus haut niveau jusqu'aux blocs de construction les plus bas.*

Dans le cadre de notre problématique, chaque type de conception sera traité différemment.

Identification des langages en conception routinière

Dans le cas de la conception routinière, on a une bonne connaissance préalable de la structure du système et des technologies employées. A partir de projets existants, on peut déduire des patrons de processus. Ces patrons permettent de pré–fixer les grandes lignes du projet dont la décomposition en blocs de construction et les langages associés aux éléments du système qui seront récurrents.

Par exemple, une voiture sera systématiquement composée de quatre roues, d'un moteur. . . ; le moteur sera représenté par un modèle thermique et un modèle mécanique. . .

L'identification des langages et des transformations nécessaires se fait, dans ce cas, sur la base de l'expérience acquise. On dispose en quelque sorte d'un modèle de projet standard par rapport auquel le modèle de projet effectif ne connaîtra que peu de variations.

Identification des langages en conception innovante

Être en conception innovante suppose une connaissance *a priori* de la solution logique répondant aux attentes du client ainsi que des technologies employées pour répondre à ces attentes. Si la connaissance du domaine est assez précise, elle peut suffire à identifier de manière précoce les langages employés.

On peut identifier des tâches de développement et leur associer des langages sans pour autant connaître dans le détail le système à concevoir. Le langage employé peut être directement déduit de la technologie.

Par exemple, VHDL, Verilog ou encore SystemC dans le cas de l'électronique numérique et OrCAD/PSpice dans le cas de l'électronique analogique.

Si de grosses variations peuvent exister d'un système à l'autre, le fait de travailler dans un domaine connu permet d'employer des méthodes de conception relativement stables.

Cependant en conception innovante, la connaissance de l'architecture du système est limitée. S'il est possible de rédiger un patron pour les premières couches de bloc de construction, on ne peut pousser la démarche jusqu'aux blocs de construction de rang inférieur. On ne peut donc plus utiliser un pattern unique prédéterminé en début de projet.

Notre recommandation est d'utiliser des patrons au niveau des blocs de construction en employant la connaissance *a priori* des processus du bloc de construction (projets antérieurs, informations des blocs de construction de couche supérieure, *etc.*). Une fois la conception de

ce bloc de construction terminée, les patrons des blocs de construction de couches inférieures sont déterminés.

Le concepteur ne dispose plus d'une connaissance exhaustive en début de projet mais d'une connaissance limitée à un bloc de construction et disponible juste avant le lancement de ses processus.

Identification des langages en conception créative

En conception créative, on ne dispose plus de connaissances préalables sur le système. On est contraint d'attendre les choix de conception pour en déduire les langages nécessaires. Il n'est plus possible d'anticiper sur les processus du projet. L'utilisation de patrons n'est plus envisageable.

Les fonctions de manipulation de langages de l'environnement de développement étant nécessaires au développement du système, elles peuvent être considérées comme des produits contributeurs. A ce titre, leur intégration fera partie intégrante du projet et suivra les mêmes processus que tout autre produit contributeur.

Résumé des méthodes d'identification selon le type de conception

La connaissance préalable des langages de conception est fortement dépendante du type de projet considéré. Notre proposition est d'adapter la technique d'identification des langages à la connaissance disponible. Selon les cas, on disposera d'une connaissance exhaustive et précoce des langages (conception routinière), d'une connaissance parcellaire qui sera complétée au cours du projet (conception innovante) ou, dans le pire des cas (conception créative), de pas de connaissances. Dans ce cas cependant, les besoins sont traités comme des produits contributeurs, et, sont, à ce titre, traités en cours de projet.

La figure 4.15 résume le type de traitement selon la conception considérée. En conception routinière, les langages sont déduits d'un patron couvrant l'ensemble du projet. En conception innovante, ils sont déduits successivement de patterns limités à un bloc de construction. Enfin, en conception créative, le manque de visibilité sur le projet rend impossible l'utilisation de patrons.

FIG. 4.15: *Patrons utilisé selon le type de conception. En conception routinière le patron couvre l'ensemble du projet.*
En conception innovante un patron couvre uniquement les processus associés à un bloc de construction. Une fois la conception de bloc de construction terminée, de nouveau patrons sont utilisés pour les blocs de construction de couche inférieure.
En conception créative, la faible connaissance du projet empêche l'emploi de patrons.

4.4.4 Synthèse pour l'identification précoce des besoins pour la conception système guidée par les modèles

La formalisation des processus de projet prend une place de plus en plus importante dans leur gestion. A l'origine utilisée pour valider et contrôler les processus de projet, la formalisation permet aussi d'identifier des comportements récurrents dans les projets. L'expérience acquise, à son tour formalisée, est alors capitalisée en favorisant la mise en place des futurs projets. **Dans ce document, nous avons mis en avant la possibilité d'identification précoce des langages de développement et des transformations. Nous avons montré que les informations issues du comportement de projets antérieurs pouvaient être à la source de cette identification.**

Dans le cadre des réflexions actuelles portant sur les environnements de développement par modèles (projets TOPCASED, DOMINO), la méthode que nous avons décrite peut être employée pour guider les efforts de conception. En se basant sur des projets de référence, les langages et les transformations de base devant être intégrés dans les environnements peuvent être déterminés. A plus long terme, une telle méthode peut servir à réduire la complexité de l'environnement. En proposant des profils limitant les langages et les transformations à ceux et celles strictement nécessaires, l'emploi d'un environnement de transformations de modèle est facilité pour un utilisateur qui doit développer un système dans un contexte précis.

L'identification précoce des langages de développement et des transformations de modèles n'est que l'une des possibilités offertes par la formalisation des processus. L'analyse des projets antérieurs et la détermination d'un comportement récurrent doit permettre de faciliter d'autres aspects du projets. En améliorant la connaissance préalable des éléments nécessaires à la conduite du projet, nous espérons faciliter leur déroulement et favoriser l'anticipation.

4.5 Conclusion

Ce chapitre est venu compléter les travaux théoriques présentés jusqu'alors selon trois axes.

Dans la première section, nous avons non seulement donné à l'utilisateur les moyens de s'assurer de la cohérence des modèles qu'il manipule mais aussi les moyens de s'assurer que des règles de cohérence entre modèles distincts sont bien respectées. Le développement d'une telle méthode et d'un outil de validation est l'une des contributions de nos travaux. La validité des modèles de processus étant une thématique peu traitée et la validité des relations inter–modèles, dans le cas des processus, n'ayant jusqu'alors pas été envisagée.

Dans la seconde section, nous nous sommes intéressés à l'application de notre méthodologie à un projet concret. Cette application a permis de démontrer la pertinence de l'approche tout en venant l'enrichir. On a pu notamment confronter notre approche aux problématiques posées par le suivi d'un projet.

Enfin, dans la troisième section, nous avons proposé une extension de la démarche en introduisant une notion de modèle de système propre au MDE. Nous montrons ainsi que disposer d'une représentation conjointe des processus et de la structure du système permet le développement de nouveaux modes de conception. L'adaptation que nous proposons peut être à la base d'une application des concepts défendus par la démarche d'ingénierie système guidée par les modèles.

Conclusion générale et perspectives

Les systèmes conçus par l'Homme et fabriqués industriellement sont de plus en plus complexes. Leur développement a donné lieu à l'émergence d'une discipline nouvelle : l'ingénierie système qui a pour ambition d'élaborer les méthodes et les outils nécessaires à ces développements. Des efforts de plus de cinquante ans ont permis d'élaborer de nombreuses normes et recommandations donnant lieu à des outils de conception et de vérification. L'analyse que nous faisons dans le chapitre 1 de notre présentation est que, si l'inventaire des besoins est parfaitement rassemblé et si les processus à employer sont bien identifiés, la mise en œuvre reste encore ouverte à l'improvisation. Notre idée est donc de profiter des acquis et de proposer une méthodologie nouvelle basée sur :

- la formalisation d'un processus générique,
- une démarche d'adaptation aux spécificités métiers et projets,
- le développement d'outils pour la validation des processus formalisés et leurs mises en œuvre effective.

Pour développer notre méthodologie, nous avons choisi de l'appliquer à un standard de référence de l'ingénierie système : l'EIA-632. Notre première étape a été, partant des recommandations textuelles de cette norme, de choisir un langage de modélisation, SPEM/UML, et de construire un modèle formel de l'EIA-632. Ce modèle ainsi que la démarche de modélisation qui a conduit à sa création sont présentés au chapitre 2 de cette thèse. Cette démarche n'est cependant pas limitée à l'EIA-632 et, dans le cas où un autre des standards de l'ingénierie système (IEEE 1220 ou l'ISO 15288) est employé comme référant, elle peut être utilisée de la même manière pour construire un modèle générique des processus du projet quel que soit le processus de référence considéré.

Si les processus génériques décrits dans les normes d'ingénierie système sont à la base de la conduite et du suivi du projet, on sait, par les recommandations des normes elles–mêmes, qu'il est nécessaire d'adapter ces processus à l'environnement du projet. Cette étape d'adaptation, pourtant indispensable en pratique à l'application des processus, se fait actuellement de manière désordonnée et incontrôlée. La seconde de nos contributions a été de montrer, au cha-

pitre 3, que cette spécialisation des processus pouvait être, elle aussi, formalisée. Nous avons choisi de réaliser cette spécialisation en deux étapes : spécialisation au métier, puis spécialisation au projet. **La définition d'une démarche de formalisation innovante, reposant sur les concepts de l'ingénierie des modèles, dans laquelle les recommandations normatives se voient spécialisées à des domaines d'activité et à des projets précis conformément à la philosophie d'application de l'ingénierie système, est le cœur de nos travaux.** Grâce à elle, nous avons montré qu'il est possible de maîtriser de bout en bout l'application des processus de l'ingénierie système en disposant, au final, de processus adaptés à un cadre d'application tout en respectant les règles de bonnes pratiques définies dans les normes d'ingénierie système.

Notre processus de formalisation et de spécialisation des recommandations des normes de l'ingénierie système étant défini, nous avons, dans le dernier chapitre de cette thèse, tenté de renforcer notre démarche pour la rendre pleinement opérationnelle :

- Nous avons donné à l'utilisateur des moyens et des outils originaux de validation de modèles de processus et de transformations de modèles mises en jeux dans notre démarche de formalisation/spécialisation. On s'est notamment intéressé à la question de la conservation des propriétés lors des opérations de spécialisation des processus.
- Nous avons montré comment peut se faire la mise en œuvre de processus tels que nous les proposons. Pour cela, nous avons illustré la mise en application des processus de l'EIA-632 dans le cadre d'un projet de conception didactique.
- Enfin, nous avons montré que la formalisation des processus est une première étape indispensable à la mise en place d'une démarche d'ingénierie système guidée par les modèles. En cela, nous nous rapprochons de projets tels que TOPCASED qui tendent à la création de plates–formes de développement intégralement orientées modèles.

En définissant un processus de formalisation des recommandations normatives de l'ingénierie système, nous contribuons d'une part à la mise en œuvre industrielle de l'IS et, d'autre part, aux avancées scientifiques de ce domaine.

Du point de vue industriel, ce travail permet d'aider un chef de projet dans la mise en place d'une démarche d'ingénierie système sur son projet. Nous lui permettons de profiter des avantages d'une telle démarche en l'assistant dans les problèmes concrets qu'il rencontre. La formalisation des normes d'IS, conformément aux principes de l'IDM, l'aide à s'approprier et à communiquer les concepts qu'elles contiennent. Les déviations et les adaptations, nécessaires à l'application des normes dans un contexte donné, sont rendues explicites et maîtrisées.

Enfin, une fois construit, le modèle métier est réutilisable dans d'autres projets ce qui permet de rentabiliser cet effort initial de modélisation.

A noter que l'application de ce processus met en évidence les contradictions ou les incohérences qui peuvent survenir dans l'application des normes dites « métier ».

C'est pourtant sur une échelle plus large qu'apparaît la contribution majeure de ce travail. Notre méthodologie couvre l'ensemble des projets nécessaires à l'ingénierie d'un système. **Elle permet d'appliquer les processus structurants de l'ingénierie système sur tous les projets impliqués tout en laissant chaque service ou équipe de projet libre d'effectuer des adaptations nécessaires à sa spécialité et à son cadre de travail. On s'assure ainsi d'une cohérence globale dans la réalisation du système sans pour autant brider les intervenants.** Dans le contexte actuel, où la complexité des systèmes dépasse largement les compétences d'*une* équipe de conception et où se multiplient les interventions ponctuelles centrées sur *une* fonctionnalité du système (citons par exemple le développement de l'Airbus A380), il est en effet essentiel de laisser à chacun une liberté de fonctionnement propre à sa spécialité tout en s'assurant, au plus haut niveau, d'une organisation et d'une cohérence générale du déroulement du projet.

D'un point de vue scientifique, nos travaux ont donné lieu à de nombreuses avancées de la discipline d'ingénierie système.

Nous avons montré qu'une formalisation des recommandations de l'ingénierie système est possible. Cette formalisation a de nombreux avantages, dans la rédaction et dans l'application des normes. Accompagner systématiquement un standard de son modèle :

- permet une rédaction des recommandations sur la base d'un modèle et non plus à partir d'un document textuel,
- évite les problèmes d'interprétation et garantit une communication efficace des concepts,
- facilite la mise en application des processus décrits,
- et, pour peu qu'elles soient elles–aussi formalisées, assure la compatibilité de normes complémentaires avec les normes générales.

Pour cela, nous avons choisi un langage de description de processus en SPEM/UML qui permet la représentation des concepts majeurs de l'ingénierie système tels que ses processus, le concept de cycle de vie ou encore la structure récursive de système et ses blocs de constructions.

Sur la base de ce langage, nous avons construit un modèle complet de l'EIA-632 à partir duquel ont été mis en évidence des comportements implicites ou cachés.

Nous avons montré que le concept de spécialisation des processus, tel qu'il est recommandé par l'ingénierie système, peut être fait de manière formelle et structurée en se plaçant dans le paradigme de l'IDM. Nous avons donné corps à ce concept en définissant une démarche de spécialisation reposant sur des transformations de modèles des processus génériques décrits par les normes d'ingénierie système. Nos travaux se trouvent à l'intersection de l'ingénierie système [Che67, SC95, INC04, Adc07] et de l'ingénierie des modèles [Béz05, Sch06] et peuvent s'apparenter à l'ingénierie des méthodes [RSM95, BS96, Dah98, RR01b, RR01a, Den01] employée dans les systèmes d'information. Ils s'attachent cependant à la définition de processus adaptés au développement de systèmes au sens large : système physique, système logiciel, services ou encore une combinaison de ces derniers. Un des intérêts majeurs de la maîtrise de la spécialisation des processus est qu'elle permet de détecter l'introduction de comportements (désirés ou non) dans les processus spécialisés qui seront effectivement mis en œuvre. Dans cette optique, nous avons défini un outil de validation de modèles de processus qui vient compléter notre méthodologie en permettant de s'assurer de la présence, ou de l'absence, de propriétés logiques dans un modèle.

Nous avons illustré comment un modèle formel peut être utilisé pour guider la conduite des processus de projet et comment il structure l'organisation et le déroulement du projet.

Enfin, par notre démarche, nous avons montré que l'ingénierie des modèles dépasse le cadre de l'ingénierie logicielle et qu'elle peut être appliquée avec succès dans d'autres domaines. En l'associant aux transformations de modèles de système plus classiques déjà utilisées en conception, nous sommes convaincus que notre démarche de formalisation peut être à la base d'une nouvelle méthodologie de gestion de projet proche de l'ingénierie système guidée par les modèles.

Plusieurs limitations sont cependant à prendre en compte dans l'application de notre démarche.

Premièrement, il ne faut pas perdre de vue que l'objectif premier d'un projet est de réaliser un système conforme à ses exigences dans le respect des coûts, des délais et des contraintes fixées par le cahier des charges. Si l'application de bonnes pratiques peut aider à atteindre cet objectif, elle ne garantit pas d'y arriver à coup sûr. En se basant sur de bonnes pratiques, notre proposition agit de même : si elle favorise la réussite d'un projet par son action structurante, elle ne peut garantir sa réussite.

D'autre part, si l'application de notre méthode permet de détecter des conflits entre recommandations, la résolution de ces conflits est en revanche laissée libre. Dans le cas d'un conflit,

il est de la responsabilité de chacun de choisir quelles recommandations suivre et quelles recommandations écarter.

Les aspects humains de l'application de cette démarche sont aussi à prendre en compte. Son rôle structurant dans l'organisation inter–projets peut ne pas être perçue immédiatement et, de ce fait, être ressentie comme une contrainte par les chefs de projet. Elle doit s'accompagner d'une action de formation préalable pour éviter qu'elle ne soit rejetée par les acteurs du projet.

Enfin, la dernière limitation est intrinsèque à la démarche d'ingénierie système. Les processus décrits sont composés essentiellement d'actions intellectuelles de conceptualisation du produit, de ses composants ou de ses produits contributeurs qui retardent le lancement effectif de la production. On attend que le produit soit entièrement spécifié et défini avant de commencer la production du moindre composant. Dans la pratique cependant il est nécessaire de donner du travail en continu à la production ou de prendre des risques en anticipant par rapport au processus idéal. Dans ce cas, se pose le problème de la gestion des déviations par rapport au comportement attendu.

Plusieurs perspectives se dégagent des travaux présentés dans ce mémoire de thèse.

La première d'entre elles concerne l'outillage de la démarche. Il serait souhaitable d'associer à notre démarche une plate–forme dédiée qui permette la gestion des modèles et les opérations de transformation et de vérification des modèles.

À cela s'ajoutent des perspectives de recherche. On peut s'intéresser notamment à l'intégration de modèles de maturité : les modèles de maturité sont des indicateurs utilisés pour mesurer le degré de maturité que l'entreprise a acquis dans l'application des processus de l'IS. Il pourrait être intéressant de les formaliser à leur tour pour disposer d'un indicateur adapté aux processus formalisés.

Un second axe de recherche pourrait être de s'intéresser non plus à des projets sur le point de démarrer mais à des projets déjà engagés. Il a été montré que, dans ce cas, la mise en place d'une démarche d'ingénierie système doit se faire par des actions spécifiques. Dans ce cadre, on peut s'intéresser à la prise en compte du passif dans la démarche et à la représentation de la migration des processus initiaux à des processus d'IS.

Enfin, le dernier axe, sans doute le plus prometteur, serait de pousser à bout la démarche d'ingénierie guidée par les modèles. Nous avons, dans le dernier chapitre de ce mémoire, montré quelles pouvaient être les bases de cette réflexion. Cette nouvelle vision de la démarche d'ingénierie souffrait jusqu'à présent d'un manque de maîtrise et de formalisation des proces-

sus. Nos travaux pourraient être à la base d'un couplage effectif entre processus d'ingénierie et évolution des modèles de système au cours de son cycle de vie.

Bibliographie

[Aca] Académie française. Dictionnaire de l'académie française. http ://a-tilf.atilf.fr/academie9.htm. neuvième édition.

[Adc07] Rick Adcock. Principles and practices of systems engineering. http://incose.org.uk/Downloads/AA01.1.4_Principles%20&%20practices%20of%20SE.pdf, 2007.

[AFI07] AFIS. Présentation de l'AFIS et de l'Ingénierie Système. Plaquette de présentation générale de l'Association Française d'Ingénierie Système, juillet 2007.

[AFN00] AFNOR. FD X50-179 management de la qualité – guide pour l'identification des exigences des clients, Décembre 2000.

[AFN02] AFNOR. FD X50-127 outils de management – maîtrise du processus de conception et développement, Avril 2002.

[AFN03a] AFNOR. FD X50-128 outils de management – lignes directrices pour le processus achat et approvisionnement, Mai 2003.

[AFN03b] AFNOR (Association Française de Normalisation). AFNOR Z 67-288 Systems Engineering - System Life-Cycle Processes, Novembre 2003.

[AFN04] AFNOR. FD X50-189 systèmes de management – lignes directrices pour leur intégration, Janvier 2004.

[AFN05] AFNOR. FD X50-176 outils de management – management des processus, Octobre 2005.

[APFPB+06] Arnaud Albinet, Marie-Agnès Péraldi-Fradi, Dimitru Potop-Butucaru, Yves Sorel, Olivier Casse, Jean-Louis Boulanger, Hubert Dubois, Stéphane Badreau, and Sylvain Begoc. Méthode de modélisation pour la valida-

tion et la traçabilité des exigences (memvatex). Technical report, CEA-List, 2006. http ://www.memvatex.org/home/liblocal/docs/MeMVaTEx-SOTA-Website.pdf.

[Bar97] Sylvain Barbeau. *Méthodologie d'ingénierie des systèmes aérospatiale*. PhD thesis, Université de Nantes, 1997.

[BB01] Terry A. Bahill and Clarck Briggs. The systems engineering started in the middle process : A consensus of systems engineers and project managers. *Systems Engineering*, 4(2) :156–167, 2001.

[BBLC+03] Pierre Bazex, Jean-Paul Bodeveix, Christophe Le Camus, Thierry Millan, and Christian Percebois. Vérification de modèles UML fondée sur OCL. In *INFORSID, Nancy, 03/06/03-06/06/03*, pages 185–200, http ://praxinsa.insa-lyon.fr, juin 2003. Inforsid (actes électroniques).

[BCC+00] Loyd Baker, Paul Clemente, Bob Cohen, Larry Permenter, Byron Purves, and Pete Salmon. Foundational concepts for model driven system design. white paper, INCOSE Model Driven System Design Interest Group, 2000.

[Béz05] Jean Bézivin. On the unification power of models. *Software and System Modeling*, 4(2) :171–188, 2005.

[BG98] A. T. Bahill and B. Gissing. Re-evaluating systems engineering concepts using systems thinking. *IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews*, 28(4) :516–527, 1998.

[Bla00] Jean-Christophe Blaise. *Apport d'une modélisation de l'information normative à l'intégration des règles de sécurité des machines en conception*. PhD thesis, CRAN, Nancy, 2000.

[BLWvH05] G.Maarten Bonnema, Ilanit F. Lutters-Weustink, and Fred J.A.M. van Houten. Introducing systems engineering to industrial design engineering students with hands-on experience. In *18th International Conference on Systems Engineering*, pages 408–413, 16-18 August 2005.

[Boa95a] John Boarder. Systems engineering as a process. In *IEEE Colloquium on Systems Engineering for Profit*, pages 1–4, 22 March 1995.

[Boa95b] John C. Boarder. The system engineering process. In *Proceedings of 1995 IEEE Annual International Engineering Management Conference, 'Global Engineering Management : Emerging Trends in the Asia Pacific'*, pages 293–298, 25-28 June 1995.

[Bri04] British Standards Institution. SO/IEC 19760 – System Engineering – A guide for the Application of ISO/IEC 15288 System Life Cycle Processes, February 23 2004.

[BRJ00] Grady Booch, James Rumbaugh, and Ivar Jacobson. *Guide de l'utilisateur UML*. Eyrolles, 3 mars 2000.

[BS96] S. Brinkkemper and M. Saeki. *Method Engineering : Principles of method construction and tool support*. Chapman &Hall, 1996.

[BY02] Yaneer Bar-Yam. Large scale engineering and evolutionary change : Useful concepts for implementation of forcenet. Technical report, New England Complex Systems Institute, 2 September 2002.

[BY03] Yaneer Bar-Yam. When systems engineering fails-toward complex systems engineering. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 2021 – 2028, 5-8 October 2003.

[BYAB+04] Yaneer Bar-Yam, Mary Ann Allison, Ron Batdorf, Hao Chen, Hòa Generazio, Harcharanjit Singh, and Steve Tucker. The characteristics and emerging behaviors of system-of-systems. Technical report, NECSI : Complex Physical, Biological and Social Systems Project, January 7 2004.

[Béz04] Jean Bézivin. Sur les principes de base de l'ingénierie des modèles. *L'Objet*, 10(4) :145–157, 2004. Colloque en l'honneur de J.-F. Perrot.

[Cap01] George F. J. Caple. The generic unified systems engineering metamodel - a diagramatic representation of the total systems engineering process. In *The 20th Conference on Digital Avionics Systems*, pages 4E3/1–4E3/9, 14-18 October 2001.

[Cap03] F. J. Caple, George. The route to the intelligent systems engineering enterprise. In *DASC'03 The 22nd Digital Avionics Systems Conference*, volume 2, pages 6.B.3.1– 6.B.3.10, 12-16 October 2003.

[CCCC06a] Benoit Combemale, Alain Caplain, Xavier Crégut, and Bernard Coulette. Vers une vérification d'un procédé de développement modélisé en SPEM. In *Formalisation des Activités Concurrentes*, Toulouse, France, 23/03/06-24/03/06 2006.

[CCCC06b] Benoit Combemale, Xavier Crégut, Alain Caplain, and Bernard Coulette. Modélisation rigoureuse en SPEM de procédé de développement. In *Conférence sur les Langages et Modèles à Objets*, pages 135–150, Nîmes, France, 22 mars 2006. Hermès Science Publications.

[Cha99] Frederic Chambolle. *Un modèle produit piloté par les processus d'élaboration : Application au secteur automobile dans l'environnement STEP*. PhD thesis, Centrale Paris, 1999.

[Che67] Harold Chestnut. *Systems Engineering Methods*. Wiley, 1967.

[Cla02] Siobhàn Clarke. Extending standard uml with model composition semantics. *Sci. Comput. Program.*, 44(1) :71–100, 2002.

[CLCF+05] Agusti Canals, Christophe Le Camus, Marion Féau, Guillaume Jolly, Vincent Bonnafous, and Petre Bazavan. Une utilisation opérationnelle d'ATL : l'intégration de la transformation de modèles dans le projet TOPCASED. *Génie Logiciel*, 73 :21–26, 2005.

[CNR] CNRT Aéronautique & Espace. Projet tocased. site web http ://www.topcased.org/.

[Col06] Philippe Collet. État de l'art sur la contractualisation et la composition. Technical report, Laboratoire d'Informatique Fondamentale de Lille, Aout 2006. Chapitre 5 Ingénierie des modèles.

[Com05] Benoit Combemale. Spécification et vérification de modèles de procédés de développement. Rapport de master, Institut National Polytechnique de Toulouse, juin 2005.

[Dah98] A. N. W. Dahanayake. Evaluation of the strength of computer aided method engineering for product development process modeling. In *DEXA '98 : Proceedings of the 9th International Conference on Database and Expert Systems Applications*, pages 394–410, London, UK, 1998. Springer-Verlag.

[DC90] P. Dumas and G. Charbonnel. *La méthode OSSAD*. Editions d'organisation, 1990.

[Def01] Defense Acquisition University Press. System engineering fundamentals. http ://www.dau.mil/pubs/pdf/SEFGuide%2001-01.pdf, January 2001.

[Den01] R. Deneckere. *Approche d'extension de méthodes fondée sur l'utilisation de composants générique*. PhD thesis, University of Paris 1-Sorbonne, 2001.

[DMSS07] H. Demmou, M. Messaadia, N. Sadou, and A.E.K. Sahraoui. Intégration de la sûreté de fonctionnement dans les processus d'ingénierie système. In *7ème Congres International de Génie Industriel*, Trois Rivières (Canada), 5-8 juin 2007.

[DOD07a] DOD. Dod architecture framework version 1.5, volume i : Definitions and guidelines, 23 April 2007. http ://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_I.pdf.

[DOD07b] DOD. Dod architecture framework version 1.5, volume ii : Product descriptions, 23 April 2007. http ://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_II.pdf.

[DOD07c] DOD. Dod architecture framework version 1.5, volume iii : Architecture data description, 23 April 2007. http ://www.defenselink.mil/cio-nii/docs/DoDAF_Volume_III.pdf.

[Dor06] Teresa Doran. IEEE 1220 : For Practical Systems Engineering. *Computer*, 39(5) :92–94, May 2006.

[DSLT04] V. Debruyne, F. Simonot-Lion, and Y. Trinquet. EAST-ADL an Architecture Description Language, Validation and Verification Aspects. In *IFIP World Computer Congress - Workshop on Architecture Description Language*, Toulouse, France, 27 August 2004.

[Ele99] Electronic Industries Alliance. EIA-632 – Processes for Engineering a System, January 1999.

[EMW06] Bill Esler, Keith Mason, and Ann Williams. L'initiative SPEEDS de l'union européenne vise à atteindre une position leader mondial en

conception de systèmes embarqués. Communiqué de presse, 2006. http ://www.humbugpr.com/speeds/SPE001F-Launch-FR.doc.

[Est07] Jeff A. Estefan. Survey of model-based systems engineering (MBSE) methodologies. Technical report, INCOSE MBSE Focus Group, 25 May 2007.

[FG05] Patrick Farail and Pierre Gaufillet. TOPCASED un environnement de développement open source pour les systèmes embarqués. *Génie Logiciel - Magazine de l'ingéniérie du logiciel et des systèmes*, 73, June 2005.

[FM99] Kevin Forsberg and Harold Mooz. System engineering for faster, cheaper, better. In *Proceedings of the ninth annual international symposium of the INCOSE*, 1999. http ://www.incose.org/sfbac/welcome/fcb-csm.pdf.

[Fre84] R. E. Freeman. *Strategic Management : A Stakeholder Approach*. Pitman, Boston, 1984.

[Fri03] Sanford A. Friedenthal. UML for systems engineering request for proposal. Technical report, OMG, March 2003.

[Gau06] Pierre Gaufillet. Topcased - an eclipse-based development environment for critical systems and softwares. In *Eclipse Summit Conference*, Esslingen, Germany, October 2006.

[GE07] Citlalih Gutiérrez-Estrada. *Méthodes et outils de la conception système couplée à la conduite de projet*. PhD thesis, LESIA, INSA de Toulouse, 6 février 2007.

[GFD05] Tudor Girba, Jean-Marie Favre, and Stéphane Ducasse. Using meta-model transformation to model software evolution. *Electr. Notes Theor. Comput. Sci.*, 137(3) :57–64, 2005.

[Gir99] Philippe Girard. *Etude de la conduite de la conception des produits manufacturés Contribution à l'ingénierie des systèmes de conception*. PhD thesis, Université Bordeaux 1, 1999.

[Gla06] Robert L. Glass. The standish report : does it really describe a software crisis ? *Commun. ACM*, 49(8) :15–16, 2006.

[Gro97] Groupe de travail IS (Ingénieire Système). Ingénierie système pourquoi ? – comment ? Technical report, Association Française d'Ingénieirie Système, 1997.

[Gui07] David Guihal. *Modélisation en langage VHDL-AMS des systèmes pluridisciplinaires*. PhD thesis, Ecole doctorale GEET, LAAS, 7 Av de Colonel Roche, Toulouse France, 25 mai 2007.

[HH04] P. Hastono and S.A. Huss. Automatic generation of executable models from structured approach real-time specifications. In *Proceedings The 25th IEEE International Real-Time Systems Symposium (RTSS)*, Lisbon, Portugal, 5-8 December 2004.

[HKK04] B. Hardung, T. Kölzow, and A. Krüger. Reuse of software in distributed embedded automotive systems. In *Fourth ACM International Conference on Embedded Software (EMSOFT)*, Pisa, Italy, 27-29 September 2004.

[Hon04] Eric C. Honour. Understanding the value of systems engineering. In *Proceedings of the INCOSE International Symposium*, Toulouse, France, 2004.

[Hon06] Eric C. Honour. A Practical Program of Research to Measure Systems Engineering Return on Investment (SE-ROI). In *Proceedings of the INCOSE International Symposium*, Orlando, FL, 2006.

[HS04] Raf Haesen and Monique Snoeck. Implementing consistency management techniques for conceptual modeling. In *UML 2004 Workshop on Consistency Problems in UML-based Software Development III*, pages 99–113, Lisbon Portugal, October 2004.

[INC04] INCOSE. Systems engineering handbook. Technical report, INCOSE, 2004.

[Ins03] Project Management Insitute. *Management de projet Un référentiel de connaissances*. AFNOR, 2003. 4ème tirage.

[IRI07] IRIT. Projet domino, 2007. Site web http ://neptune.irit.fr/Public/AnglaisV2/Domino/ficheResumeDOMINO.html.

[ISA07] ISAF. Règles de classe international un mètre. http ://www.classe1metre.org/site/download/jauge/, 2007. V1.

[ISO04] ISO/CEI. Directives ISO/CEI – partie 2 : Règles de structure et de rédaction des Nomes internationales, 2004. cinquième édition (§ 3.1.).

[IST07] IST Sixth Framework Programme. Projet modelplex, 2007. Site web http ://www.modelplex-ist.org/.

[ITE] ITEA. Projet itea east–eea. Site web http ://www.east-eea.net.

[Jam05a] Sonia Jamal. *Environnement de procédé extensible pour l'orchestration. Application aux services web*. PhD thesis, Université Joseph Fourier – Grenoble I, 13 décembre 2005.

[Jam05b] Mo Jamshidi. System-of-systems engineering – a definition. In *IEEE SMC 2005*, Big Island, Hawaii, 2005. http ://ieeesmc2005.unm.edu/SoSE_Defn.htm.

[JB02] Julian Johnson and Jozsef Bedocs. System engineering conceptual model. In *OMG Technical Meeting, SE Domain Special Interest Group (SE DSIG), INCOSE International Workshop*, January 2002. initial conceptual model.

[JD07] P. A. "Trisha" Jansma and Mary Ellen Derro. If you want good systems engineers, sometimes you have to grow your own ! In *IEEE Aerospace Conference*, pages 1–15, 3-10 March 2007.

[JMS02] Natalia Juristo, Ana M. Moreno, and Andrés Silva. Is the european industry moving toward solving requirements engineering problems ? *IEEE Software*, 19(2) :70–77, Nov/Dec 2002.

[Jou07] Deborah Jouin. Glossaire de la WfMC. Technical report, Workflow Management Coalition, 2007.

[Kea05] Charles Keating. Research foundations for system of systems engineering. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 2720–2725, Waikoloa, Hawaii, October 10–12 2005.

[Kru95] Philippe Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6) :42–50, 1995.

[KS07] J. Konate and A.E.K. Sahraoui. Collaboration in requirements engineering process. In *13th International Conference on Concurrent Enterprising (ICE 2007)*, pages 107–114, Sophia Antipolis (France), 4–6 juin 2007.

[Lab04] Michel Labrousse. *Proposition d'un modèle conceptuel unifié pour la gestion dynamique des connaissances d'entreprise.* PhD thesis, Centrale Nantes, 2004.

[Lar03] Etienne Lardeur. *Amélioration de la performance de l'ingénierie dans un contexte d'Ingénierie Système : Cas du développement conjoint des produits*

automobiles et de leurs systèmes de fabrication. PhD thesis, Centrale Paris, 2003.

[LFM00] Howard Lykins, Sanford Friedenthal, and Abraham Meilich. Adapting UML for an Object Oriented Systems Engineering Method (OOSEM). In *Proceedings of the 2000 INCOSE Symposium*, 2000.

[LMO05] Hervé Leblanc, Thierry Millan, and Ileana Ober. Démarche de développement orienté modèles : de la vérification de modèles à l'outillage de la démarche . In Sébastien Gérard, Jean-Marie Favre, Pierre-Alain Müller, and Xavier Blanc, editors, *Ingénierie Dirigée par les Modèles , Paris, 30/06/05-01/07/05*, pages 125–140. CEA List - ISBN 2-7261-1284-6, juin 2005.

[Lon03] Barthélémy Longueville. *Capitalisation des processus de décision dans les projets d'innovation : application à l'automobile*. PhD thesis, Centrale Paris, 2003.

[Mal06] Hugues Malgouyres. *Définition et détection automatique des incohérences structurelles et comportementales des modèles UML – Couplage des techniques de métamodélisation et de vérification basée sur la programmation logique*. PhD thesis, LESIA, INSA de Toulouse, 28 novembre 2006.

[Mar98a] James N. Martin. Evolution of eia 632 from an interim standard to a full standard. In *INCOSE 1998 Symposium*, 1998.

[Mar98b] James N. Martin. Overview of the EIA 632 Standard – Processes for Engineering a System, 1998.

[Mer01] Samuel Mercier. L'apport de la théorie des parties prenantes au management stratégique : une synthèse de la littérature. In *Xième Conférence de l'Association Internationale de Management Stratégique*, 13–15 juin 2001.

[MHLH05] Chantal Morley, Jean Hugues, Bernard Leblanc, and Olivier Hugues. *Processus Métiers et systèmes d'information : Evaluation, modélisation, mise en œuvre*. InfoPro. Dunod, mars 2005.

[MJS05] M. Messaadia, M.H.El Jamal, and A.E.K Sahraoui. On systems engineering deployment and requirements evolution. In *18th International Conference on Systems Engineering (ICEng'05)*, pages 427–433, Las Vegas (USA), 16–18 Août 2005.

[MM06] Hugues Malgouyres and Gilles Motet. A UML model consistency verification approach based on meta-modeling formalization. In Hisham Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC)*, pages 1804–1809, Dijon, France, April 23–27 2006. ACM.

[Moda] Model Futures Ltd. http ://www.modaf.com/.

[Modb] Model Futures Ltd. http ://www.modaf.org.uk/.

[MVDS05] Tom Mens and Ragnhild Van Der Straeten. On the use of formal techniques to support model evolution. In Sébastien Gérard, Jean-Marie Favre, Pierre-Alain Muller, and Xavier Blanc, editors, *IDM05, Actes des 1ères Journées sur l'Ingénierie Dirigée par les Modèles*, 2005. http ://idm.imag.fr/idm05/actes.pdf.

[oEE99] IEEE (Institute of Electrical and Electronics Engineers). IEEE 1220 Standard for application and Management of the Systems Engineering Process, Janvier 1999.

[Oli95] David W. Oliver. Systems engineering and software engineering, contrasts and synergism. In *Proceedings of the 1995 International Symposium and Workshop on Systems Engineering of Computer Based Systems*, pages 125–132, 1995.

[Oli03] David W. Oliver. System engineering conceptual model (draft 12). http ://syseng.omg.org/SE_Conceptual%20Model/SE_Conceptual_Model.htm, March 2003.

[OMG05] OMG. Software process engineering metamodel specification, January 2005. Version 1.1.

[OMG06a] OMG. Business process modeling notation specification, February 2006. Final adopted specification.

[OMG06b] OMG. SysML Specification v. 1.0, May 2006. Final Adopted Specification.

[OMG07] OMG. Software process engineering metamodel specification, March 2007. Version 2.0.

[oT06] Georgia Institute of Technology, editor. *Frontiers in Design & Simulation Research*, March 2006.

[oT07] Georgia Institute of Technology, editor. *Frontiers in Design & Simulation Research*, May 2007.

[Par07] SysML Partners. Sysml faq. http ://www.sysmlforum.com/FAQ.htm, 2007.

[Ped06] James E. Pederson. Creating a tool independent system engineering environment. In *IEEE Aerospace Conference*, page 8 p, 4-11 March 2006.

[Pel03] René Pelfresne. Modéliser, pourquoi ? comment ? Journée d'étude – ADBS, 20 mai 2003.

[Pen97] Jean Michel Penalva. *La modélisation par les systèmes en situation complexes*. PhD thesis, Université Paris 11, 1997.

[PET] PETZL. Epi et normes. http ://fr.petzl.com/petzl/frontoffice/static/EPI/normes/norGene_FR.j

[PMI96] PMI Standrds Committee. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. Project Management Institute, 1996. Third Edition.

[RB06] S. Rochet and C. Baron. *Handbook of Research on Nature Inspired Computing for Economics and Management*, volume 2, chapter An Evolutionary Algorithm for Decisional Assistance to Project Management. Rennard, Jean-Philippe, 2006.

[RO03] Jog Roj and Martin Owen. Bpmn and business process management. Technical report, Popkin Software, September 2003.

[Rom96] Mohamed Romdhani. *Ingénierie des systèmes complexes avec la méthode de conception concurrente co-design matériel/logiciel*. PhD thesis, INPG, 1996.

[RR01a] J. Ralyté and C. Rolland. An approach for method reengineering. In *Proceedings of the 20th International Conference on Conceptual Modeling*, Yokohama, Japan, 2001.

[RR01b] J. Ralyté and C. Rolland. An assembly process model for method engineering. In *Proceedings of the 13th CAISE'01*, Interlaken, Switzerland, 2001.

[Rés] Réseau National de recherche et d'innovation en Technologies Logicielles (RNTL). Projet openembededd. site web http ://openembedd.org/.

[RSM95] C. Rolland, C. Souveyet, and M. Moreno. An approach for defining ways of working. *Information Systems*, 20(4) :337–359, 1995.

[SC95] R. Shishko and R.G. Chamberlain. Nasa systems engineering handbook. Technical Report SP-610S, NASA Center for AeroSpace Information, June 1995. http ://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19960002194_1996102194.pdf.

[SC07] Avi Soffer and Shalom Cohen. Alleviating complexity of modeling in system engineering tools. In *ICSEM '07 International Conference on Systems Engineering and Modeling*, pages 110–117, 20-23 March 2007.

[Sch06] D.C. Schmidt. Model-driven engineering. *IEEE Computer*, 39 :25–31, February 2006.

[SG86] Alfred Z. Spector and David K. Gifford. A computer science perspective of bridge design. *Commun. ACM*, 29(4) :267–283, 1986.

[SGE+04] Greg Straw, Geri Georg, Song Eunjee, Sudipto Ghosh, Robert France, and James M. Bieman. Model composition directives. *Lecture notes in computer science*, 3273 :84–97, 2004.

[SRB+06] Silvers, Julia Rutherford, Bowdin, A. J. Glenn, O'Toole, J. William, Nelson, and Kathleen Beard. Towards an international event management body of knowledge (embok). *Event Management*, 9(4) :185–198, January 2006.

[SSW+06a] Konstantinos Sagonas, Terrance Swift, David S. Warren, Juliana Freire, Prasad Rao, Baoqiu Cui, Ernie Johnson, Luis de Castro, Rui F. Marques, Steve Dawson, and Michael Kifer. The XSB System Version 3.0 Volume 1 : Programmer's Manual. Technical report, XSB, July 2006.

[SSW+06b] Konstantinos Sagonas, Terrance Swift, David S. Warren, Juliana Freire, Prasad Rao, Baoqiu Cui, Ernie Johnson, Luis de Castro, Rui F. Marques, Steve Dawson, and Michael Kifer. The XSB System Version 3.0 Volume 2 : Libraries, Interfaces and Packages. Technical report, XSB, July 2006.

[Sta94] Standish Group. The chaos report. Technical report, Standish Group International, 1994. www.standishgroup.com/sample_ research/PDFpages/-Chaos1994.pdf.

[Sta99] Standish Group. Chaos : A recipe for success. Technical report, Standish Group International, 1999. http ://www.standishgroup.com/sample_research/PDFpages/chaos1999.pdf.

[Sta01] Standish Group. Extreme chaos. the 2001 update to the chaos report. Technical report, Standish Group International, 2001.

[SV06] Jean-Pierre Seuma Vidal. *Maîtrise de la cohérence des modèles UML d'applications critiques*. PhD thesis, Institut National des Sciences Appliquées de Toulouse, Laboratoire d'Etude des Systèmes Informatiques et Automatiques, 19 Juin 2006.

[SV07] Alberto L. Sangiovanni-Vincentelli. Quo vadis sld : Reasoning about trends and challenges of system-level design. *Proceedings of the IEEE*, 95(3) :467–506, March 2007.

[SYS] SYSTEM@TIC. Projet opendevfactory. Site web http ://www.usine-logicielle.org.

[Tho93] Bernhard Thomé, editor. *Systems engineering : principles and practice of computer-based systems engineering*. John Wiley and Sons Ltd., Chichester, UK, UK, 1993.

[TMHS06] Caroll Thronesbery, Jane T. Malin, Kritina Holden, and Danielle P. Smith. Tools to support human factors and systems engineering interactions during early analysis. In *IEEE Aerospace Conference*, pages 1–9, 4-11 March 2006.

[Var05] Élise Vareilles. *Conception et approches par propagation de contraintes : contribution à la mise en œuvre d'un outil d'aide interactif*. PhD thesis, Institut national polytechnique de Toulouse, Centre de Génie Industriel de l'École des Mines d'Albi-Carmaux, 24 Juin 2005 2005.

[VdBH+04] S Voget, G. de Boer, P. Heidl, F. Adis, and Virnich U. Analysis of vehicle wide software concepts within the European funding project ITEA-EAST/EEA. In *VDI conference AUTORAG*, Waldorf-Wiesloch, Germany, March 2004.

[WAHH99] E. R. Widmann, G. E. Anderson, G. J. Hudak, and T. A. Hudak. The taxonomy of systems engineering competency for the new millennium. In *Proceedings of the ninth annual international symposium of the INCOSE*, 1999.

[Whi04] Stephen A. White. Introduction to BPMN. Technical report, IBM Corporation, May 2004.

[Wik07] Wikipedia. Activité socioéconomique, 2007.

[WW03] David Watt and Keith Willey. The project management - systems engineering dichotomy. In *Engineering Management Conference, 2003. IEMC '03. Managing Technologically Driven Organizations : The Human Side of Innovation and Change*, pages 306–310, Canberra, Australia, 2003.

[XSB07] XSB research group. Welcome to the home of XSB ! http ://xsb.sourceforge.net, 2007.

[YS06] A. Yahiaoui and A.E.K. Sahraoui. A systems engineering environment for integrated building design. In *European Systems Engineering Conference 2006*, Edinburgh (GB), 18-20 Septembre 2006.

Liste des acronymes

A

AFIS	Association Française d'Ingénierie Système	
AFNOR	Association Française de NORmalisation	
ANSI	American National Standards Institute	
AUP	Agile Unified Process	

B

BPEL	Business Process Execution Language for Web Services	eq. à BPEL4WS
BPEL4WS	Business Process Execution Language for Web Services	eq. à BPEL
BPM	Business Process Management	
BPMI	Business Process Management Initiative	
BPML	Business Process Modeling Language	
BPMN	Business Process Model Notation	

C

CWM	Common Warehouse Metamodel	

D

DOD	Department Of Defense	
DoDAF	Department of Defense Architecture Framework	
DOMINO	DOMaINes et prOcessus méthodologique	
DSL	Domain-Specific Language	
DTD	Document Type Definition	

E

EPI	Équipement de Protection Individuel	
EIA	Electronic Industries Alliance	
EssUP	Enssential unified process	
EUP	Enterprise Unified Process	

I

IDM	Ingénierie Dirigée par les Modèles	eq. à MDE
IEEE	Institute of Electrical and Electronics Engineers	
INCOSE	INternational COuncil on Systems Engineering	
IS	Ingénierie Système	
ISAF	International SAiling Federation	

L

LATTIS	Laboratoire Toulousain de Technologie et d'Ingénierie des Systèmes	

M

MBSEM	Model-Based System Engineering Methodology	
MDA	Model Driven Architecture	
MDE	Model Driven Engineering	eq. à IDM
MDSD	Model Driven System Design	
MoDAF	UK Ministry of Defence Architectural Framework	
MOF	Meta-Object Facility	

N

NASA	National Aeronautics and Space Administration	
NAT	Nato Architecture Framework	
NCOSE	National COuncil on System Engineering	

O

OASIS	Organization for the Advancement of Structured Information Standards	
OCL	Object Constraint Language	
OMG	Object Management Group	
OOSEM	Object-Oriented System Engineering Method	

OSSAD	Office Support System Analysis and Design	

P

PBS	Product Breakdown Structure	

Q

QVT	Query/View/Transformation	

R

RUP	Rational Unified Process	
RUP SE	Rational Unified Process for System Engineering	

S

SDP	Structure de découpage du projet	voir WBS
SPEEDS	SPEculative and Exploratory Design in Systems engineering	
SPEM	Software Process Engineering Metamodel	
SysML	Systems Modeling Language	

U

UML	Unified Modeling Language	
UP	Unified Process	
UPDM	UML Profile for DoDAF/MoDAF	

V

V&V	Validation et Vérification	

W

WBS	Work Breakdown Structure	voir SDP
WfMC	Workflow Management Coalition	
W3C	World Wide Web Consortium	

X

XML	eXtensible Markup Language	
XPDL	XML Process Definition Language	

XUP	eXtreme Unified Process

2

2TUP	Two Tracks Unified Process

Liste des définitions

Glossaire

A

Activité [Wik07] (définition 1)

Ensemble distinct d'actions identifiées, organisé selon un processus logique, observable en tant que tel. Il peut désigner aussi une ou plusieurs tâches exécutées par un ou plusieurs employés à l'intérieur d'un processus.

B

Bloc de construction (définition 12)

Il est un élément dans la décomposition structurelle du système. Il conduit à une représentation du cadre conceptuel du système projeté utilisé pour organiser les exigences, le travail, et les autres informations associées à l'ingénierie système.

E

Équipement de Protection Individuel (EPI) (définition 15)

Dispositif de protection individuelle préservant une personne d'un risque menaçant sa sécurité. Cette définition est étendue aux dispositifs associés de façon solidaire ainsi qu'aux composants interchangeables.

I

Ingénierie Dirigée par les Modèles (IDM) (définition 13)

L'Ingénierie dirigée par les modèles renvoie à l'utilisation systématique de modèles comme artéfacts de conception primaires dans le cycle de vie. L'IDM est un domaine de l'informatique qui met à disposition des outils, des concepts et des langages pour créer et transformer ces modèles.

Ingénierie Système (IS) [Tho93] (définition 2)

L'ingénierie système est une approche et des moyens interdisciplinaires permettant la réalisation et le déploiement de systèmes réussis. Elle peut être vue comme l'application de techniques d'ingénierie à l'ingénierie des systèmes, aussi bien que comme l'application d'une approche systématisée aux efforts d'ingénierie.

Systems Engineering is an interdisciplinary approach and means for enabling the realization and deployment of successful systems. It can be viewed as the application of engineering techniques to the engineering of systems, as well as the application of a systems approach to engineering efforts.

Ingénierie Système guidée par les Modèles (définition 8)

L'Ingénierie système guidée par les modèles est une démarche qui propose l'extension des principes de l'IDM à l'ingénierie des systèmes. Il s'agit de considérer l'ensemble des parties du systèmes comme des modèles et non plus uniquement ses composants logiciels. Dans ce contexte, les opérations de transformation de modèles sont guidées par les processus de l'ingénierie système.

M

Méthode [Est07] (définition 5)

Une méthode est attachée aux techniques de réalisation d'une tâche. Elle définit le « comment faire » de chaque tâche (dans ce contexte méthode, technique, pratique et procédure sont souvent interchangeables). A tous niveaux, les tâches de processus sont réalisées en utilisant des méthodes. Cependant, chaque méthode est aussi, elle–même, un processus avec sa séquence de tâches à réaliser selon des méthodes particulières. En d'autres mots, le « comment » d'un niveau d'abstraction devient le « quoi » du niveau inférieur.

Méthodologie [Est07] (définition 7)

Une méthodologie est définie comme une collection de processus, de méthodes et d'outils reliés. Une méthodologie est essentiellement une « recette » et peut être comprise comme l'application de processus, méthodes et outils relatifs à une classe de problèmes qui ont quelque chose en commun.

Métier (définition 14)

L'ensemble des spécificités techniques ou non liées à la réalisation d'un produit ou d'un service.

N

Norme [ISO04] (définition 9)

Document, établi par consensus et approuvé par un organisme reconnu, qui fournit, pour des usages communs et répétés, des règles, des lignes directrices ou des caractéristiques, pour des activités ou leurs résultats, garantissant un niveau d'ordre optimal dans un contexte donné.

O

Outil [Est07] (définition 6)

Un outil est attaché à une méthode particulière, pour augmenter l'efficacité de réalisation de la tâche. Le rôle d'un outil est de faciliter le « comment ».

P

Partie intéressées (ou partie prenante) [Mer01] (d'après[Fre84])

Une partie prenante est un individu ou un groupe d'individus qui peut affecter ou être affecté par la réalisation des objectifs organisationnels.

Processus [Est07] (définition 4)

Un processus est une séquence logique de tâches réalisées pour atteindre un objectif particulier. Un processus définit ce qui est à faire sans préciser le « comment faire ». La structure d'un processus peut se présenter sous différents niveaux d'agrégation pour permettre les analyses et répondre aux différents besoins d'aide à la décision.

Procédure d'entreprise (ou processus d'entreprise) [Jou07] (définition 10)

Une procédure d'entreprise est une procédure qui systématise l'organisation et la politique d'une entreprise dans le but d'atteindre certains des objectifs de cette entreprise.

S

Système [SC95] (définition 3)

Un système est un ensemble de composants interdépendants qui interagissent entre eux de façon organisée vers un but commun. Les composants d'un système peuvent être divers et se composer de personnes, d'organismes, de procédures, de logiciels, d'équipements ou d'installations.

A system is a set of interrelated components which interact with one another in an organized fashion toward a common purpose. The components of a system may be quite diverse, consisting of persons, organisations, procedures, software, equipement or facilities.

Système (selon EIA-632) (définition 11)

Agrégation de produits finaux et de produits contributeurs dans un but donné.

Table des figures

Liste des tableaux

Annexes

Annexe 1 : Correspondances des processus normalisés d’IS avec le processus SIMILAR

Processus SIMILAR		Processus de conception système de l'EIA-632
S	Poser le problème	Processus de définition des exigences (4.3.1)
I	Étudier les alternatives	Processus de définition de la solution (4.3.2) Exigences 17 a & b, 18 c & d, 22 et 23
M	Modéliser le système	Représentations logiques de la solution (exigence 17)
I	Intégrer	Identifier et définir les interfaces (exigence 17b2)
L	Lancer le système	Réalisation du produit (4.4)
A	Évaluer la performance	processus de vérification du produit (4.5.3)
R	Re–évaluer	De nombreuses tâches itératives (4.3 note 5)

TAB. 6: *Correspondances entre processus SIMILAR et processus de conception système de l'EIA-632 [BG98].*

Processus SIMILAR		Processus d'ingénierie système de l'IEEE 1220
S	Poser le problème	Analyse des exigences (6.1)
I	Étudier les alternatives	Synthèse (6.5) Analyse du système (6.7)
M	Modéliser le système	Analyse fonctionnelle (6.3)
I	Intégrer	
L	Lancer le système	Synthèse (6.5)
A	Évaluer la performance	Contrôle (6.8)
R	Re–évaluer	Validation des exigences de base (6.2) Vérification fonctionnelle (6.4) Vérification matérielle (6.6)

TAB. 7: *Correspondances entre processus SIMILAR et processus d'ingénierie système de l'IEEE 1220 [BG98].*

Annexe 2 : Traductions françaises des exigences et processus de l'EIA-632

EXIGENCES DU PROCESSUS DE FOURNITURE	12 – Gestion des résultats	22 – Analyses d'efficacité
1 – Fourniture du produit	13 – Dissémination de l'information	23 – Analyses des compromis
		24 – Analyses des risques
EXIGENCES DU PROCESSUS D'ACQUISITION	EXIGENCES DU PROCESSUS DE DÉFINITION DES EXIGENCES	EXIGENCES DU PROCESSUS DE VALIDATION DES EXIGENCES
2 – Acquisition du produit	14 – Exigences de l'acquéreur	25 – Validation de l'expression des exigences
3 – Performances des fournisseurs	15 – Exigences des autres parties intéressées	26 – Validation des exigences de l'acquéreur
	16 – Exigences techniques du système	27 – Validation des exigences d'une autre partie intéressée
EXIGENCES DU PROCESSUS DE PLANIFICATION	EXIGENCES DU PROCESSUS DE DÉFINITION DE LA SOLUTION	28 – Validation des exigences techniques du système
4 – Stratégie de mise en œuvre du processus	17 – Représentations logiques de la solution	29 – Validations des représentations logiques de la solution
5 – Définition de l'effort technique	18 – Représentations physiques de la solution	
6 – Ordonnancement et organisation	19 – Exigences spécifiées	EXIGENCES DU PROCESSUS DE VÉRIFICATION DU SYSTÈME
7 – Plans techniques	EXIGENCES DU PROCESSUS D'IMPLÉMENTATION	30 – Vérification de la solution de conception
8 – Directives de travail	20 – Implémentation	31 – Vérification du produit final
EXIGENCES DU PROCESSUS D'ÉVALUATION	EXIGENCES DU PROCESSUS DE TRANSITION VERS L'UTILISATION	32 – Disponibilité du produit contributeur
9 – Progrès relatifs aux plans et aux calendriers	21 – Transition vers l'utilisation	EXIGENCES DU PROCESSUS DE VALIDATION DES PRODUITS FINAUX
10 – Progrès relatifs aux exigences	EXIGENCES DU PROCESSUS D'ANALYSES DES SYSTÈMES	33 – Validation des produits finaux
11 – Revues techniques		
EXIGENCES DU PROCESSUS DE CONTRÔLE		

TAB. 8: *Liste des exigences liées aux processus de l'EIA-632 (version française)*

Annexe 3 : Diagrammes d’activité des phases du cycle de vie de l’EIA-632

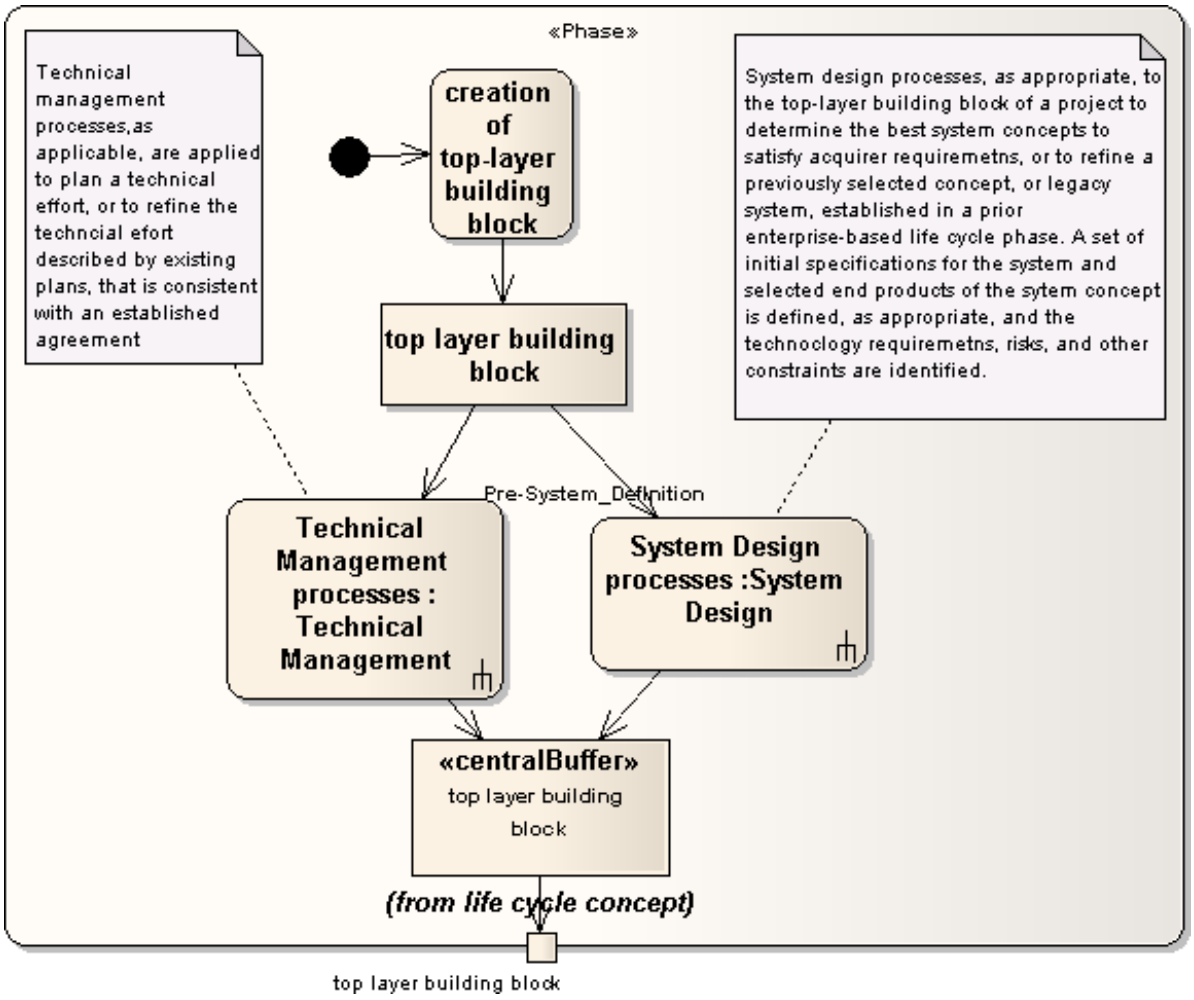


FIG. 1: *Définition du pré–système.*

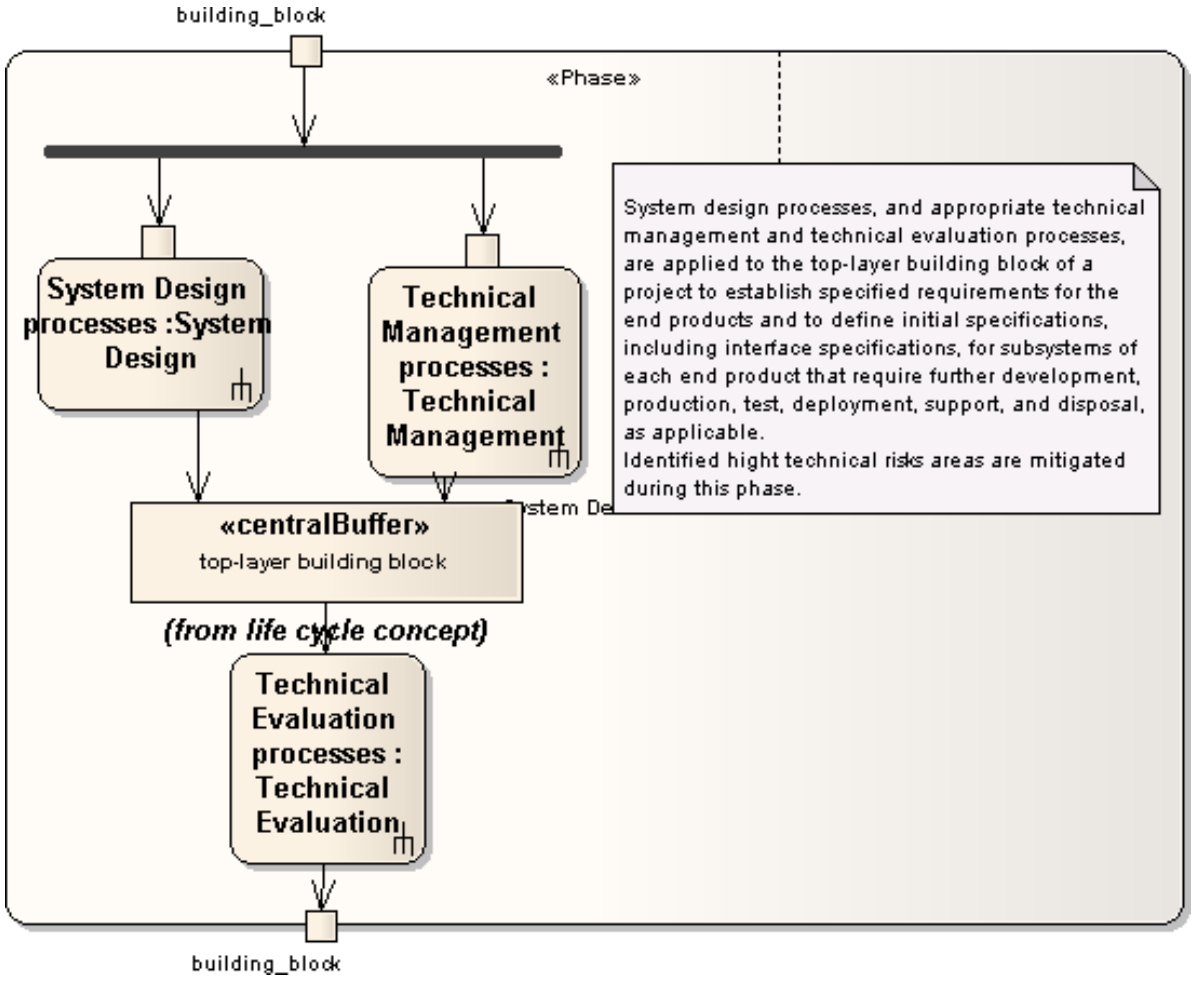


FIG. 2: *Définition du système.*

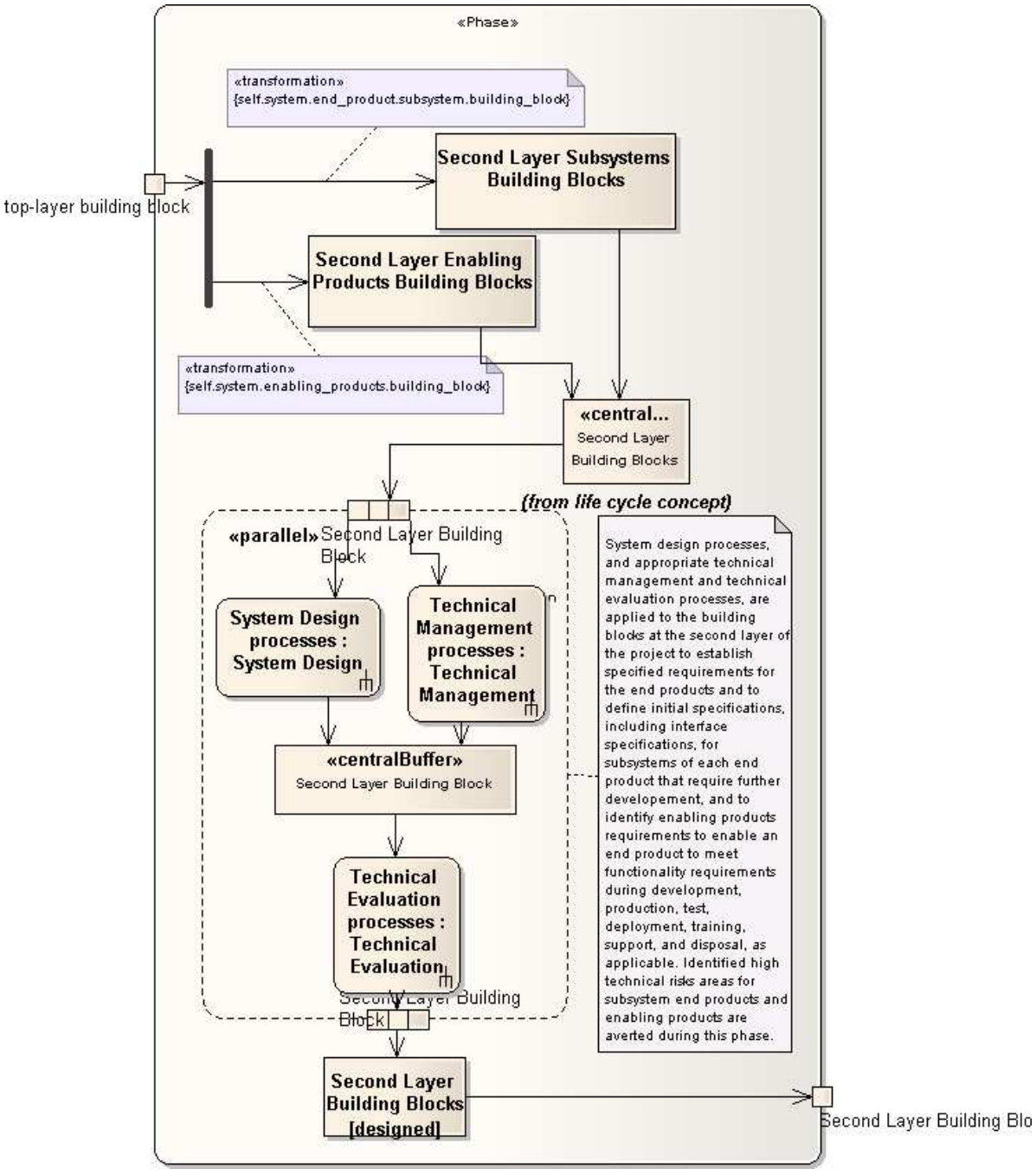


FIG. 3: *Conception des sous–systèmes.*

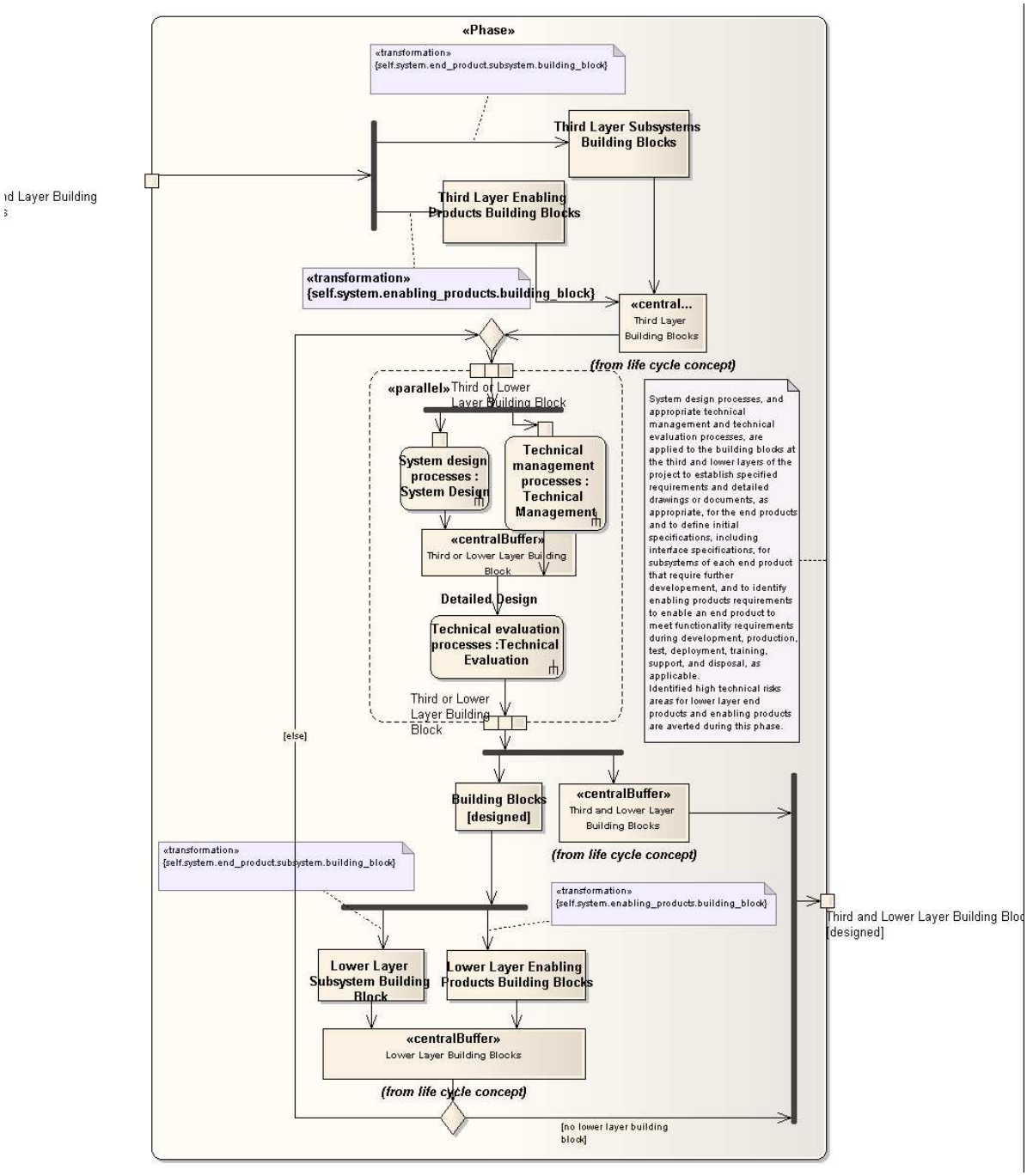


FIG. 4: *Conception détaillée.*

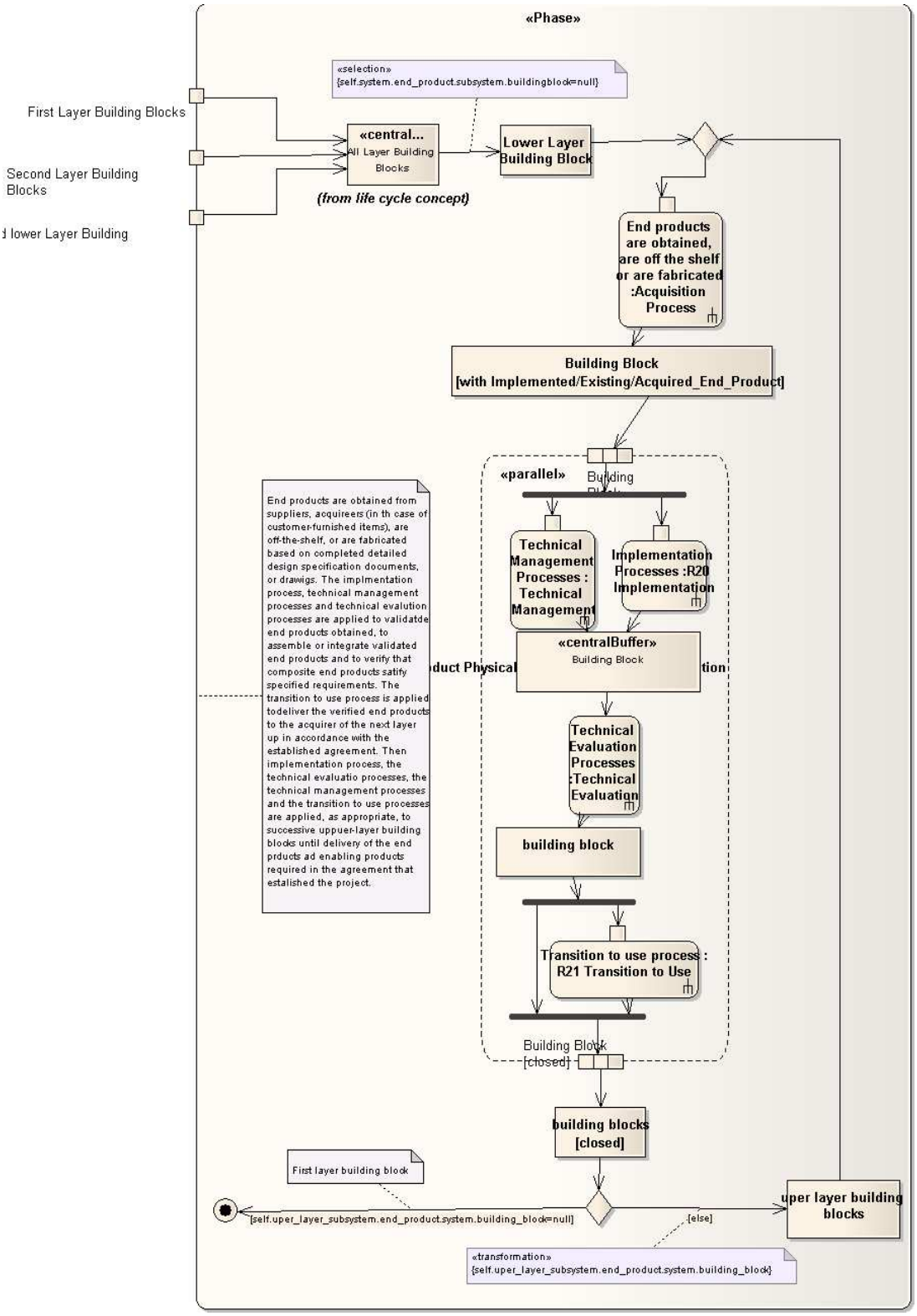


FIG. 5: *Intégration, test et évaluation des produits finaux.*

Annexe 4 : Détails des figures 2.7(a), 2.13, 2.14 et 2.15

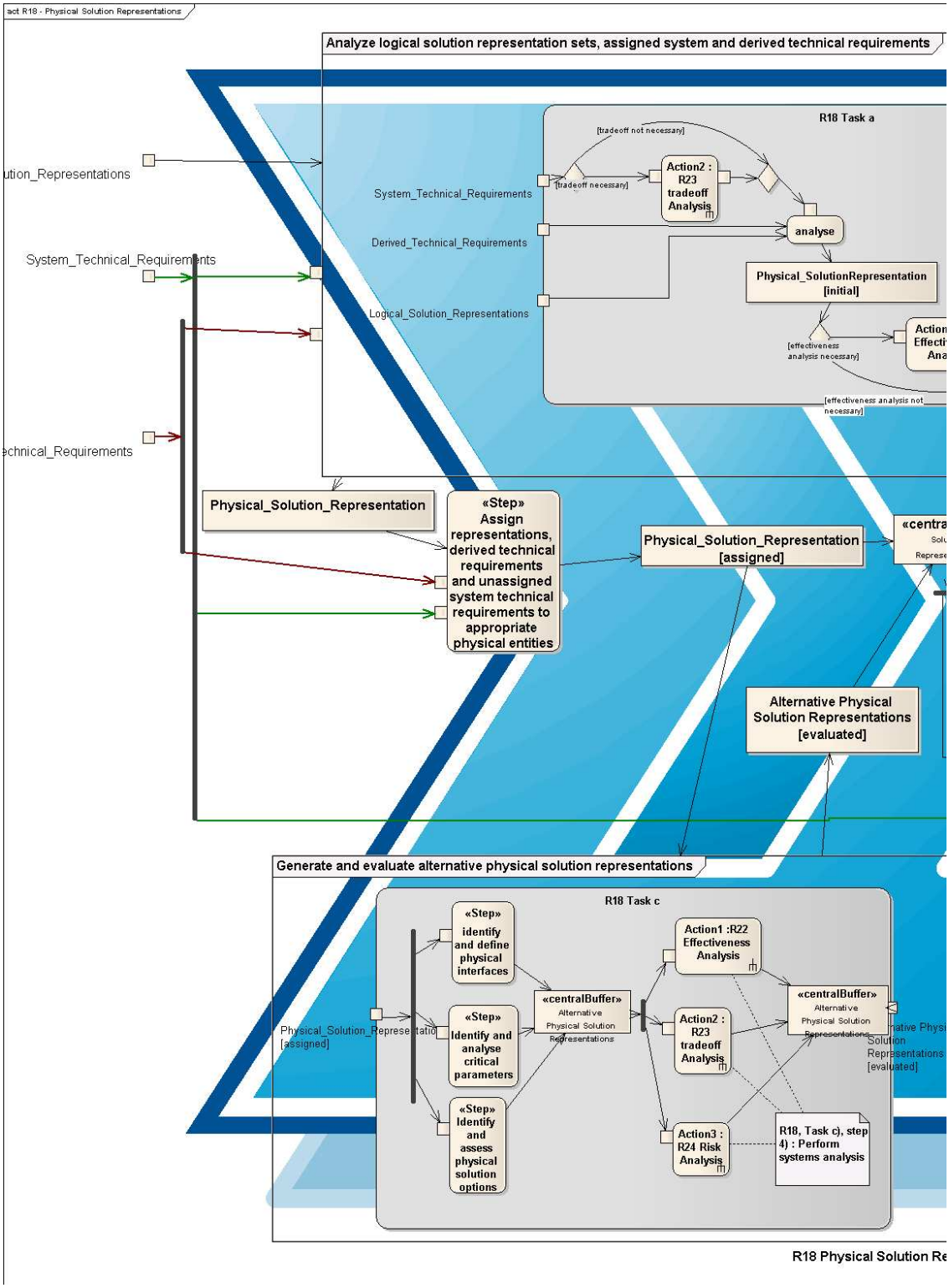


FIG. 6: *Vue de détail du diagramme d'activité de l'exigence de représentation physique de la solution (1/2).*

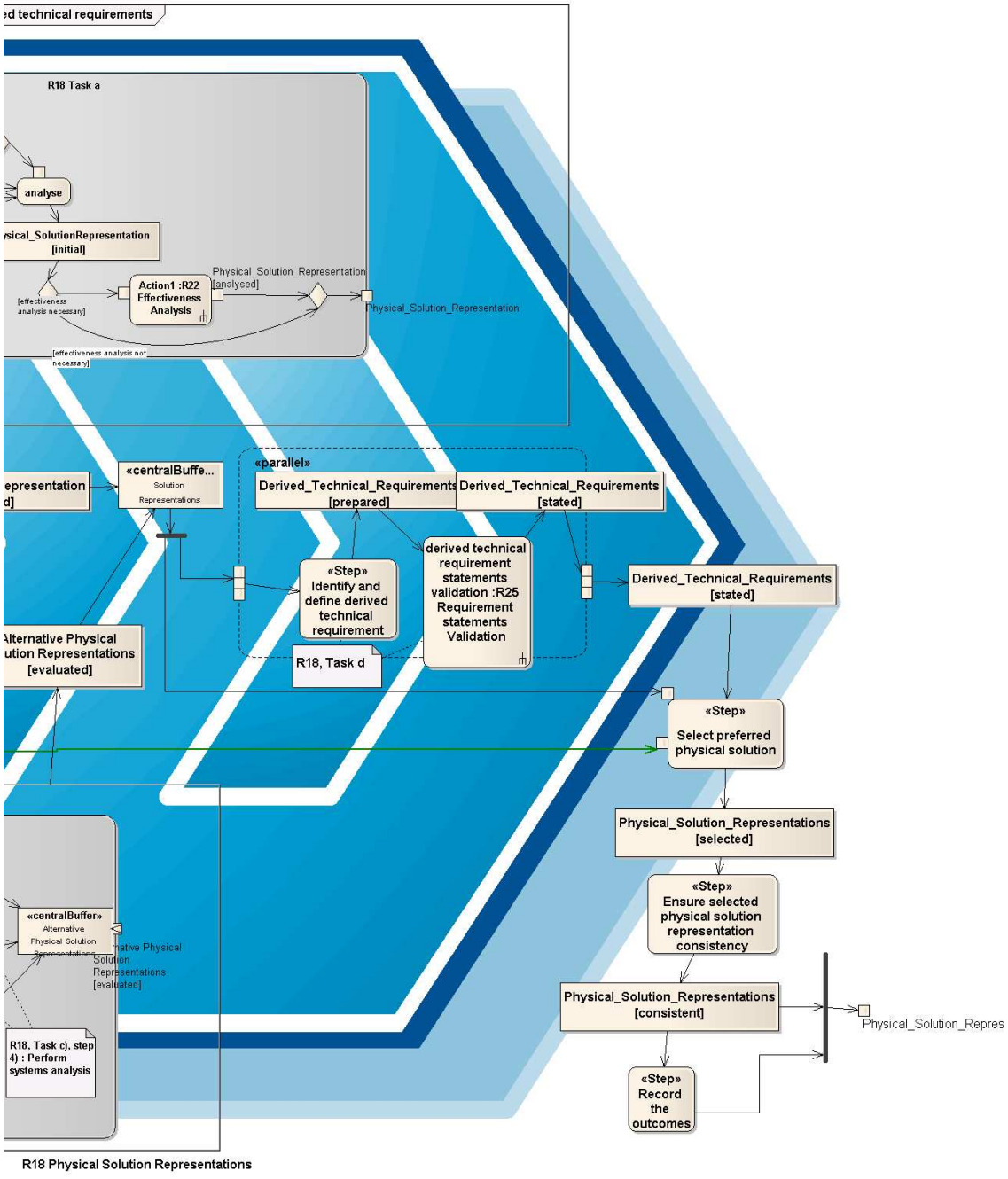


FIG. 7: *Vue de détail du diagramme d'activité de l'exigence de représentation physique de la solution (2/2).*

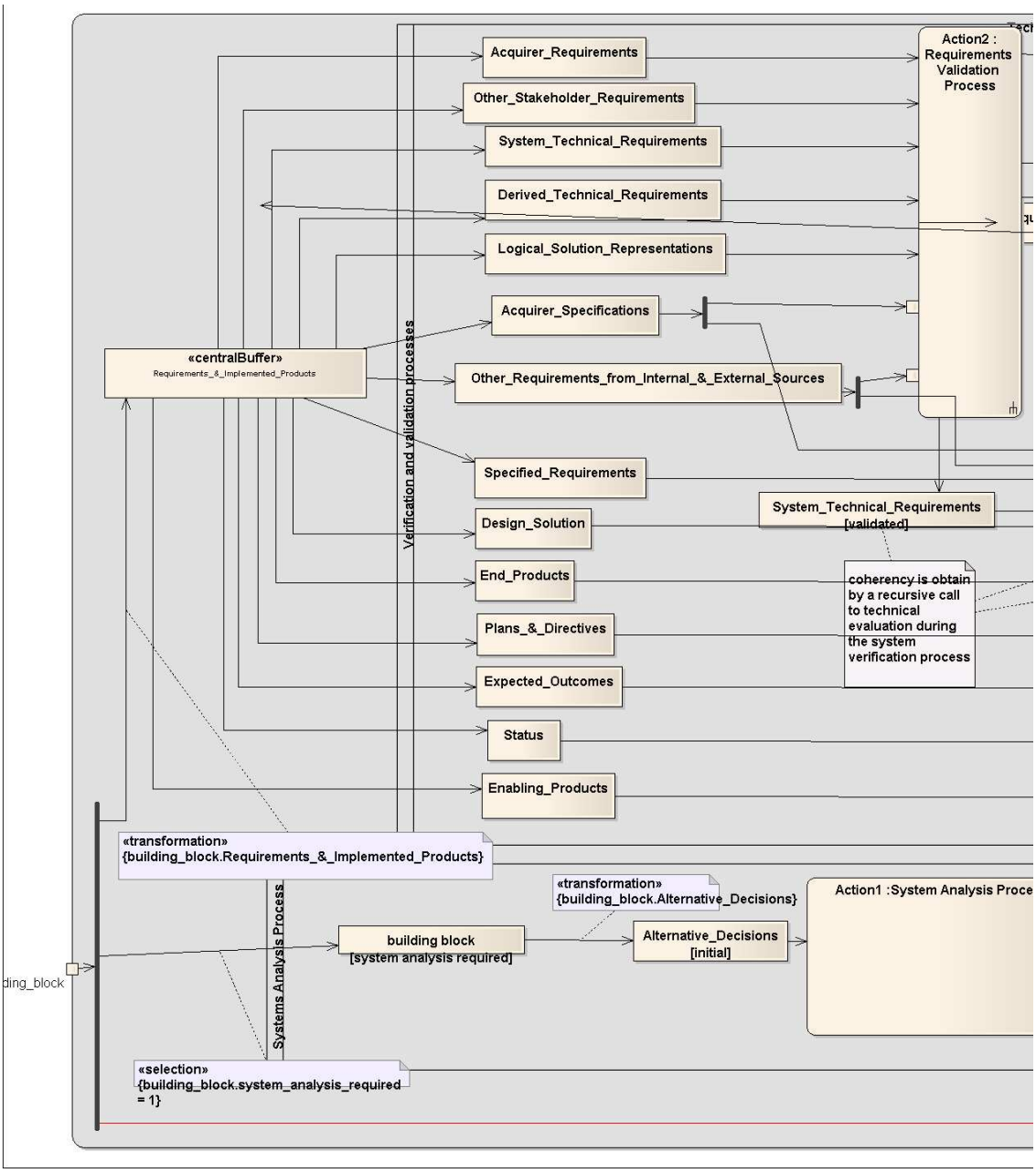


FIG. 8: *Vue de détail du groupe d'évaluation technique (1/2).*

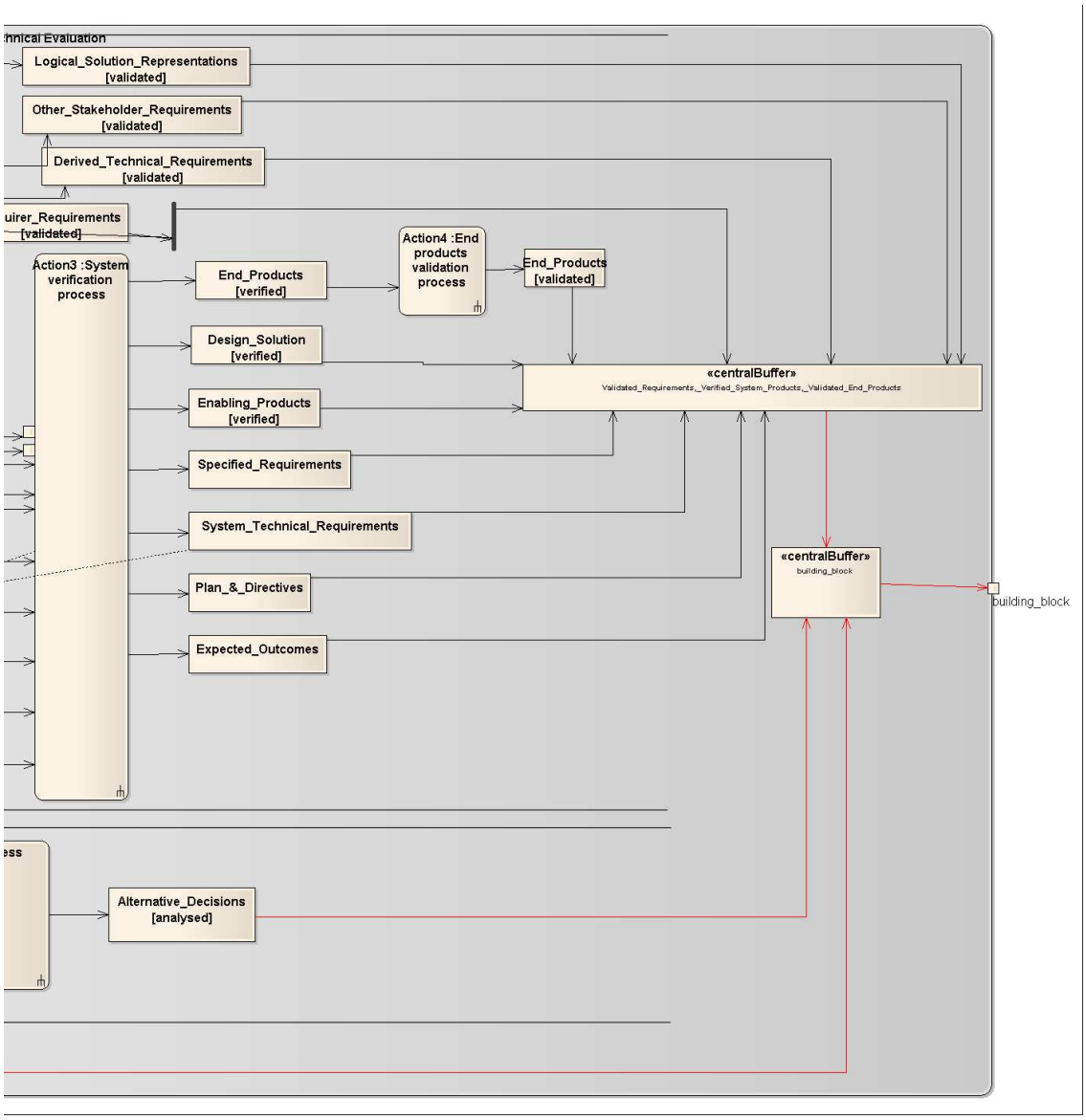


FIG. 9: *Vue de détail du groupe d'évaluation technique (2/2).*

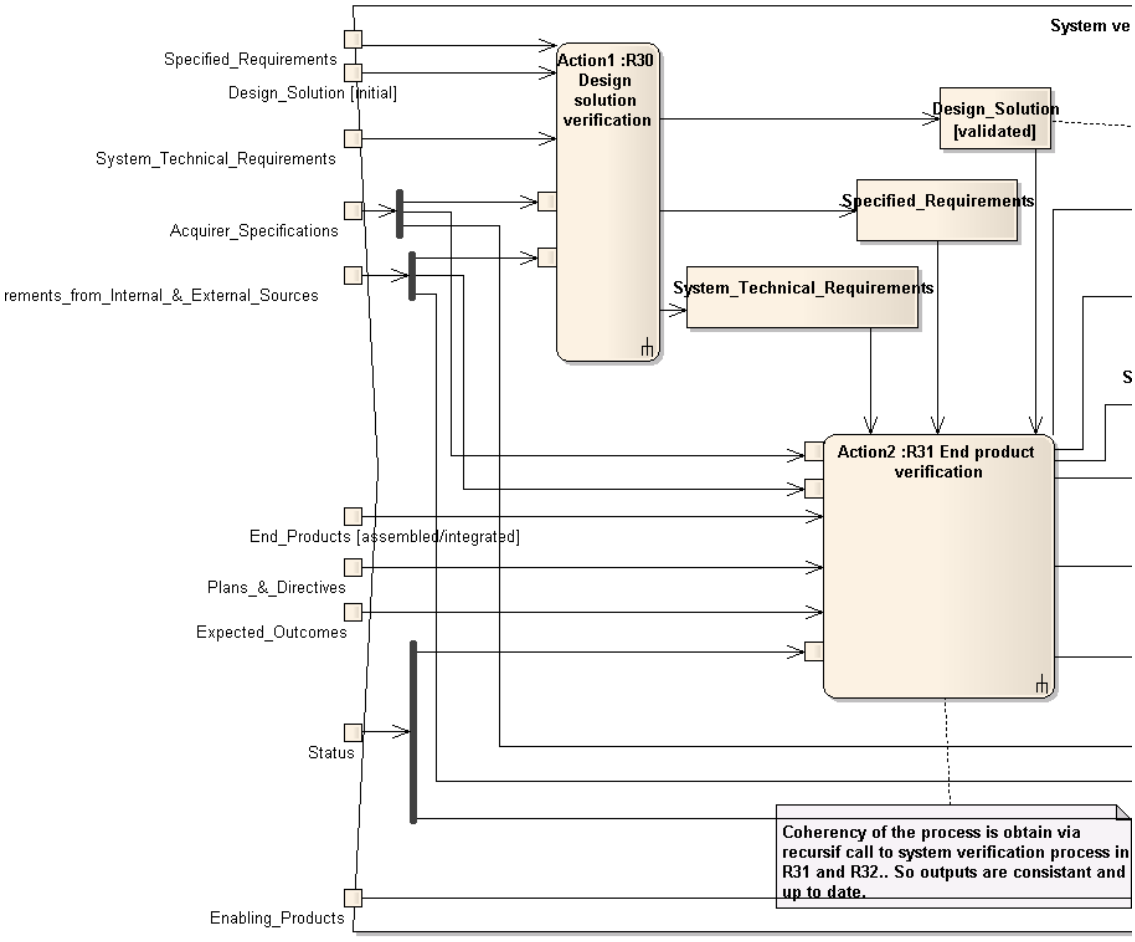


FIG. 10: *Vue de détail du processus de vérification du système (1/2).*

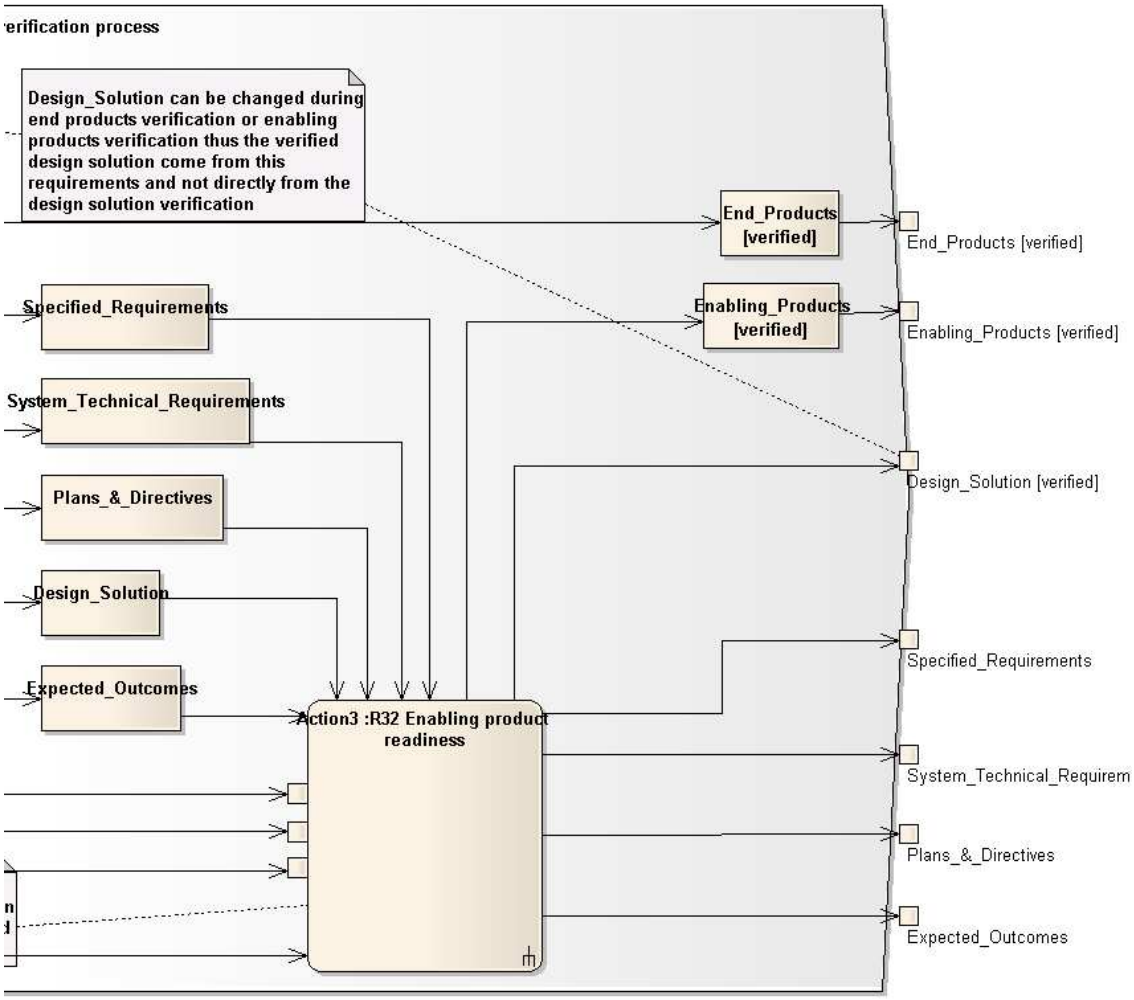


FIG. 11: *Vue de détail du processus de vérification du système (2/2).*

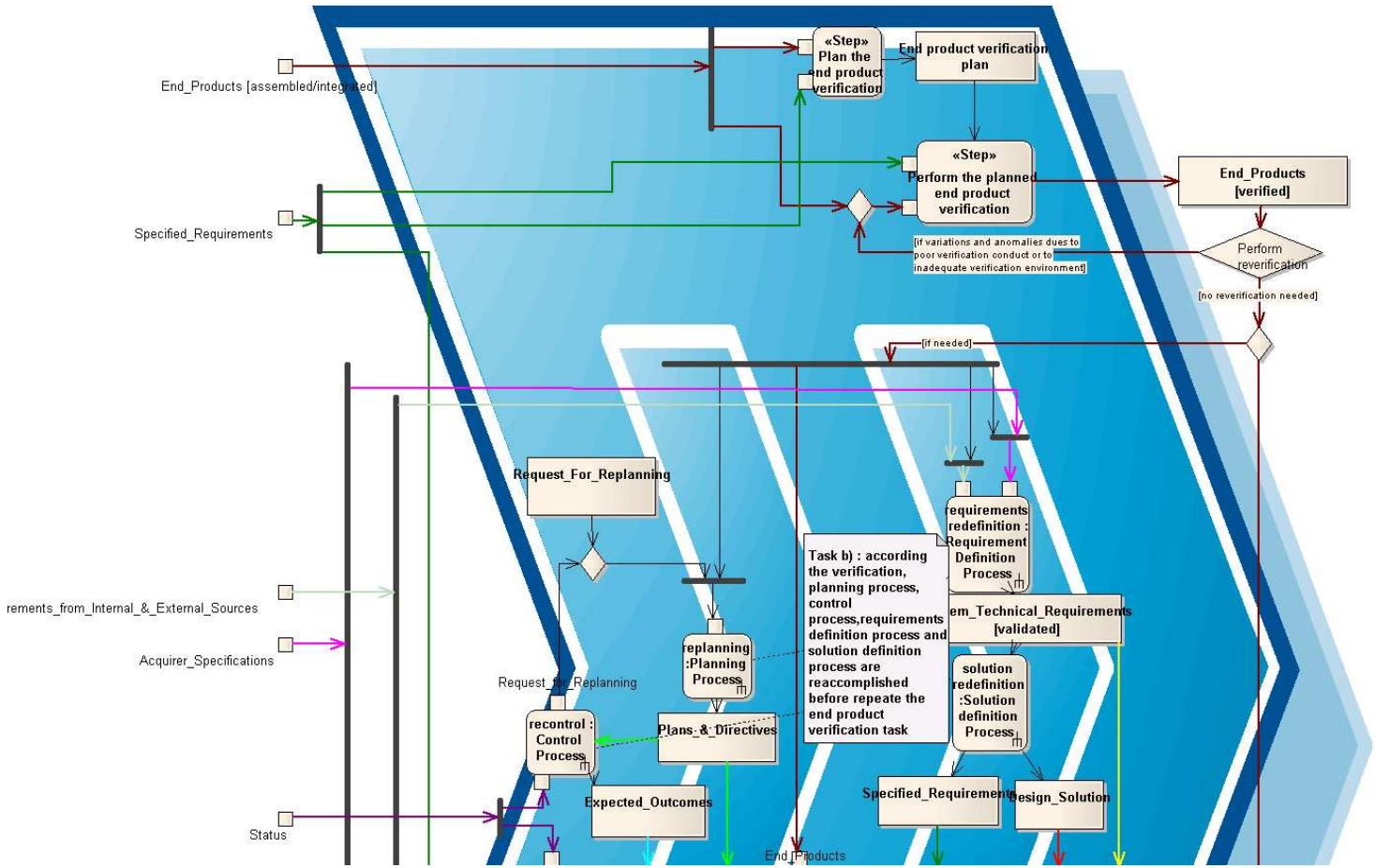


FIG. 12: *Vue de détail de l'exigence 31 – vérification du produit final. (1/2).*

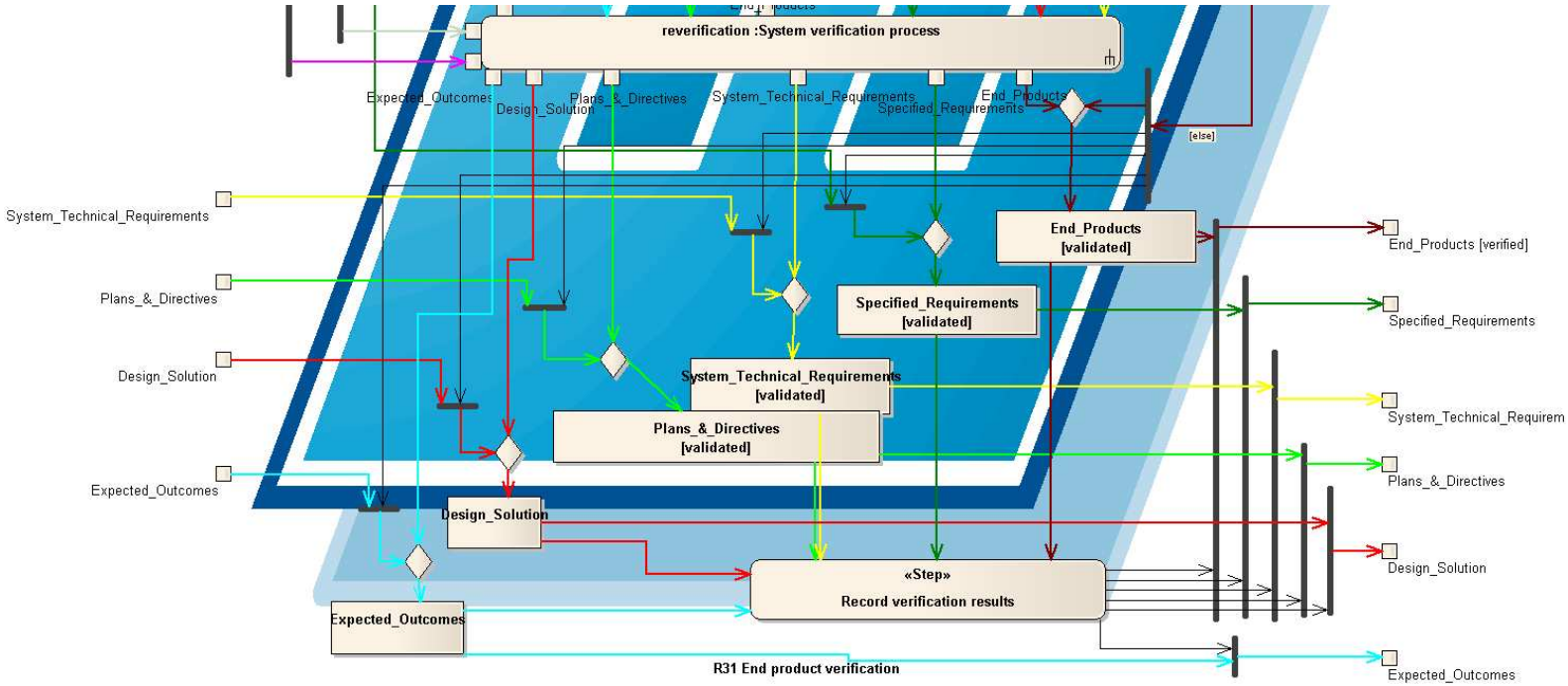


FIG. 13: *Vue de détail de l'exigence 31 – vérification du produit final. (2/2).*

Annexe 5 : Exigences et plan de vérification pour EPI

Normes caractéristiques	EN 1891 Type A	EN 1891 Type B
Caractéristiques	Cordes tressée gainée à faible coefficient d'allongement	Cordes tressée gainée à faible coefficient d'allongement
Diamètre	$\oslash$ 8.5 à 16 mm	$\oslash$ 8.5 à 16 mm
Nouabilité	$k < 1.2$	$k < 1.2$
Glissement gaine	$< 20 + 10\,(\oslash - 9mm)$ pour $\oslash < 12$	$< 20 + 10\,(\oslash - 9mm)$ pour $\oslash < 12$
	$< 20 + 5\,(\oslash - 12mm)$ pour $\oslash > 12$	$< 20 + 5\,(\oslash - 12mm)$ pour $\oslash > 12$
Performances dynamiques	$M = 100kg$ 2 m corde, 0.60 m chute $F < 6kN$, 5 chutes sans rupture	$M = 80kg$ 2 m corde, 0.60 m chute $F < 6kN$, 5 chutes sans rupture
Résistance statique	$Fr > 22kN$, 15 kN avec nœud en 8	$Fr > 18kN$, 12 kN avec nœud en 8
Allongement de 50 à 150 kg	$< 5\%$	$< 5\%$

TAB. 9: *Exemple d'exigences fixées par les normes pour les EPI de type corde semi–statiques [PET].*

Normes caractéristiques	EN 341 classe A
Test statique	
Traction	$12kN/3mn$
Force de retenue pour la charge d'utilisation maximale	$F < 120N$
Essai énergie de descente	100 descentes de 100 m avec une masse de 75 kg. Évaluation de température du descendeur.
test dynamique	chute de 60 cm, 4 m de corde au dessus du descendeur, charge d'utilisation maxi ($> 100kg$)
Test de fonctionnement	faire la hauteur de descente maximale avec les charges maxi et mini. On doit pouvoir maintenir la vitesse de descente entre 0.5 et 2 m/s.

TAB. 10: *Exemple de plan de vérification fixé par des normes pour EPI de type descendeur.*

Annexe 6 : Documentation utilisateur de l’outil de validation

UML2LP

Documentation utilisateur

Auteur	Version	Modifs
Jérémy Maës	Initiale	

Institut National des Sciences Appliquées de Toulouse
Année 2007

Sommaire

I/ Introduction

UML2LP est un programme écrit en JAVA permettant de transformer un modèle UML associé à un métamodèle en règles et faits de programmation logique. Il dispose également d'une interface avec XSB Prolog pour permettre à l'utilisateur de rapidement exploiter les fichiers Prolog généré par la transformation. Cet outil fut développé par Hugues Malgouyres puis amélioré par Jérémy Maës tous deux de l'Institut National des Sciences Appliquées de Toulouse.

II/ Installation

Ce programme est prévu pour être installé sous Unix, tout fonctionnement sous un autre type d'OS n'est aucunement garanti. Afin de fonctionner, le programme nécessite l'installation de JAVA ainsi que l'installation de XSB Prolog. Ces installations sont décrites plus précisément ci-dessous.

1) Installation sous Unix

i. Installation de Java

Pour fonctionner, le programme UML2LP nécessite la présence de J2SE version 1.4 ou supérieure. Téléchargez J2SE sur le site http://java.sun.com/j2se/1.4.2/download.html et installez-le sur votre machine et notez le répertoire d'installation que nous appellerons *MY_JAVA_DIR* (ce chemin sera nécessaire pour la suite de l'installation).

Si vous souhaitez vérifier que J2SE est bien installé sur votre machine, ouvrez un shell et tapez la commande suivante :

```
MY_JAVA_DIR/bin/java -version
```

Vous devriez obtenir quelque chose comme cela :

```
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2-b28)
Java HotSpot(TM) Client VM (build 1.4.2-b28, mixed mode)
```

Vérifiez que vous avez bien la version 1.4 ou supérieure.

Si vous n'obtenez pas de résultat lorsque vous tapez la commande précédemment indiquée, cela signifie que J2SE ne doit pas être correctement installé. Recommencez l'installation de J2SE avant de poursuivre.

ii. Installation de XSB

L'étape suivante consiste à installer XSB Prolog 2.7.1 (ou supérieur). Téléchargez le fichier tar depuis http://xsb.sourceforge.net puis extrayez l'arborescence entière d'*XSB* dans un dossier que nous appellerons *MY_XSB_DIR*.

Suivez les étapes indiquées dans le document XSB INSTALL, toutefois vous devez requérir le support d'interprolog ainsi que la génération de la librairie dynamic XSB :

```
cd MY_XSB_DIR/build
./configure --with-interprolog --with-includes='MY_JAVA_DIR/include
MY_JAVA_DIR/include/linux'
./makexsb
./makexsb dynmodule
```

Assurez-vous que tout s'exécute sans erreur. Vous pouvez vérifier que XSB Prolog fonctionne correctement en tapant dans un shell la commande :

```
MY_XSB_DIR/config/i586-pc-linux-gnu/bin/xsb
```

Vous devriez obtenir quelque chose comme cela :

```
XSB Version 2.6 (Duff) of June 24, 2003
[i586-pc-linux-gnu; mode: debug; engine: slg-wam; gc: indirection; scheduling: local]
| ?-
```

Si vous n'obtenez pas de résultat lorsque vous tapez la commande précédemment indiquée, cela signifie que XSB Prolog ne doit pas être correctement installé. Recommencez l'installation de XSB Prolog avant de poursuivre.

iii. Installation de UML2LP

Extrayez l'arborescence entière d'*UML2LP.tar.gz* dans un dossier que nous appellerons *MY_UML2LP_DIR*.

Regardez ensuite dans le dossier *MY_UML2LP_DIR/unixScripts/*. Editez le fichier *unixVariables.sh* et modifiez la valeur de la variable *XSB_BIN_DIRECTORY* pour qu'elle contienne le chemin du dossier contenant l'exécutable XSB (*MY_XSB_DIR/config/i586-pc-linux-gnu/bin/)*.

Assurez-vous que les scripts *MY_UML2LP_DIR/unixScripts/unixVariables.sh* et *MY_UML2LP_DIR/*UML2LP.sh soient exécutables (chmod a+x *.sh dans *MY_UML2LP_DIR*).

Vous pouvez vérifier que UML2LP se lance correctement en exécutant le script *MY_UML2CLP_DIR/UML2LP.sh*. Si tout se passe bien, vous devriez voir apparaître l'interface graphique du logiciel (une fenêtre portant le titre *UML2LP*). Sinon veuillez vérifier que vous avez correctement respecté la procédure d'installation et au besoin recommencez l'installation.

Si vous modifiez votre installation de XSB Prolog, assurez-vous de spécifier correctement le nouveau chemin contenant l'exécutable XSB dans le fichier *MY_UML2LP_DIR/unixScripts/unixVariables.sh*. Si vous modifiez votre installation de JAVA, assurez-vous simplement que la version 1.4 ou supérieure est toujours installée (testez avec la commande MY_JAVA_DIR/bin/java -version).

2) *Installation sous Windows*

Le logiciel n'a pas été prévu pour être installé sous Windows.

III/ Utilisation

1) *Lancer UML2LP*

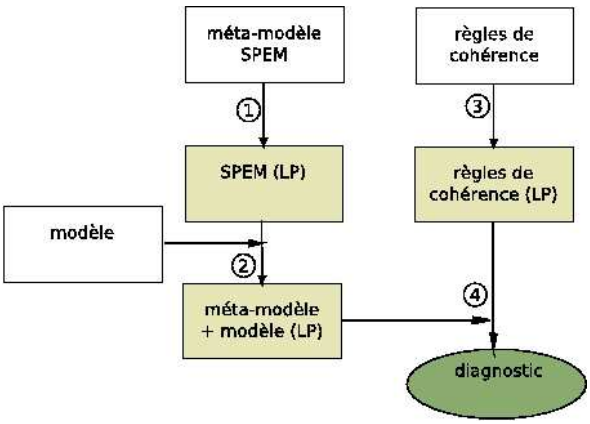
Pour lancer UML2LP, lancez le script *UML2LP.sh* situé dans le dossier *MY_UML2LP_DIR* (dossier où a été extrait UML2LP). Une fenêtre devrait apparaître si tout fonctionne comme il faut. Si ce n'est pas la cas, il faut vérifier que l'installation fut correctement réalisée.

2) *Utiliser UML2LP*

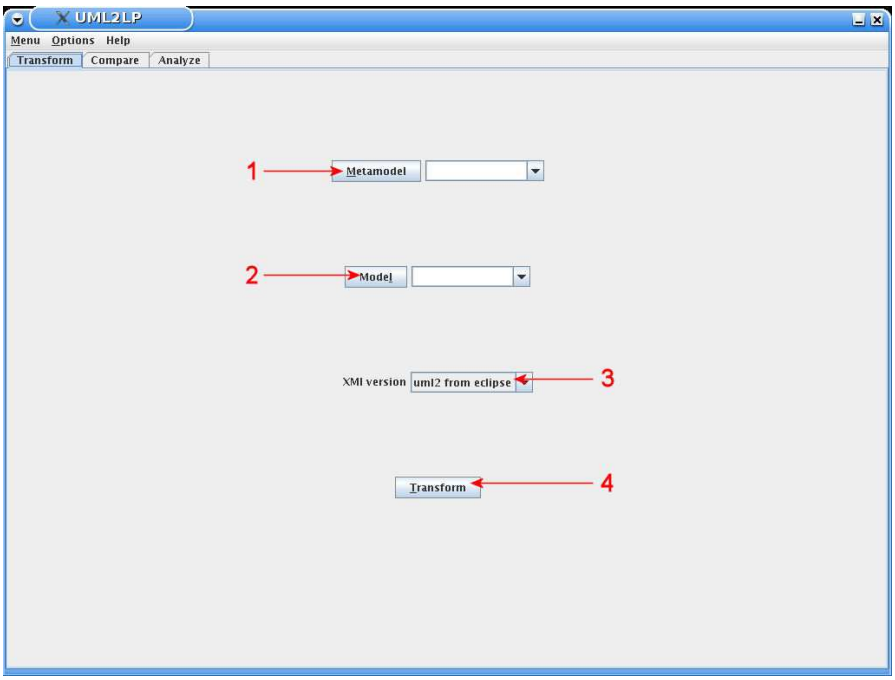
Le programme UML2LP se découpe en trois onglets : “Transform”, “Compare” et “Analyze”. L'onglet “Transform” vous permet de choisir un modèle accompagné de son métamodèle et de le transformer en règles et faits Prolog. L'onglet “Compare” réalise la même opération que “Transform” mais avec deux modèles. L'onglet “Analyze” est une interface avec un moteur XSB Prolog pour permettre à l'utilisateur de rapidement exploiter les résultats de ses transformations. Toutefois rien ne vous empêche d'exploiter les fichiers de sortie dans n'importe quel moteur XSB Prolog.

i. *Transformation vers LP*

Dans l'onglet “Transform” vous avez la possibilité de transformer un modèle défini par un métamodèle en règles et faits Prolog. Voici le schéma de principe :



Voici une image de l'onglet “Transform” du logiciel UML2LP :



1) Le bouton “Metamodel” permet d'ouvrir un sélecteur de fichier pour choisir le métamodèle correspondant au modèle que vous souhaitez transformer. Vous pouvez également directement taper le chemin vers le fichier contenant le métamodèle que vous voulez ou vous pouvez sélectionner un fichier récemment ouvert dans la liste déroulante.

2) Le bouton “Model” permet d'ouvrir un sélecteur de fichier pour choisir le modèle que vous souhaitez transformer. Vous pouvez également directement taper le chemin vers le fichier contenant le modèle que vous voulez ou vous pouvez sélectionner un fichier récemment ouvert dans la liste déroulante.

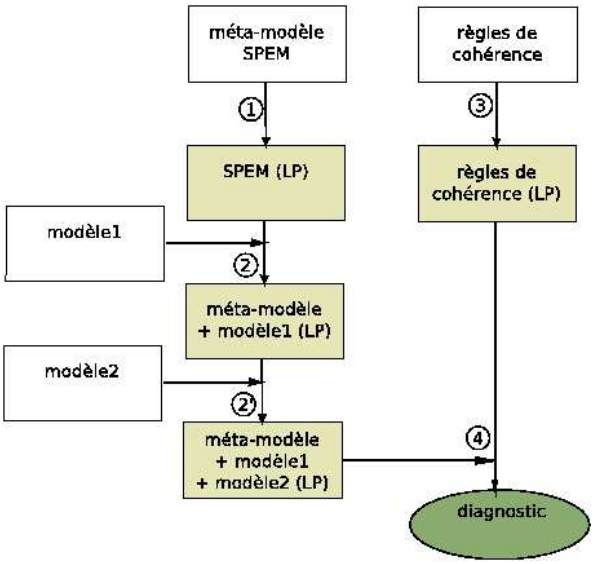
3) Vous devez choisir dans cette liste déroulante la version xmi de votre fichier modèle. Il y a 3 choix possibles : “uml2 from eclipse”, qui correspond aux modèles directement exportés par le logiciel eclipse, “rpy” pour rhapsody, qui correspond aux modèles générés par rhapsody, et “Omondo” pour le plug-in eclipse éponyme.

4) Lorsque vous cliquez sur le bouton “Transform” un sélecteur de fichier s'ouvre. Choisissez dans quel fichier vous souhaitez enregistrer le résultat de la transformation puis appuyez sur le bouton “Save” et la transformation proprement dite sera lancée. A la fin du processus, selon si ce dernier a réussi ou échoué, une petit boite de dialogue apparaît pour informer du résultat. En cas d'échec veillez à bien vérifier les chemins de vos fichiers et la version xmi.

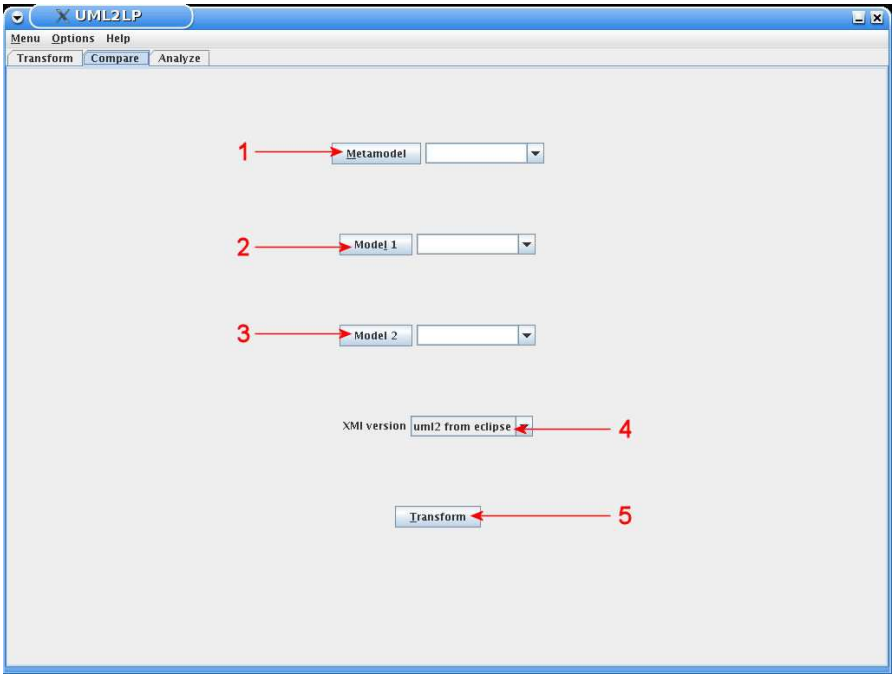
Durant le processus de transformation, vous ne pouvez plus cliquer sur l'interface graphique, il vous faut attendre la fin du processus. Il peut également arriver que le logiciel ferme brutalement durant ce processus. Dans ce cas recommencez l'opération. Si d'autres crash surviennent, cela signifie que le logiciel est incapable de traiter les fichiers que vous avez fournis en entrée.

ii. Comparaison de deux modèles

Dans l'onglet “Compare” vous avez la possibilité de transformer deux modèles définis par un métamodèle en règles et faits Prolog. Les deux modèles sont pris pour etre fusionnés en un seul. Concrètement le contenu de la racine du second modèle est inséré dans la racine du premier modèle. Voici le schéma de principe :



Voici une image de l'onglet “Compare” du logiciel UML2LP :



1) Le bouton “Metamodel” permet d'ouvrir un sélecteur de fichier pour choisir le métamodèle correspondant au modèle que vous souhaitez transformer. Vous pouvez également directement taper le chemin vers le fichier contenant le métamodèle que vous voulez ou vous pouvez sélectionner un fichier récemment ouvert dans la liste déroulante.

2) Le bouton “Model 1” permet d'ouvrir un sélecteur de fichier pour choisir le premier modèle que vous souhaitez transformer. Vous pouvez également directement taper le chemin vers le fichier contenant le premier modèle que vous voulez ou vous pouvez sélectionner un fichier récemment ouvert dans la liste déroulante.

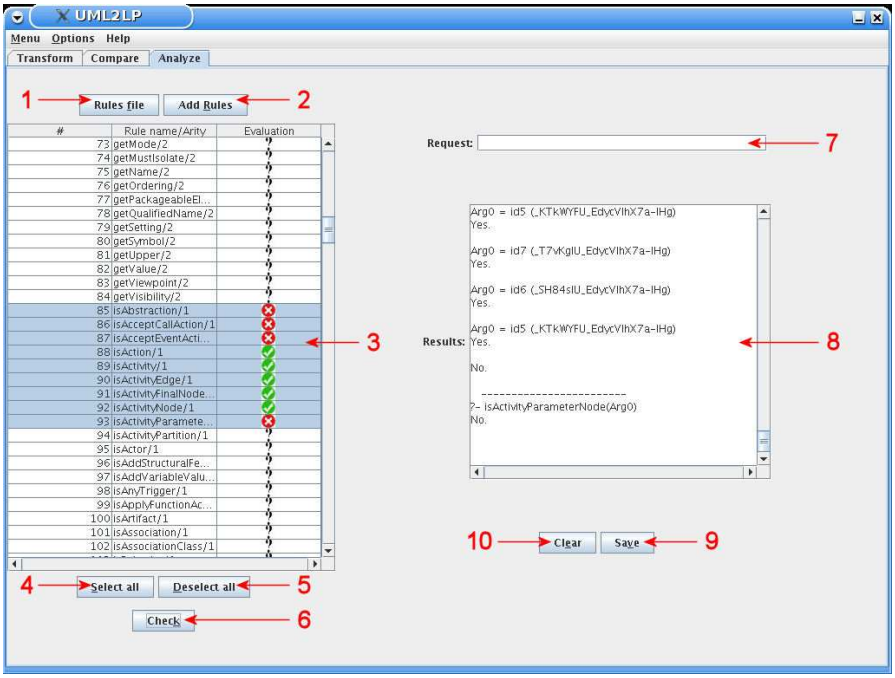
3) Le bouton “Model 2” permet d'ouvrir un sélecteur de fichier pour choisir le deuxième modèle que vous souhaitez transformer. Vous pouvez également directement taper le chemin vers le fichier contenant le deuxième modèle que vous voulez ou vous pouvez sélectionner un fichier récemment ouvert dans la liste déroulante.

4) Vous devez choisir dans cette liste déroulante la version xmi de vos modèles. Il y a 3 choix possibles : “uml2 from eclipse”, qui correspond aux modèles directement exportés par le logiciel eclipse, “rpy” pour rhapsody, qui correspond aux modèles générés par rhapsody, et “Omondo” pour le plug-in eclipse éponyme.

5) Lorsque vous cliquez sur le bouton “Transform” un sélecteur de fichier s'ouvre. Choisissez dans quel fichier vous souhaitez enregistrer le résultat de la transformation puis appuyez sur le bouton “Save” et la transformation proprement dite sera lancée. A la fin du processus, selon si ce dernier a réussi ou échoué, une petit boîte de dialogue apparaît pour informer du résultat. En cas d'échec veillez à bien vérifier les chemins de vos fichiers et la version xmi.

iii. Analyse

Dans cet onglet, vous avez la possibilité de consulter vos fichiers sur un moteur XSB Prolog et d'envoyer des commandes afin de vérifier certaines propriétés si vous le désirez. Toutefois rien ne vous empêche d'utiliser n'importe quel moteur XSB Prolog pour analyser vos fichiers. Voici une image de l'onglet “Analyze” du logiciel UML2LP :



1) Le bouton “Rules file” permet d'ouvrir un sélecteur de fichier pour choisir le fichier contenant les règles que vous souhaitez vérifier. Ce fichier est consulté et les règles extraites de l'analyse du fichier apparaîtront dans la table en (3) avec le format indiqué par les en-têtes de colonne : numéro de la règle, nom/arité, évaluation. **Cette opération écrase toutes les règles qui étaient précédemment présentes dans la table.**

2) Le bouton “Add rules” permet d'ouvrir un sélecteur de fichier pour choisir le fichier contenant les règles que vous souhaitez vérifier. Ce fichier est consulté et les règles extraites de l'analyse du fichier apparaîtront dans la table en (3) avec le format indiqué par les en-têtes de colonne : numéro de la règle, nom/arité, évaluation. **Cette opération n'écrase pas les règles déjà présente dans la table mais les ajoutent en fin de liste.**

3) Cette zone contient une table dans laquelle sont stockées toutes les différentes règles. On y trouve comme information le numéro de la règle (arbitraire), le nom de la règle avec son arité, et enfin son évaluation. Lors du chargement du fichier, les règles ne sont pas évaluées.

- Une icône en forme de petit point d'interrogation indique que la règle n'est pas évaluée. Lorsqu'une règle est évaluée, le point d'interrogation est remplacé par une icône indiquant le résultat de l'évaluation.
- Une icône verte indique un résultat positif, c'est-à-dire qu'il y a au moins une réponse à la requête.
- Une icône rouge indique un résultat négatif, c'est-à-dire qu'il n'y a aucune réponse pour la requête.

Dans cette table, vous pouvez sélectionner les différentes règles que vous souhaitez vérifier simplement en cliquant dessus. En faisant CTRL + clic, la règle cliquée est ajoutée à la sélection. En faisant SHIFT + clic, toutes les règles depuis l'élément sélectionné avant le clic jusqu'à l'élément cliqué seront sélectionnées. En faisant CTRL + SHIFT + clic, toutes les règles depuis l'élément sélectionné avant le clic jusqu'à l'élément cliqué seront ajoutées à la sélection.

4) Le bouton “Select all” permet comme son nom l'indique de sélectionner toutes les règles présentes dans la table en (3).

5) Le bouton “Deselect all” permet comme son nom l'indique de désélectionner toutes les règles sélectionnées dans la table en (3).

6) Le bouton “Check” lance l'évaluation des différentes règles sélectionnées. Un résultat sommaire est donné dans le champ évaluation de la table sous forme d'icône pour une visualisation rapide. En plus de cela, les résultats complets des règles vérifiées apparaîtront dans la zone de texte résultat en (8), vous pourrez ainsi les sauvegarder si vous le désirez. Lorsqu'une règle est vérifiée, elle est évaluée par le moteur XSB Prolog sous cette forme : *nomRegle(Arg0, Arg1, ...)* le nombre de variables dépend de l'arité associée à cette règle. Si vous souhaitez mettre des valeurs particulières à la place de variables, vous pouvez taper votre requête dans le champ approprié en (7).

7) Vous pouvez taper vos commandes Prolog dans ce champ de texte. Lorsque vous appuyez sur la touche *return*, votre commande est envoyée et le résultat doit apparaître dans la zone de texte correspondante en (8). Si vous souhaitez recevoir les réponses suivantes, appuyez de nouveau sur la touche *return* en envoyant soit la meme commande soit une chaîne de caractères vide (la commande n'est pas effacée lorsqu'elle est envoyée). Si la commande est incorrecte, un message apparaît pour vous en informer dans la zone texte "Results" en (8).

8) Cette zone de texte indique les résultats des opérations réalisées. Elle n'est pas éditable. Elle dispose de barres de défilement.

9) Le bouton "Save" permet d'ouvrir un sélecteur de fichier afin de choisir un fichier dans lequel enregistrer le contenu de la zone de texte résultat. Attention, toutes les données présentes sur ce fichier sont supprimées par la procédure.

10) Le bouton "Clear" permet d'effacer le contenu de la zone de texte résultat. Si vous ne souhaitez sauvegarder qu'un résultat précis dans un fichier, il vous est conseillé d'appuyer sur ce bouton, de lancer l'opération qui produit le résultat que vous souhaitez, puis d'enregistrer le contenu de la zone de texte dans un fichier (9).


Le moteur XSB Prolog ne dispose que d'une fonction permettant d'obtenir un but déterministe, c'est-à-dire qu'elle ne permet d'avoir qu'un seul résultat. Afin de contourner le problème, les commandes envoyées par l'utilisateur sont encapsulées dans un *findall* de la manière suivante : *findall((variables), (commande_utilisateur), L)*. Par exemple si l'utilisateur envoie la commande *test(Arg0, cst1)*, le moteur XSB Prolog exécutera au final *findall((Arg0), (test(Arg0, cst1)), L)*. C'est la liste L contenant les résultats qui est ensuite fournie. Cela signifie donc que tous les résultats sont calculés dès l'envoi de la commande. Si la commande envoyée est inadéquate, le moteur XSB Prolog peut se bloquer dans le *findall*. Il n'y a alors pas d'autre possibilité que de redémarrer le programme.


Attention, il n'est pas possible de redémarrer le moteur Prolog sans redémarrer le programme, toutes les clauses et prédicats chargés resteront chargés jusqu'à ce que vous fermiez le programme. Il est donc possible que si vous chargez plusieurs fichiers d'affilé, certaines clauses ou prédicats se recoupent et provoquent des erreurs. Il peut même arriver que le moteur Prolog ne veuille pas compiler un fichier donc si vous avez ce type d'erreur lorsque vous essayez de charger un fichier de règles, il est recommandé de relancer le programme et de réessayer.

iv. Menus

"Menu" contient un bouton "Quitter" qui comme son nom l'indique permet de quitter l'application. Une fenêtre apparaîtra vous demandant de confirmer que vous souhaitez quitter l'application. ALT + Q est un raccourci clavier permettant

de quitter l'application. Attention si vous cliquez sur la petite croix pour fermer la fenêtre, il n'y aura pas de demande de confirmation.

"Options" contient un boîte à cocher et un bouton. Parfois lors de la résolution d'une requête Prolog, plusieurs chemins peuvent mener à une même réponse. Si vous ne souhaitez voir apparaître la réponse qu'une seule fois, vous avez la possibilité de cocher l'option "Suppr. redundancies" dans le menu "Options". Ainsi toutes les redondances seront supprimées garantissant l'unicité de chaque résultat lors d'une requête. Le bouton "Load ID file" permet d'ouvrir un sélécteur de fichiers afin de choisir un fichier contenant les correspondances entre les IDs de programmation logique et les IDs du xmi, un fichier de correspondance entre les IDs xmi et les IDs Prolog est automatiquement généré à chaque transformation, veuillez vous référer à la partie "Fichiers de sortie" pour plus d'informations.

"Aide" contient un "A propos de..." qui ouvre une fenetre d'information donnant des renseignements sur l'application.

IV/ Stabilité

Des problèmes d'instabilité ont été observés et il peut arriver que le programme se termine brutalement sans raison apparente comme par exemple à l'ouverture d'un sélecteur de fichier, afin de réduire cette instabilité la dimension de la fenêtre est fixée en (1024x768). Si vous êtes victime d'une instabilité il vous faudra relancer le programme.

Si le programme ferme brutalement lors d'une transformation, il est possible que cela soit dû au fait que le processus de transformation n'arrive pas à traiter les fichiers fournis en entrée et lève une exception. Malheureusement à cause des problèmes d'instabilité cela provoque la plupart du temps une fermeture brutale du programme. Il est tout de même recommandé de tenter à nouveau la transformation et si le problème persiste, cela signifie que le programme est incapable de traiter correctement les fichiers fournis en entrée. Peut-être la section traitant du métamodèle et du modèle permettra de résoudre le problème.

Il est également possible, bien que ce soit beaucoup plus rare, que le programme se ferme brutalement lors d'une requête Prolog. Comme précédemment il est recommandé de réessayer avant de conclure que la requête ne peut être traitée par le programme (il est également recommandé de vérifier la justesse de la requête).

V/ Formats des fichiers

1) Metamodèle

Le programme traite les métamodèles selon la norme uml2. Toutefois certains éléments sont à prendre en considération pour que la transformation se

déroule bien :

- Seuls les éléments portant le nom de “ownedMember” situés à la racine du document sont pris en compte pour la formation des métafaits ainsi que leurs éléments fils (exemple : <ownedMember xmi:type="uml:Class" xmi:id="_m8qbC686EdiEh75YJ_3n8g" name="Element" isAbstract="true">.

- Afin de pouvoir fonctionner correctement en Prolog, certaines chaînes de caractères peuvent être modifiées. En effet, en Prolog les constantes et les prédicats commencent par des minuscules, les majuscules et les '_' sont réservés pour les variables.
  Par exemple :
  <ownedMember xmi:type="uml:Class" xmi:id="_m8qbJK86EdiEh75YJ_3n8g" name="Comment">
  <ownedAttribute xmi:id="_m8qbJq86EdiEh75YJ_3n8g" name="body" type="_m8qbB686EdiEh75YJ_3n8g"/>
  </ownedMember>
  En Prolog nous obtiendrons ceci : comment(Id,Body).
  Le nom “Comment” fut modifié pour créer le prédicat “comment”. Le nom “body” fut modifié pour créer la variable “Body”. Uniquement la première lettre est modifiée si besoin est.

- Les IDs des métamodèles ne sont pas stockées et ne sont pas utilisées pour identifier des faits. Ce sont les noms qui sont utilisés pour former les prédicats. Attention donc à ne pas avoir des éléments de même nom sous peine de mélanger les prédicats, bien que certains peuvent toujours être identifiés s'ils ont des arités différentes. Exemple : <ownedMember xmi:type="uml:Class" xmi:id="_m8qbJK86EdiEh75YJ_3n8g" name="Comment"/> donnera le prédicat comment(...).


Si vous avez des problèmes durant la transformation, vérifiez bien que les éléments à la racine portent le nom de “ownedMember” et non pas “packagedElement”. Il est egalement possible que des problèmes surviennent si certains attributs ont une chaîne de caractères nulle en guise de valeur.

2) *Modèle*

Le programme dispose de trois types d'extracteurs d'informations pour les modèles : “uml2 from eclipse”, “rpy” pour rhapsody, et “Omondo”. Chacun est fait pour prendre en compte les spécificités de ces modèles. Toutefois il est conseillé d'utiliser le type “uml2 from eclipse” car les autres types d'extracteurs n'ont pas été corrigés et peuvent présenter un plus grand nombre de bugs.

Tableau des fonctionnalités :

Type	Extension	Stéréotypes
uml2 from eclipse	*.uml2;*.uml	

rpy	*.xmi	
Omondo	*.xmi	

Les informations suivantes sont spécifiques au type "uml2 from eclipse", elles peuvent vous aider à comprendre ce qu'il se passe lors de la transformation et éventuellement vous aider à régler des problèmes que vous pourriez rencontrer :

- Les éléments pris en compte dans le modèle sont les éléments qui disposent d'une ID (attribut "xmi:id").

- Les éléments dont le type n'a pas pu être déterminé à l'aide de l'attribut "xmi:type" ou par héritage sont abandonnés et aucun fait en programmation logique ne sera créé à partir d'eux. Ceci peut arriver lorsqu'il n'y a pas d'attribut "xmi:type" ou lorsque le type trouvé n'est pas dans le métamodèle. Il existe une exception : les éléments se nommant "uml:Model" sont considérés de type "uml:Model" et n'ont donc pas besoin d'avoir un attribut "xmi:type" (exemple : <uml:Model xmi:id="idModel" name="nameModel">).

- Afin de pouvoir fonctionner correctement en Prolog, certaines modifications sont apportée aux IDs en particulier et aux chaînes de caractères en général. En effet, en Prolog les constantes et les prédicats commencent par des minuscules, les majuscules et les '_' sont réservés pour les variables. De nouvelles IDs sont donc générées lors de la transformation du modèle en faits afin de s'assurer de la bonne fonctionalité en Prolog. Toutefois une correspondance entre les nouvelles et les anciennes IDs est conservée en mémoire après une transformation et, lors d'une requête Prolog, si le résultat est une ID, la correspondance sera donnée en même temps (exemple : xmi:id="_DcxBMIVAEdycVIhX7a-lHg" correspondra à l'ID "id0" en Prolog). Attention toutefois, si le programme est quitté, il vous faudra recharger manuellement le fichier des correspondances que vous souhaitez en utilisant l'option adéquate dans le menu (voir les notes sur les fichiers de sortie).

3) Gestion des stéréotypes

Une spécificité a été implémentée dans le processus de transformation afin de gérer les stéréotypes du métamodèle présents sous forme de profile dans le modèle. Toutefois cette fonctionnalité est restrictive car implémentée seulement pour un type de profile.

Explications : lors de la transformation du métamodèle, un traitement spécial est réalisé pour les éléments de nom "ownedMember" situés à la racine et de type "uml:Stereotype". Ces éléments là sont traités de la meme manière que des éléments de type "uml:Class". Ce traitement est effectué afin de permettre aux stéréotypes de générer des métafaits qui pourront être instanciés lors de la

transformation du modèle en faits. Les stéréotypes sont donc considérés comme des classes toutefois ils gardent un marqueur indiquant qu'ils sont des stéréotypes. Le seul moyen d'obtenir un fait à partir d'un métafait issu d'un stéréotype est d'avoir un élément de ce type dans le modèle :
<profileSPEM:NomDuStereotype xmi:id="id" base_X="idDeLElement"/>. Attention il est nécessaire que le préfixe du nom de l'élément soit "profileSPEM". Le "NomDuStereotype" doit contenir le nom du stéréotype que l'on souhaite appliquer sur un élément du modèle dont l'ID est référencée dans l'attribut "base_X" (X pouvant contenir n'importe quelle chaîne de caractères pourvu que le nom de l'attribut commence par "base_"). Exemple :
Dans le métamodèle :
<ownedMember xmi:type="uml:Stereotype" xmi:id="_HEMpUGLSEdywkqNqxTe0ag" name="SPEMActivity">
<generalization xmi:id="_JW0zQGLSEdywkqNqxTe0ag" general="_tChFkGLQEdywkqNqxTe0ag"/>
</ownedMember>
Dans le modèle :
<node xmi:type="uml:CallBehaviorAction" xmi:id="_KTkWYFU_EdycVIhX7a-lHg" name="Specification" outgoing="_eBQB91U_EdycVIhX7a-lHg" incoming="_dneQBIU_EdycVIhX7a-lHg _fWCqplU_EdycVIhX7a-lHg"/>
et :
<profileSPEM:SPEMActivity xmi:id="_99Oq4Fb2EdyIC7s5y8Cnbg" base_Action="_KTkWYFU_EdycVIhX7a-lHg"/>

Les correspondances ont été mises en couleur pour une meilleure visualisation. Les contraintes de cette utilisation sont liées au fait que cette fonctionnalité fut créée pour satisfaire une nécessité spécifique.

4) Fichiers de ressources

Le programme utilise un certain nombre de fichiers pour remplir certaines fonctionnalités, si ces fichiers sont effacés par erreur il est possible que certaines de ces fonctionnalités ne marchent plus. Dans le dossier *MY_UML2LP_DIR/Ressources/* il y a normalement :

- *greencheck.jpg* : il s'agit de l'image d'une petite coche sur rond vert. Cette image est utilisée dans la table des règles de l'onglet "Analyze".
- *interrogation.jpg* : il s'agit de l'image d'un petit point d'interrogation. Cette image est utilisée dans la table des règles de l'onglet "Analyze".
- *redcross.jpg* : il s'agit de l'image d'une petite croix sur rond rouge. Cette image est utilisée dans la table des règles de l'onglet "Analyze".
- *MetaTransFiles* : ce fichier est utilisé pour stocker les chemins des métamodèles sélectionnés lors de transformations. Ainsi ces chemins apparaîtront dans les boîtes de séléction même après un redémarrage du programme.
- *ModelTransFiles* : ce fichier est utilisé pour stocker les chemins des modèles sélectionnés lors de transformations. Ainsi ces chemins apparaîtront dans les boîtes de séléction même après un redémarrage du programme.
- *ModelTempFile* : ce fichier est utilisé lorsque l'utilisateur utilise la fonction "Transform" de l'onglet "Compare". Le document formé par la fusion des deux modèles est stocké dans ce fichier qui est ensuite lu lors de la transformation proprement dite.

5) Fichiers de sorties

Les fichiers de ce dossier sont générés lors de la transformation. Ils contiennent une partie ou une étape de la transformation et ne sont pas censé être intéressants. Le seul fichier pouvant vous être utile est le fichier nommé "mapping_of_XMI_and_LP_ID.txt". Lors d'une transformation, les correspondances entre les IDs du xmi et les IDs générées pour fonctionner en programmation logique sont stockées dans ce fichier. Vous pouvez copier et renommer le fichier si vous souhaitez garder une copie de ces correspondances allant avec le résultat d'une transformation. Il vous est possible de charger ce type de fichier à partir de l'option du menu "Load ID File", en effet lorsque vous quittez le programme, les correspondances d'IDs ne sont plus en mémoire et il est nécessaire de les charger à nouveau manuellement (pour éviter d'avoir à refaire une transformation inutile si l'on souhaite simplement exécuter des requêtes Prolog). A noter qu'il n'est pas indispensable de charger un fichier de correspondances pour faire fonctionner Prolog.

Formalisation des processus de l'Ingénierie Système :
Proposition d'une méthode d'adaptation des processus génériques à différents contextes d'application

Face à la complexité croissante des systèmes à développer, chercheurs et ingénieurs se mobilisent pour dégager des méthodologies et des outils adaptés et contribuent ainsi à créer une discipline nouvelle : l'ingénierie système, dont le progrès se situe dans un effort général de standardisation et de normalisation pour pouvoir largement partager les acquis entre tous les partenaires et rendre les outils interopérables.

Dans cette thèse, nous considérons le problème sous l'angle de la coordination des activités relevant du processus général de développement, pour, à long terme, aboutir à une représentation et à une planification de toutes ces activités. Nous nous intéressons particulièrement au problème de l'interaction des processus multiples nécessaires à la réalisation d'un projet d'ingénierie. Pour cela, nous avons effectué un programme de recherche en trois étapes :

- la formalisation d'un processus générique à partir d'un standard recommandé par la communauté (EIA-632) sous la forme d'un modèle UML utilisant le profil SPEM,
- la spécialisation de ce processus générique par l'intégration de normes et de pratiques de l'entreprise puis par l'intégration des éléments particuliers du projet,
- l'enrichissement de ce processus pour en préparer l'exploitation et la conduite.

L'approche retenue repose sur l'idée, répandue en ingénierie système, qu'il existe une solution générique aux problèmes d'ingénierie. Notre proposition est de formaliser cette solution pour permettre de l'adapter ensuite à des domaines d'activités et des projets précis par des opérations de transformation de modèles. Pour cela, nous nous reposons sur les concepts de l'IDM et les appliquons dans le cadre de l'ingénierie système.

Nous montrons que la formalisation est la base d'une démarche nouvelle d'application des processus de l'ingénierie système. Cette démarche permet d'assurer une cohérence globale et locale dans l'organisation et le déroulement des projets. En se reposant sur l'exploitation de règles de bonnes pratiques compilées dans des standards internationaux tels que l'EIA-632, elle s'assure d'un comportement global cohérent du projet et, en laissant les acteurs du projet libres d'adapter leurs pratiques effectives à leur environnement de travail, elle permet à chacun de travailler de manière optimale avec l'assurance que les processus spécifiques qu'il emploie s'insèrent au mieux dans le projet.

En cela, la démarche que nous proposons répond à la problématique d'application concrète des processus d'ingénierie système posée par l'industrie et, d'une manière plus générale, à celle de l'amélioration continue des méthodes et des produits.