Vers une nouvelle forme d'accompagnement des processus dans les systèmes interactifs : apport de la fouille de processus et de la recommandation

Joffrey Leblay

Vers une nouvelle forme d'accompagnement des processus dans les systèmes interactifs

Apport de la fouille de processus et de la recommandation

Par

JOFFREY LEBLAY

Laboratoire Informatique, Image et Interaction

LA ROCHELLE UNIVERSITÉ

ÉCOLE DOCTORAL EUCLIDE

DIRECTEUR DE THÈSE :
Ronan Champagnat

CO-DIRECTEUR DE THÈSE :
Mourad Rabah

RAPPORTEUR :
Anne Boyer

RAPPORTEUR :
Patrice Boursier

EXAMINATEUR :
Samuel Nowakowski

EXAMINATEUR :
Michel Ménard

SEPTEMBRE 2019

RÉSUMÉ

Un système d'information est un système socio-technique comprenant des processus métier et des données afférentes. Avec le développement et la démocratisation des outils informatiques, les informations stockées sont plus importantes et réparties. Il en est de même pour les processus qui deviennent de plus en plus complexes et sensibles pour les organisations. Afin d'obtenir un service, nous sommes de plus en plus amenés à composer des processus métiers pour collecter l'information, la transformer et la réinjecter.

L'objectif de cette thèse consiste à explorer la problématique du pilotage de processus afin de donner des pistes pour la fabrication d'un compagnon qui guiderait l'utilisateur dans sa phase de découverte des processus. Nous avons concentré nos efforts sur les aspects faisabilité informatique. Plus particulièrement, nous étudions la possibilité de définir une méthodologie de recommandation à partir des processus d'usage et de mettre en place l'architecture logicielle correspondante.

Les travaux présentés se trouvent à l'interface entre plusieurs domaines. Nous avons choisi une démarche de recherche qui s'appuie sur un cycle itératif. Après avoir analysé le domaine de la fouille de processus puis la recommandation, nous avons déduit qu'il fallait renforcer notre approche sur la collecte d'information. Ceci nous a conduit à mener des études sur les systèmes à base de traces. Nous avons ensuite cherché à valider la continuité de notre approche sur un cas d'étude simple. Il s'agit de personnaliser le parcours d'un étudiant au cours de sa formation. Nous avons mis en place un démonstrateur qui permet, à partir du recueil des informations issues des promotions précédentes, d'extraire de la connaissance sur les parcours des étudiants et d'émettre des recommandations sur les suites du parcours pour un étudiant particulier. Cette étude nous a permis de mettre en place notre processus de recommandation de bout en bout et de proposer une première esquisse de notre architecture.

Nous avons ensuite cherché un cas d'étude plus ambitieux pour lequel aucun processus métier n'a été prédéfini par un expert. Nous souhaitions voir s'il est possible d'identifier des comportements et/ou stratégies d'utilisateurs vis-à-vis d'un système. Nous nous sommes placés dans un contexte d'apprentissage où l'apprenant est impliqué dans une simulation d'un micro-monde. Ce cas d'étude nous a permis de montrer comment adapter notre méthodologie et comment prendre en compte les données contextuelles.

Ce cas d'étude a donné lieu à une expérimentation où deux groupes ont utilisé notre simulateur. Le premier sans recommandation, ce qui nous a permis de constituer un ensemble de traces d'exécution qui ont servi à extraire les connaissances nécessaires sur nos processus métier. Le second groupe a bénéficié de notre système de recommandation. Nous avons observé que dans ce dernier groupe le critère de performance était amélioré car les phénomènes d'essais/erreurs se trouvent considérablement réduits.

L'expérience acquise au cours de cette thèse nous pousse à orienter nos travaux vers l'aide à la personnalisation de parcours d'apprentissage. En particulier avec la définition des formations en blocs de compétences, la prise en compte du profil de l'apprenant, qu'il s'agisse de ses connaissances acquises aussi bien que de ses stratégies d'apprentissage, conduit à fabriquer une trajectoire d'apprentissage, et donc une sélection de blocs de formation, qui doit être personnalisée. La méthodologie que nous avons proposée constitue une brique pour fabriquer un tel écosystème.

Abstract

An information system is a socio-technical system comprising Business Processes and related data. With the development and democratization of IT tools, stored information is getting more important and distributed. The same is true for processes that are becoming increasingly complex and sensitive for organizations. In order to obtain a service, we had to compose business processes to collect information, transform it and reinject it.

The objective of this thesis is to explore the problematic of process control in order to give options for the fabrication of a companion that would guide the user when discovering a process. We focused on computer science aspects. In particular, we are studying the possibility of defining a recommendation methodology based on extracted processes and implementing the corresponding software architecture.

The works presented are at the interface between several domains. We chose a research approach based on an iterative cycle. After analyzing the field of process mining and recommendation, we concluded that we needed to strengthen our approach to information gathering. This led us to carry out studies on trace-based systems. We then sought to validate the continuity of our approach on a simple case study. It is about personalizing the course of a student during his training. We have set up a demonstrator which, based on the collection of information from previous promotions, extracts knowledge about the students' courses and makes recommendations on the consequences of the course for a particular student. This study allowed us to set up our end-to-end recommendation process and to propose a first sketch of our architecture.

We then looked for a more ambitious case study for which no business process was predefined by an expert. We wanted to see if it is possible to identify behaviors and / or strategies of users using a system. We have placed ourselves in a learning context where the learner is involved in a simulation of a micro-world. This case study allowed us to show how to adapt our methodology and how to take contextual data into account.

This case study gave rise to an experiment where two groups used our simulator. The first without recommendation, which allowed us to build a set of execution traces that were used to extract the necessary knowledge on our business processes. The second group benefited from our recommendation system. We observed that in the latter group the performance criterion was improved because the trial / error phenomena are considerably reduced.

The experience gained during this thesis pushes us to direct our work toward helping to personalize learning trajectory. In particular, with the definition of a class, taking into account the learner's profile, both in terms of knowledge acquired and learning strategies, leads to the creation of a learning path, and therefore a selection of training blocks, which must be personalized. The methodology we have proposed is a brick to build such an ecosystem.

Remerciements

Je tiens à remercier très chaleureusement tous ceux qui ont contribué à la réalisation de cette thèse. Je tiens tout d'abord à remercier sincèrement Nouredine Tamani, pour sa patience et les conseils qu'il m'a donnés malgré ses charges professionnelles. Il m'a soutenu quand j'étais au plus bas et m'a poussé refaire surface. Grâce à lui et aux nombreuses relectures de ce document, mon travail de rédaction a pu être amélioré.

Mes remerciements vont aussi à Ronan Champagnat et Mourad Rabah pour m'avoir accordé cette opportunité de faire cette thèse. Je les remercie pour leur encadrement durant ces années et pour leur aide toujours là même dans les moments difficile. Je remercie aussi Samuel Nowakowski avec qui j'ai travaillé durant toute cette thèse. Mon travail, notamment la rédaction d'articles de recherche et de ce manuscrit n'aurait pas été le même sans lui.

Je remercie le L3I et l'équipe e-ADAPT de m'avoir accueilli pendant mes travaux ainsi que le LORIA et l'équipe KIWI pour m'avoir fait découvrir leur travaux durant mon séjour chez eux.

Des remerciements particuliers pour ma famille qui a toujours été là pour me soutenir pendant de longues périodes de doutes ainsi que pour avoir cru en moi durant ces longues années d'études.

Enfin, j'adresse mes plus chaleureux remerciements à tous mes amis qui ont déjà ou pas encore soutenu au laboratoire L3i. Chacun a contribué à sa façon à faire que ce travail soit possible.

Un gros merci à toutes et à tous car sans vous, achever ces travaux de thèse n'aurait pas été possible.

TABLE DES MATIÈRES

Page

1 Introduction **1**

1.1 Évolution des systèmes d'information . . . 2
1.2 Objectif . . . 3
1.3 Positionnement . . . 4
1.4 Questions de recherche . . . 5
1.5 Démarche . . . 6
1.6 Plan du manuscrit . . . 6

2 Analyse des traces **9**

2.1 Introduction . . . 10
2.2 Définitions . . . 11
2.2.1 Trace . . . 11
2.2.2 Trace numérique . . . 11
2.2.3 Trace brute ou première . . . 12
2.2.4 Source de traçage . . . 12
2.2.5 Système à base de traces . . . 12
2.3 Objectifs de l'utilisation des traces . . . 12
2.3.1 Suivi de l'utilisateur . . . 13
2.3.2 Partage d'expérience . . . 14
2.4 Système à base de traces (SBT) . . . 15
2.4.1 Trace modélisée . . . 15
2.4.1.1 Application d'un modèle de traces . . . 15
2.4.1.2 Format des traces . . . 16
2.4.2 Principe d'un système à base de traces modélisées . . . 17
2.4.2.1 Collecte de traces . . . 18
2.4.2.2 Transformation des traces . . . 19
2.4.2.3 Exploitation des traces . . . 21
2.5 Systèmes à base de traces existants . . . 21
2.5.1 Streaming Application Trace Miner (SATM) . . . 22
2.5.2 Collaboration Analysis Tool (ColAT) . . . 22

2.5.3 Aplusix . . . 23
2.5.4 Modeling USEs and Tasks for Tracing Experience (MUSETTE) . . . 23
2.5.5 Trace Analysis Tool for Interaction ANAlysis (TATIANA) . . . 24
2.5.6 VISU . . . 24
2.5.7 Système à base de traces pour le calcul d'indicateur dans Moodle (SBT-IM) . . . 25
2.5.8 Projet d'Intégration de l'eXpérience pour l'Enseignement à Distance . . . 25
2.5.9 Étude comparative des SBT présentés . . . 26
2.6 Conclusion . . . 27

3 Fouille de processus 29
3.1 Introduction . . . 30
3.2 Fouille de processus (*Process Mining*) . . . 31
3.2.1 Utilisation du process mining . . . 31
3.2.2 Organisation des données . . . 32
3.2.3 Principe général de la fouille de processus . . . 33
3.3 Motifs courants dans les processus . . . 33
3.4 Modélisation des processus . . . 35
3.4.1 Réseaux de Petri (RdP) . . . 35
3.4.2 Workflow Net . . . 37
3.4.3 Process tree . . . 38
3.4.4 Graphe causal (Causal Net) . . . 38
3.5 Les algorithmes de fouille . . . 39
3.5.1 Algorithmes de base . . . 39
3.5.2 Algorithmes statistiques . . . 40
3.5.3 Algorithmes génétiques . . . 41
3.6 Propriétés du modèle miné . . . 42
3.6.1 Alignement du modèle et des traces . . . 43
3.6.2 Justesse . . . 43
3.6.3 Précision . . . 44
3.6.4 Généralisation . . . 44
3.6.5 Simplicité . . . 45
3.7 Limites . . . 45
3.7.1 Source de données hétérogènes . . . 46
3.7.2 Bruit . . . 46
3.7.3 Traces incomplètes . . . 46
3.7.4 Boucles . . . 47
3.7.5 Choix non-libres . . . 47
3.8 Principaux algorithmes . . . 48
3.8.1 Algorithme α . . . 48
3.8.2 Algorithme $\alpha+$ . . . 48
3.8.3 Algorithme $\alpha++$ . . . 49

3.8.4 *Heuristic Miner* . . . 49
3.8.5 *Inductive Miner* . . . 50
3.8.6 *Inductive Miner Incompleteness* . . . 50
3.8.7 *Inductive Miner Infrequent* . . . 50
3.9 Conclusion . . . 51

4 Recommandation **53**
4.1 Introduction . . . 54
4.2 Les systèmes de recommandation . . . 54
4.3 Terminologie et concepts . . . 55
4.3.1 Définitions . . . 56
4.3.2 Entrées d'un système de recommandation . . . 57
4.3.3 Sorties d'un système de recommandation . . . 57
4.4 Classification des systèmes de recommandation . . . 58
4.4.1 Recommandation collaborative ou par filtrage collaboratif . . . 58
4.4.2 Recommandation basée sur le contenu . . . 58
4.4.2.1 Recommandation basée sur un modèle . . . 59
4.4.2.2 Recommandation à base de contexte . . . 59
4.4.3 Filtrage hybride . . . 60
4.5 Calcul de similarité . . . 61
4.6 Entraves à la recommandation . . . 62
4.7 Bilan . . . 63
4.8 Conclusion . . . 64

5 Méthodologie pour la recommandation d'activités **67**
5.1 Introduction . . . 68
5.2 Principe de recommandation dans les processus . . . 69
5.3 Méthodologie . . . 71
5.3.1 Extraction des traces modélisées . . . 71
5.3.2 Modèle de processus métier réalisé . . . 74
5.3.3 Proposition de recommandation pour l'utilisateur . . . 75
5.4 Validation de l'architecture proposée . . . 76
5.4.1 Description du cas d'étude . . . 77
5.4.2 Format des données . . . 78
5.4.3 Application de la méthode . . . 78
5.4.4 Discussion . . . 79
5.5 Conclusion . . . 81

6 Étude de cas : Tamagotchi **83**
6.1 Introduction . . . 84
6.2 Description du cas d'étude Tamagotchi . . . 85
6.2.1 Motivations . . . 85

6.2.2 Principe du jeu . . . 86
6.2.3 Attributs du Tamagotchi . . . 86
6.2.4 Critères de vie du Tamagotchi . . . 87
6.2.4.1 Santé . . . 89
6.2.4.2 Socialisation . . . 89
6.2.4.3 Maturité . . . 90
6.2.4.4 Déroulement . . . 90
6.2.5 Activités proposées au joueur . . . 90
6.2.6 Format des traces . . . 91
6.3 Adaptation de la méthode au cas d'étude Tamagotchi . . . 93
6.3.1 Formalisation de l'approche proposée . . . 93
6.3.2 Impact des activités sur l'évolution des critères . . . 95
6.3.3 Formalisation du contexte dans le modèle de processus . . . 96
6.3.4 Recommandation à partir du modèle de processus enrichi . . . 98
6.4 Expérimentation . . . 100
6.4.1 Paramètres et protocole expérimental . . . 101
6.4.2 Présentation des résultats expérimentaux . . . 102
6.5 Bilan . . . 104
6.6 Conclusion . . . 105

7 Conclusion **107**
7.1 Synthèse des travaux . . . 108
7.2 Bilan . . . 110
7.3 Perspectives . . . 111

Liste des tableaux

TABLEAUX **Page**

2.1 Comparaison des modèles de traces 16
2.2 Comparaison des SBT 27

5.1 Exemple d'une table de traces brutes 72
5.2 Représentation des variables dans une table 72
5.3 Exemple de table de traces d'événements 74

6.1 Les attributs du Tamagotchi 88
6.2 Influence des activités sur les attributs 92
6.3 Exemple des traces de Tamagotchi 92
6.4 Traces collectées à partir de processus non structurés 97

Table des figures

FIGURES **Page**

1.1 Positionnement aux croisements des différents domaines de recherches 5

2.1 Partage d'expériences entre utilisateurs de profils différents (Sehaba, 2014) 14
2.2 Principe général d'un système à base de traces modélisées (Loghin, 2008) 18
2.3 Transformation de traces (Settouti et al., 2009) 20
2.4 Schéma fonctionnel TATIANA (Dyke et al., 2010) 24

3.1 Positionnement de la fouille de processus (Van Der Aalst, 2016) 31
3.2 Motifs élémentaires de processus 34
3.3 Exemple de boucles 34
3.4 Évolution du marquage d'un RdP 36
3.5 Exemple de Workflow-net 37
3.6 Exemple de modèle de processus en fleur 44
3.7 Exemple de choix non libre 47

5.1 Étapes de notre méthode de recommandation fondée sur les processus 70
5.2 Exemple de transformation des traces brutes en traces modélisées 71
5.3 Workflow-net correspondant aux traces modélisées 75
5.4 Interactions entre les données et les composants durant l'étape de recommandation . . 76
5.5 Représentation schématique des exécutions des traces 79
5.6 Extrait du modèle représentant l'enchaînement des décisions du jury 80

6.1 Modèle réel du jeu Tamagotchi 86
6.2 Croisement des attributs et des critères de vie 87
6.3 Interface de l'application Tamagotchi 91
6.4 Mise en évidence des modifications à l'approche proposée 94
6.5 Modèle de processus découvert depuis les traces du tableau 6.4 97
6.6 Mise en œuvre de la méthode proposée sur le cas Tamagotchi 101
6.7 Moyenne et écart-type de la durée par victoire et par groupe 103
6.8 Moyenne et écart-type du nombre de défaites consécutive avant une victoire 103
6.9 Moyenne et écart-type du pourcentage de bruit par victoire et par groupe 104

CHAPITRE 1 | Introduction

Sommaire

1.1 Évolution des systèmes d'information 2
1.2 Objectif 3
1.3 Positionnement 4
1.4 Questions de recherche 5
1.5 Démarche 6
1.6 Plan du manuscrit 6

1.1 Évolution des systèmes d'information

Un système d'information est un système socio-technique comprenant des processus métier (c'est-à-dire une succession d'activités permettant d'atteindre un objectif, par exemple lorsque nous voulons faire un gâteau nous suivons la recette) et des données afférentes. Avec le développement et la démocratisation des outils informatiques, les informations stockées sont plus importantes et réparties. Il en est de même pour les processus qui deviennent de plus en plus complexes et sensibles pour les organisations. Afin d'obtenir un service, nous sommes de plus en plus amenés à composer des processus métiers pour collecter l'information, la transformer et la réinjecter. Dans cette thèse nous étudierons les moyens pour définir un compagnon intelligent afin d'aider un utilisateur lors de la réalisation d'un service.

De plus, les entreprises laissent à leurs clients la main sur certains processus et non plus uniquement à leurs employés, formés aux processus du système d'information. Le cas le plus fréquent concerne les achats en ligne. En effet, l'utilisateur se retrouve sur un site marchand et navigue en toute liberté sur un catalogue de produits et enregistre des produits dans son panier. Une fois cette collecte terminée, il peut passer à la phase de finalisation de son achat. Cette dernière est très encadrée et met en œuvre un processus. Il en va de même avec la saisie d'informations dans des administrations. Or dans ce cas, si les processus sont clairement établis, l'enchaînement (l'assemblage) des processus par l'utilisateur n'est pas aussi facile que l'on peut le penser.

Autant un employé est formé afin d'acquérir les compétences métiers de l'entreprise, quoique de moins en moins, autant l'utilisateur n'est pas forcément disposé à lire les modes d'emplois et, malgré les efforts sur l'ergonomie, n'intègre pas la logique métier du processus qu'il essaie de réaliser. La conséquence est que le client peut se trouver dans des situations d'échecs et de stress, ce qui rend l'utilisation de l'application moins pertinente. Ce problème est bien évidemment plus prégnant lors de premières exécutions. Généralement les utilisateurs améliorent leurs pratiques au cours de l'exécution d'un processus et sont dans une dynamique « d'apprendre en faisant ». Ainsi, nous nous familiarisons au fur et à mesure des utilisations d'un processus comme le font les employés d'une entreprise. Cela explique aussi le fait qu'après avoir fait une erreur, nous savons comment la corriger et/ou ne pas la refaire.

Cependant, si l'on peut penser que l'utilisateur est prêt à s'investir pour mieux utiliser les processus métier d'un site marchand qu'il affectionne particulièrement, il n'en est pas de même pour des services que l'on utilise rarement. Il convient alors de proposer un accompagnement durant l'exécution de ce processus. Pour y arriver, il est nécessaire de collecter des informations. De plus, on observe que même si un processus est fortement structuré par morceaux, l'utilisateur parvient toujours à trouver des cheminements complexes. Nous nous situons donc dans le cadre d'un système d'information pour lequel les processus métier sont faiblement structurés et pour lesquels il y a peu de connaissances *a priori*.

Dans le cadre de cette thèse nous cherchons à mettre en place un cadre méthodologique pour définir un compagnon afin d'améliorer l'usage de processus métier. Pour cela, nous ne considérons que des processus qui sont observables depuis un environnement informatique. Par exemple, un processus observable depuis un système d'information d'entreprise laissant des traces. Nous utilisons ces traces pour trouver un lien entre une exécution en cours d'utilisation et des exécutions similaires, passées, d'utilisateurs dans le même environnement ayant atteint le même objectif. Le principe sera de proposer à l'utilisateur une suite d'activités lui permettant d'atteindre son objectif.

1.2 Objectif

Le développement des systèmes d'information, leur hétérogénéité et l'agilité nécessaire pour la performance des organisations, font que les processus sont complexes et l'utilisateur a besoin d'assistance. Mais de quelle manière doit-on l'assister ? Et est-ce possible ? Nous ne cherchons pas à mettre en place un outil de supervision qui utiliserait des raisonnements temporels pour réaliser une planification d'activités. En effet, de tels systèmes contraignent souvent l'utilisateur ne lui laissent aucune liberté. Nous ne cherchons pas, non plus, à proposer une aide pour la réalisation d'une activité. Notre objectif consiste à définir une solution pour piloter l'utilisateur dans l'exécution d'un processus, ou d'un ensemble de processus, en lui suggérant les activités à réaliser.

Suggérer l'activité à réaliser nécessite de posséder une connaissance sur le processus, l'état du système, le contexte et l'objectif de l'utilisateur. Les outils de modélisation des systèmes d'information permettent de construire une connaissance sur les processus. Cependant, il existe une divergence entre les processus prévus par le concepteur du système et ceux réalisés par les utilisateurs. Il est nécessaire de baser notre connaissance des processus métier sur les modèles d'usage extraits à partir des traces d'exécution. Nous devons également nous poser la question de la forme sous laquelle nous fournissons l'assistance à l'utilisateur.

Bien souvent, l'utilisateur est fortement contraint. Si cette contrainte est rassurante pour le concepteur du système car elle garantit une qualité moyenne élevée sur le service, elle ne permet pas à l'utilisateur de gérer des cas exceptionnels ni ne lui laisse de degrés de liberté. Ce dernier point est problématique, car l'utilisateur ne devient qu'un simple exécutant pourtant mieux à même d'interpréter une situation complexe qu'une machine, mais dont on ne sollicite pas les compétences. L'engagement de l'utilisateur, et donc la qualité du travail, s'en trouve pénalisé. Il convient de proposer des environnements où l'ordinateur est une véritable aide à l'utilisateur en réalisant les travaux pénibles et répétitif pour lesquels il n'y a aucune valeur ajoutée. L'utilisateur doit, au contraire, être sollicité afin d'apporter sa compétence à la réalisation du service.

Dans le cadre de ces travaux de thèse, nous nous plaçons uniquement sur l'axe informa-

tique. Notre but est de voir s'il existe des méthodes qui permettent d'acquérir les connaissances sur les processus réalisés, s'il existe des techniques de recommandation qui se basent sur des processus, et s'il est possible de mettre en place une architecture logicielle qui combine les deux. Ainsi, nous cherchons à définir une nouvelle forme d'accompagnement dans l'exécution de processus en nous basant sur les techniques de fouille de processus et de recommandation.

1.3 Positionnement

Nous abordons la problématique de gestion des processus et de leur pilotage avec le point de vue original que sont les jeux sérieux. En effet, les jeux sérieux visent à développer des environnements où l'utilisateur est guidé mais dans lequel on lui laisse de la souplesse afin d'améliorer son engagement. Dans le domaine des jeux sérieux, et des jeux en général, des simulations d'environnements ont été développés avec une gestion du scénario permettant de s'assurer que l'apprenant suive un parcours tout en lui offrant la possibilité de construire sa propre expérience (En-Naimi and Zouhair, 2016; Bendahmane et al., 2016). Par exemple, les travaux de l'équipe MoCAH du Laboratoire LIP6 (Toussaint and Luengo, 2015) visent à suivre un apprenant durant son exécution du processus et à anticiper les erreurs qui pourraient être commises.

Dans Ho et al. (2018), les auteurs introduisent une méthode pour aider l'utilisateur à prendre la bonne décision en fonction des données extraites de leur modèle de données qui décrit les activités possibles à effectuer et leur impact sur l'atteinte de l'objectif. Cette méthode présente une liste des activités qui optimiseraient certains critères que prend en compte la méthode, comme le temps par exemple, mais ne présente que les actions suivant la dernière action effectuée. Or dans notre cas, n'ayant pas une vue globale, l'utilisateur peut se retrouver bloqué ou faire des erreurs sans savoir comment les corriger. Dans Wang (2014), l'auteur cherche à coordonner le processus collaboratif en notifiant les utilisateurs. Les utilisateurs ne sont donc pas accompagnés dans le processus mais sont plutôt notifiés du travail à effectuer.

La fouille de processus (*Process Mining*) vise à extraire des connaissances liées aux processus à partir des journaux d'événements (logs). Elle offre divers moyens pour découvrir (*Process Discovery*), vérifier la conformité (*Conformance Checking*) et améliorer (*Enhancement*) les processus réels (Van Der Aalst et al., 2015). Chaque événement se réfère à une instance de processus (qui laisse une trace) et une activité. Les événements sont ordonnés et des propriétés additionnelles comme l'horodatage et l'utilisateur (par exemple, la personne ou la machine qui réalise l'activité) peuvent être ajoutées. De nombreux travaux ont été réalisés afin de découvrir des modèles de processus pertinents à partir des logs. Diamantini et al. (2016), par exemple, ont proposé le *Behavioral Process Mining* qui est une approche qui sert à identifier les sous-processus significatifs à partir des processus non structurés. Cette approche est fondée sur l'utilisation d'algorithmes de clustering hiérarchique appliqués aux graphes fouillés. Leur

étude s'est appuyée sur des données réelles venant d'un CHU néerlandais. Ils ont identifié les traitements et les diagnostiques effectués les plus fréquents. De même, Song et al. (2008) présentent une approche générique qui utilise les techniques de classification. Leurs logs ont été divisés en sous-ensembles homogènes. Pour chaque sous-ensemble un modèle de processus a été créé. Ils ont introduit la notion de profil de logs afin de les comparer et de les caractériser.

Nous abordons donc la problématique de l'assistance de processus avec une approche gestion des exécutions dans les jeux. Nous cherchons à extraire des connaissances des exécutions passées afin de produire des recommandations. L'originalité de nos travaux réside dans le fait d'associer des algorithmes de fouille de processus avec des algorithmes de recommandation. Ainsi, comme illustré dans la figure 1.1, nos travaux de recherche sont en jonction de trois autres domaines : la gestion des traces, la recommandation et la gestion de processus.

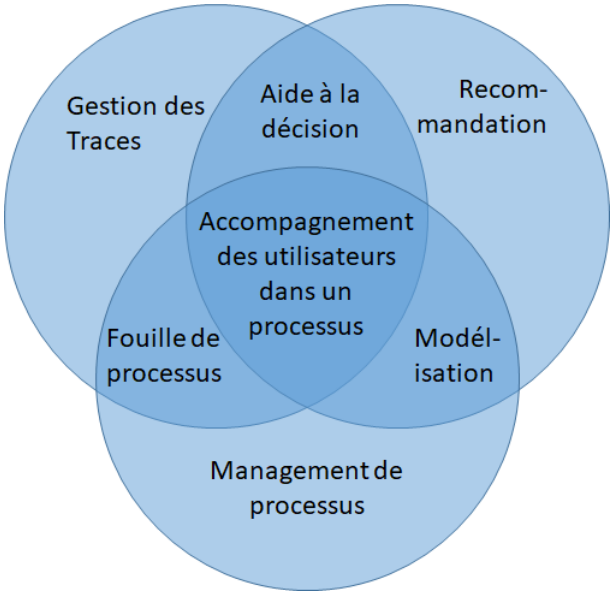

FIGURE 1.1 – Positionnement aux croisements des différents domaines de recherches

1.4 Questions de recherche

Notre ambition est de produire des recommandations à partir d'informations, extraites des traces, sur les processus métiers. Pour cela nous devons définir une méthodologie qui reposera sur l'exploitation des traces d'exécution, sur la fouille de processus et sur les méthodes de recommandation. Il est donc nécessaire de développer des expertises dans ces trois domaines.

Le questionnement de cette thèse porte sur la possibilité de définir une méthodologie de recommandation à partir des processus d'usage et de mettre en place l'architecture logicielle correspondante. Et, si cela est établi, est-il possible de fabriquer un compagnon pour aider à l'exécution de processus métier ?

Plusieurs verrous scientifiques sont à lever pour pouvoir répondre à cette question. Le premier porte sur l'analyse des traces. Est-il possible d'extraire de la connaissance sur les processus métier à partir des traces sur les systèmes que nous considérons ? C'est-à-dire dans l'exécution d'un processus où la conduite est faite par l'usager ou par composition de sous-processus avec une combinatoire telle qu'il est impossible d'avoir un modèle du processus *a priori*. Nous devons donc déterminer si les algorithmes utilisés dans le cadre de la fouille de processus sont pertinents pour notre cas d'étude.

Deuxièmement, nous devons produire une recommandation en cours d'exécution. Quelle forme doit prendre cette recommandation ? Est-il possible de produire une information pertinente dans un contexte d'exécution. C'est-à-dire dans un temps raisonnable pour permettre l'interaction avec l'utilisateur.

Finalement, est-il possible de mettre en place une architecture avec un système de trace pertinent pour permettre l'analyse des traces puis la recommandation ?

1.5 Démarche

Nos travaux de recherche se trouvent à l'interface entre plusieurs domaines. Afin de pouvoir nous positionner de manière pertinente, nous avons choisi une démarche de recherche qui s'appuie sur un cycle itératif. Après avoir analysé le domaine de la fouille de processus puis la recommandation, nous avons déduit qu'il fallait renforcer notre approche sur la collecte d'information. Ceci nous a conduit à mener des études sur les systèmes à base de traces.

Nous avons ensuite cherché à valider la continuité de notre approche sur un cas d'étude simple. Il s'agit de personnaliser le parcours d'un étudiant au cours de sa formation. Nous avons mis en place un démonstrateur qui permet, à partir du recueil des informations issues des promotions précédentes, d'extraire de la connaissance sur les parcours des étudiants et d'émettre des recommandations sur les suites du parcours pour un étudiant particulier. Cette étude nous a permis de mettre en place notre processus de recommandation de bout en bout et de proposer une première esquisse de notre architecture.

Nous avons ensuite cherché un cas d'étude plus complexe pour lequel il n'existe pas de processus *a priori*. Cette itération nous a permis de démontrer que notre architecture et notre méthodologie étaient robustes et qu'il était possible d'adapter la méthodologie à des cas particuliers.

1.6 Plan du manuscrit

Ce manuscrit de thèse est organisé en 7 chapitres. À la suite de cette introduction où nous exposons notre problématique, nos enjeux et notre volonté de résoudre notre problème en

combinant des approches issues des systèmes à base de traces, de la fouille de processus et de la recommandation, nous consacrons trois chapitre à l'état de l'art.

Le chapitre 2 porte sur l'analyse des traces. Après une présentation du concept de traces et des définitions associées, nous décrivons les fonctions que doit mettre en œuvre un système à base de traces. Nous terminons ce chapitre par une revue des systèmes à base de traces utilisés dans le contexte de la formation.

Le chapitre 3 aborde le domaine de la fouille de processus. Après avoir décrit la problématique et les enjeux, ce chapitre fournit les éléments de modélisations nécessaires pour appréhender les algorithmes présentés. Les grands principes des algorithmes de fouille de processus sont présentés, ainsi que les mesures couramment utilisées. Ce chapitre se poursuit par l'étude des limites de ces algorithmes, en regard du cas particulier que nous abordons, puis se termine par une description des algorithmes les plus populaires. La conclusion de ce chapitre nous permet de présenter l'algorithme de fouille de processus que nous retenons pour la suite de nos travaux.

Le chapitre 4 traite de la recommandation. Il se commence par une présentation des systèmes de recommandation et de leurs composants. Une classification des systèmes de recommandation est présentée. Puis nous décrivons les mesures de similarités et les problèmes rencontrés lors de la mise en place d'un système de recommandation. À la suite de cette étude nous en déduisons le système de recommandation que nous devons mettre en place pour notre étude.

Le chapitre suivant, chapitre 5, décrit la méthodologie que nous mettons en place pour superviser l'exécution des processus métier. Nous décrivons les trois étapes de la méthodologie : transformation des traces brutes en traces modélisées ; extraction des modèles de processus à l'aide de l'algorithme *inductive miner* ; recommandation basée sur la recherche de chemins optimaux. Nous validons notre approche sur un cas d'étude. Il s'agit d'analyser les parcours des étudiants de DUT Informatique à partir des traces recueillies sur sept années. Nous validons notre approche en nous servant de ces données pour construire notre base de connaissance et en simulant les recommandations sur la dernière année.

Le chapitre 6, permet d'éprouver la robustesse de notre méthodologie en abordant le cas d'un système où l'utilisateur conduit tout seul le processus métier. Nous lui offrons un ensemble d'activités possibles et un objectif et il doit construire son processus pour atteindre son but. Nous prenons comme exemple un jeu basé sur le jeu Tamagotchi car il est représentatif d'un comportement d'un apprenant qui doit mettre en place une stratégie d'apprentissage. Cette étude nous permet de décrire comment adapter notre méthodologie à ce cas d'étude.

Le chapitre 7 conclut ce manuscrit en présentant une synthèse des problèmes abordés et solutions proposées puis, après un bilan, présente les perspectives à donner à nos travaux.

CHAPITRE 2 | Analyse des traces

Sommaire

2.1 Introduction 10
2.2 Définitions 11
2.2.1 Trace 11
2.2.2 Trace numérique 11
2.2.3 Trace brute ou première 12
2.2.4 Source de traçage 12
2.2.5 Système à base de traces 12
2.3 Objectifs de l'utilisation des traces 12
2.3.1 Suivi de l'utilisateur 13
2.3.2 Partage d'expérience 14
2.4 Système à base de traces (SBT) 15
2.4.1 Trace modélisée 15
2.4.2 Principe d'un système à base de traces modélisées 17
2.5 Systèmes à base de traces existants 21
2.5.1 Streaming Application Trace Miner (SATM) 22
2.5.2 Collaboration Analysis Tool (ColAT) 22
2.5.3 Aplusix 23
2.5.4 Modeling USEs and Tasks for Tracing Experience (MUSETTE) . . . 23
2.5.5 Trace Analysis Tool for Interaction ANAlysis (TATIANA) 24
2.5.6 VISU 24
2.5.7 Système à base de traces pour le calcul d'indicateur dans Moodle (SBT-IM) 25
2.5.8 Projet d'Intégration de l'eXpérience pour l'Enseignement à Distance 25
2.5.9 Étude comparative des SBT présentés 26
2.6 Conclusion 27

2.1 Introduction

Lors de l'interaction entre un utilisateur et un environnement numérique, de nombreuses données sont générées. Ces données, appelées traces, peuvent être très abondantes et sont porteuses d'informations qui pourront être utiles pour la représentation et l'analyse de la logique d'exécution des activités d'un programme et des interactions avec l'utilisateur pris dans un contexte particulier. Le système à base de traces définit ainsi la façon d'observer, de collecter et d'analyser les traces pertinentes qui permettront, par exemple, de récupérer une représentation du processus ou de suivre l'exécution d'une application en conformité avec un contexte spécifique.

L'idée de conserver des enregistrements ou des opérations effectuées par un système informatique n'est pas nouvelle. Par exemple, les fichiers de log (*logfiles*) contiennent toutes les opérations effectuées lors de l'exécution d'un programme. Cela permet de disposer d'un outil qui facilite la compréhension des opérations effectuées et de constater *a posteriori* ce qui s'est passé. Il s'agit non seulement de comprendre les opérations effectuées par le système informatique mais aussi les interactions avec ses utilisateurs. Les travaux menés en IHM ont notamment permis de formaliser ces aspects (Kavalanekar et al., 2008).

Avec le temps, les traces, ou traces numériques, ont de plus en plus attiré l'intérêt et l'attention des chercheurs car l'idée de conserver les enregistrements des différentes opérations pour analyse et exploitation ultérieures est intéressante. Régulièrement, des documents référençant des cas concrets d'utilisation des traces dans le domaine de la sécurité, de l'information, du *e-Learning*, *etc.*, sont publiés et chacun d'eux a une définition et un usage propre des traces (Liu et al., 2018).

Bien utilisées, les traces permettent d'obtenir un regard sur les activités du système et sont donc souvent le meilleur point de vue que l'on puisse avoir *a posteriori*. Cela explique pourquoi elles sont le point central dans certains systèmes, allant d'une simple fenêtre de configuration d'une application jusqu'à l'ensemble des outils disponibles pour l'utilisateur à un instant donné (Wang, 2016). Elles permettent notamment de récupérer les informations de l'utilisateur et de les analyser afin de visualiser les interactions (Marty and Mille, 2009). Mais il est aussi possible de récupérer et d'agréger les informations de plusieurs utilisateurs afin d'obtenir des informations sur les masses (comme dans le *crowd-sensing* (Wirz et al., 2012), les interfaces utilisateurs intelligentes (Bigham et al., 2009), les systèmes de recommandation collaboratifs (Gras and Boyer, 2016), *etc.*).

Les traces sont par ailleurs le point central de certains systèmes (Settouti, 2011; Champin et al., 2003; Arana et al., 2004). Les méthodes de gestion des traces permettent de récupérer des données et des informations depuis les journaux d'événements des applications. Elles peuvent être des données textuelles sur l'état de l'application (Qiang, 2013), comme les fichiers de logs, et issues de plusieurs méthodes de collecte (Ho et al., 2018), comme la récupération

d'une trace avant et/ou après l'utilisation d'une activité.

L'objectif de ce chapitre est de décrire l'état des travaux de recherche dans le domaine des traces afin d'avoir un ensemble de données sur lequel baser nos travaux. En effet, nous visons à développer un cadre méthodologie pour définir un compagnon pour améliorer l'usage des processus métier. La première étape consiste à étudier comment collecter et traiter les traces d'exécutions pour pouvoir les utiliser par la suite. Nous commençons par une définition des traces de manière globale en section 2.2. Nous présentons ensuite les objectifs de l'utilisation des traces en section 2.3 puis traiterons des différentes méthodes de collecte, transformation et exploitation des traces dans la section 2.4. Nous comparerons les systèmes à base de traces existants en section 2.5.9 et enfin nous concluons cet état de l'art en section 2.6.

2.2 Définitions

Cette section commence par définir la notion de traces puis introduit différents concepts en lien tels que les traces brutes, les sources de traçage ou les systèmes à base de traces.

2.2.1 Trace

Le concept de *trace* apparaît dans différents contextes avec de nombreuses définitions. Dans la nature, une trace est une marque, une indication ou un objet qui démontre l'existence d'une activité passée. Selon le dictionnaire Larousse, une trace est « une suite d'empreintes ou de marques que laisse le passage d'un être ou d'un objet ; marque laissée par une action quelconque ; ce à quoi on reconnaît que quelque chose a existé ; ce qui subsiste d'une chose passée ». Le terme trace fait ainsi référence à plusieurs sens. Ainsi, selon le contexte le terme trace peut avoir un sens spécifique (Settouti et al., 2006). En littérature par exemple, une trace est ce qui subsiste de quelque chose du passé sous la forme de débris, de vestiges, *etc.*. En psychologie, la trace désigne ce qui subsiste dans la mémoire d'un événement passé.

Néanmoins nous pouvons donner une définition générale en accord avec toutes ces définitions. En effet, la trace est issue d'une interaction entre deux entités, tangibles ou non, par contact, physique ou non et dont la forme et la nature est fortement dépendante de son environnement.

2.2.2 Trace numérique

Par extension, la *trace numérique* dans les systèmes d'information est une marque d'activité ou d'interaction entre une entité et le système d'information, médiée par un dispositif informatique (Settouti et al., 2006). La trace numérique est intimement liée aux notions d'historique et de traçabilité, ce qui nous permet de dire que « la trace numérique est la trace de l'activité d'un utilisateur qui utilise un outil informatique pour mener à bien une activité,

s'inscrivant sur un support numérique » (Settouti et al., 2009). Par soucis de simplification, une trace désignera une trace numérique dans la suite du mémoire.

2.2.3 Trace brute ou première

Quand une trace est générée dans le système, suite à une activité ou une interaction, elle est considérée comme une *trace brute*. Une trace brute est la trace sans aucune interprétation ni transformation et qui représente la trace telle qu'elle a été collectée ou générée (Marty and Mille, 2009). Elle peut avoir un format défini par le modèle de la collecte des traces du système ou du composant considéré. On parle aussi de *trace première*. Les logs et les journaux d'événements systèmes sont des exemples de traces brutes. Elles sont souvent collectées depuis une source de traçage.

2.2.4 Source de traçage

Une *source de traces* ou *source de traçage* est une entité contenant et créant des traces conformément au modèle de collecte. Un fichier log d'une application ou un flux de données en sont des exemples, ayant de surcroît la particularité de fournir les traces en temps réel. Une des sources de traces classiques est l'Interface Homme-Machine (IHM) permettant à l'utilisateur d'interagir avec le système ou la machine (Dix, 2009) et, s'agissant d'interactions avec l'environnement, cela crée des traces, car toute interaction peut-être enregistrée ou *tracée*. Ces traces sont ensuite envoyées en entrée de l'outil ou le composant du système qui nous donne la possibilité de gérer ces informations en tant que telles au sein de l'environnement informatique. Ce composant est nommé *système à base de traces*.

2.2.5 Système à base de traces

Le système à base de traces (SBT) est une solution générique au problème de modélisation, de stockage et de manipulation des traces. Il décrit les données de la trace afin que cette dernière soit utilisable dans un contexte particulier de l'application considérée : définit son modèle de collecte, de représentation, de stockage, d'exploitation, *etc*. Les SBT sont présentés plus en détail dans la section 2.4.

2.3 Objectifs de l'utilisation des traces

Les traces peuvent être utilisées de différentes façons et avoir différents usages. Elles peuvent :

— rendre compte de l'activité du système et permettre de détecter des anomalies soit dans son exécution soit dans l'interaction avec les utilisateurs ou d'autres systèmes. Dans ce

cas, les données tracées servent à corriger les défaillances éventuelles et à améliorer le comportement du système ;

— permettre d'adapter l'exécution de l'application à l'utilisateur courant en fonction des expériences tracées des utilisateurs précédents. Ici, l'objectif est le partage d'expérience entre utilisateurs passés et présents ;

— permettre d'adapter l'exécution de l'application en fonction de la progression et du comportement de l'utilisateur lui-même. L'analyse des traces concerne alors un utilisateur unique dans le but de comprendre ce qu'il fait afin de lui proposer la meilleure ou la plus adéquate exécution possible ;

— permettre d'analyser les séquences d'activités des utilisateurs et identifier les comportements types. Ce qui permet notamment de recenser les procédures d'interaction ou d'activité et d'améliorer les processus métiers ;

— permettre d'analyser les activités et les interactions dans un but statistique ou comptable. L'objectif ici est de fournir aux administrateurs ou aux gestionnaires une vue analytique de ce qui s'est passé dans le système.

Nous abordons ci-après deux exemples de cas d'utilisation des traces ayant un rapport avec notre problématique : le suivi de l'utilisateur et le partage d'expérience.

2.3.1 Suivi de l'utilisateur

Selon (Loghin, 2008; ?), dans un environnement informatique pour l'apprentissage humain (EIAH), un enseignant cherche à reproduire les conditions auxquelles il est habitué (présentiel classique) et souhaite comprendre et rester au courant du déroulement de l'activité pédagogique mise en place afin de pouvoir réagir de différentes manières. Mais, il existe des difficultés pour l'enseignant de percevoir l'activité de chaque apprenant ou groupe d'apprenants :

— moins de retour sur le déroulement de la session que dans une situation d'apprentissage classique ;

— moins de retour sur la performance des apprenants (nombre d'exercices terminés, score, *etc.*) ;

— moins de retour sur le degré de collaboration entre les apprenants.

Le système doit fournir à l'enseignant des indicateurs d'un plus haut niveau d'abstraction qui l'aident à apprécier non seulement quantitativement mais surtout qualitativement le travail des étudiants pour adapter la session d'apprentissage.

C'est dans cette optique que s'inscrivent la plupart des systèmes de gestion de traces des EIAH. L'objectif consiste à accorder à tous les acteurs (y compris l'apprenant) la possibilité d'avoir un meilleur retour dans leurs expériences de formation ou d'apprentissage.

2.3.2 Partage d'expérience

Les expériences constituent une source de connaissances qui pourrait aider à apprendre à partir des solutions qui ont fonctionné dans des contextes définis. En nous basant sur ces expériences, appelées expériences partagées, nous pouvons renforcer le transfert de compétences, améliorer les performances et faciliter l'extraction des connaissances à partir des cas vécus. Ces expériences sont les traces générées de plusieurs utilisateurs différents. L'objectif principal de cette utilisation des traces est de proposer des modèles et des outils permettant au système de transformer les traces partagées par un utilisateur source u_1 pour qu'elles soient adaptées aux utilisateurs cibles u_2 et u_3 quels que soient leurs profils et besoins spécifiques, comme illustré dans la Figure 2.1 (Sehaba, 2014). Un profil d'utilisateur décrit l'ensemble des informations d'un utilisateur. Elles peuvent être les capacités, les compétences, les préférences, *etc*.

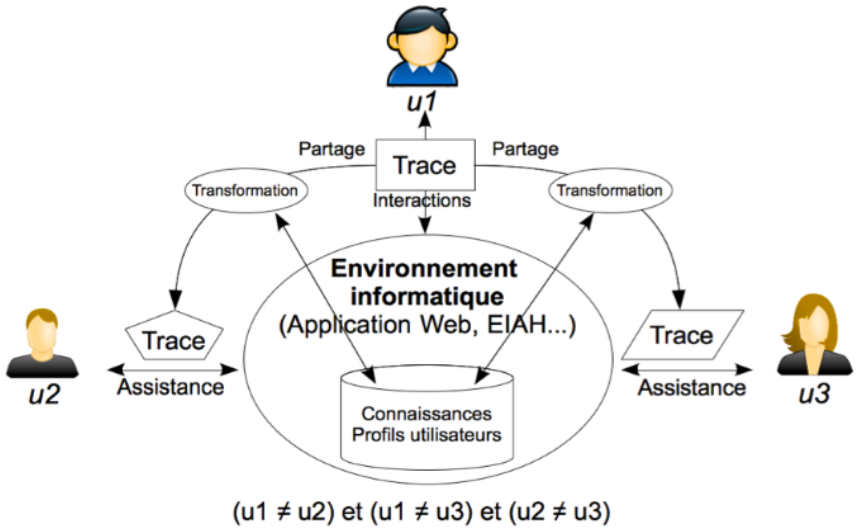

FIGURE 2.1 – Partage d'expériences entre utilisateurs de profils différents (Sehaba, 2014)

Le partage d'expérience peut intervenir dans les cas suivants :

— l'adaptation du contenu : elle consiste à ajouter, supprimer ou modifier les éléments de la trace ;

— l'adaptation de présentation : elle concerne l'affichage des traces et ses propriétés. Elle choisit la méthode de visualisation de la trace la plus convenable en fonction de ses propriétés (Clauzel et al., 2009; Bétrancourt et al., 2011) ou du profil de l'utilisateur ;

— l'adaptation des modalités d'interaction : elle se concentre sur la façon de faire pour réaliser les tâches. La plupart des travaux se focalisent sur ce type d'adaptation.

2.4 Système à base de traces (SBT)

Afin d'utiliser les traces dans notre approche d'accompagnement de processus, nous nous sommes basé sur les systèmes informatiques permettant et facilitant l'exploitation des traces (Settouti et al., 2009). Le SBT est une solution générique au problème de modélisation et de manipulation des traces. Afin de définir de tels systèmes, nous définissons et formalisons tout d'abord le concept de modèle de traces puis nous présentons le principe du système à base de traces modélisées en précisant les composants génériques des SBT.

2.4.1 Trace modélisée

Une trace numérique dans un système informatique respecte un modèle de trace décrivant les objets qui en font partie. Ce modèle pourra représenter de manière formelle et abstraite les éléments constitutifs d'une trace (Ho et al., 2018). Chaque trace peut être associée à un modèle de trace dont la définition est : « Un modèle de trace est un modèle représentant la trace d'une manière formelle. Il contient notamment les propriétés et attributs de la trace : date et heure, identifiant d'utilisateur, action réalisée, *etc*. »

L'objectif d'un modèle de traces est d'expliciter et de décrire de façon abstraite les objets qui font partie de la trace (Loghin, 2008; Champin et al., 2013). Par exemple, dans le cas des fichiers de journalisation du serveur Apache, le modèle de trace est *Common LogFile Format* [1] explicitant la sémantique de la trace.

Une *trace modélisée* ou *m-trace* est une trace qui est associée à un modèle de traces (Settouti, 2011). Une m-trace est toujours associée à un modèle de trace définissant les éléments qui la composent : observés et relations.

Un modèle de traces permet donc de définir les données tracées, ou enregistrées, via ses attributs. Il est ainsi possible d'identifier les informations nécessaires à trouver dans les données collectées ou les traces brutes. Cependant, un modèle de traces ne permet pas de définir quelles données seront effectivement collectées, il impose les contraintes que les traces brutes peuvent ou non respecter. Si c'est le cas, alors les systèmes se basant sur ce modèle respecteront le modèle de traces défini.

2.4.1.1 Application d'un modèle de traces

Pour appliquer un modèle de traces et obtenir des m-traces conformes au modèle choisi à partir des traces premières, il est possible d'utiliser des outils comme des extracto-chargeurs de

1. Format standardisé pour la représentation des traces des principaux serveurs web (`https://www.w3.org/Daemon/User/Config/Logging.html`)

type ETL [2] (Kimball and Caserta, 2011) ou des API [3] de modélisation propres au modèle choisi comme l'est XESame pour le modèle d'événements (Verbeek et al., 2010), abordé ci-après.

Toutefois, avant de modéliser des traces, il est nécessaire d'identifier quels sont les besoins du cas considéré. Par exemple, avec la fouille de processus, il est nécessaire d'observer au moins l'activité effectuée et l'exécution à laquelle elle appartient (Van Der Aalst et al., 2003). À notre connaissance, il n'existe que deux modèles de traces capables de définir les traces pour la fouille de processus : le modèle d'événements (Van Der Aalst and Weijters, 2004) et le modèle de traces numériques (Cordier et al., 2013).

Le modèle de traces numériques est un modèle plus général car il ne représente pas spécifiquement un type de trace. Il permet d'associer des attributs, qui doivent être définis par l'utilisateur du modèle, afin d'étiqueter les valeurs qui pourront être utilisées plus tard (Cordier et al., 2013).

TABLE 2.1 – Comparaison des modèles de traces

	Capable de contraindre les traces qui représentent les activités effectuées durant l'exécution d'un processus	**Adapté aux algorithmes de récupération de l'organisation des activités du processus**
Modèle de traces numériques	Oui	Non
Modèle d'événement	Oui	Oui

Comme résumé dans la table 2.1, le modèle de traces numériques est capable de représenter les traces des événements tels que l'exécution d'une activité, mais il est aussi nécessaire que les algorithmes, logiciels et méthodes qui utiliseront les traces soient compatibles avec le modèle choisi. Ce qui n'est pas le cas pour ce dernier. La majorité des logiciels et algorithmes permettant de récupérer une représentation de l'organisation d'un processus se basent plutôt sur le modèle d'événements.

2.4.1.2 Format des traces

Que ce soit la trace première ou la trace transformée (m-trace), elle respecte un schéma de données prédéfini dépendant du modèle considéré. Pour la trace première, ce schéma est dicté par le modèle de trace de la collecte et pour la m-trace le schéma est défini par le modèle des traces issues de la transformation. La trace résultante est classiquement un enregistrement

2. Un ETL (*Extract-Transform-Load*) est un module logiciel permettant d'effectuer des synchronisations d'information d'une source de données vers une autre.

3. Une API (A*pplication Programming Interface*) est une bibliothèque de fonctionnalités par laquelle un logiciel offre des services à d'autres logiciels.

de plusieurs champs sauvegardés dans une table unique ayant une colonne pour chacun des champs de l'enregistrement complet. Ces champs représentent les états des différentes informations observées, ou transformées, spécifiées par le modèle correspondant. Il est à noter que le stockage de la trace n'est pas obligatoirement assuré par une base de données dédiée. Souvent, notamment pour les traces premières, le stockage est fait dans des fichiers textes (ou fichiers de logs) où les différentes données (champs) d'un enregistrement sont séparées par un caractère de séparation spécifique tel qu'un espace, une tabulation ou une virgule (cas des fichiers CSV).

Cependant, des normes destinées à la structuration des traces commencent à émerger. Par exemple, le standard XES à base d'XML a été défini pour faciliter la structuration et la portabilité des traces [4]. Il a été développé pour stocker les flux de traces et faciliter la communication notamment entre les outils de création et les outils de recherche destinés à la fouille de processus.

2.4.2 Principe d'un système à base de traces modélisées

Un système à base de traces modélisées est tout système à base de traces dont le fonctionnement implique, à des degrés divers, la gestion, la transformation et la visualisation de traces modélisées explicitement (Settouti, 2011).

La Figure 2.2 décrit le principe général d'un système à base de traces modélisées. Il existe différents types de systèmes à base de traces adoptant des approches variées dans leur modèle de fonctionnement concernant par exemple la définition des sources de traçage ou la récupération des traces. Toutefois, nous retrouvons systématiquement plusieurs fonctionnalités principales que l'on peut répartir en trois phases et qui seront assurées par trois composants distincts du SBT.

La collecte de traces concerne :

— la définition des sources de traçages dans une application interactive ;

— la définition du modèle de traces ;

— la récupération des traces.

La transformation de traces assure :

— la définition des modèles de transformation ;

— la transformation des traces obtenues de la phase de collecte.

L'analyse de traces effectue :

— l'analyse et l'exploitation des traces selon les buts d'utilisation définis.

Nous décrivons, ci-après, plus en détails ces trois phases.

4. eXtensible Event Stream : standard IEEE pour l'interopérabilité dans les journaux d'événements et les flux d'événements `http://www.xes-standard.org`

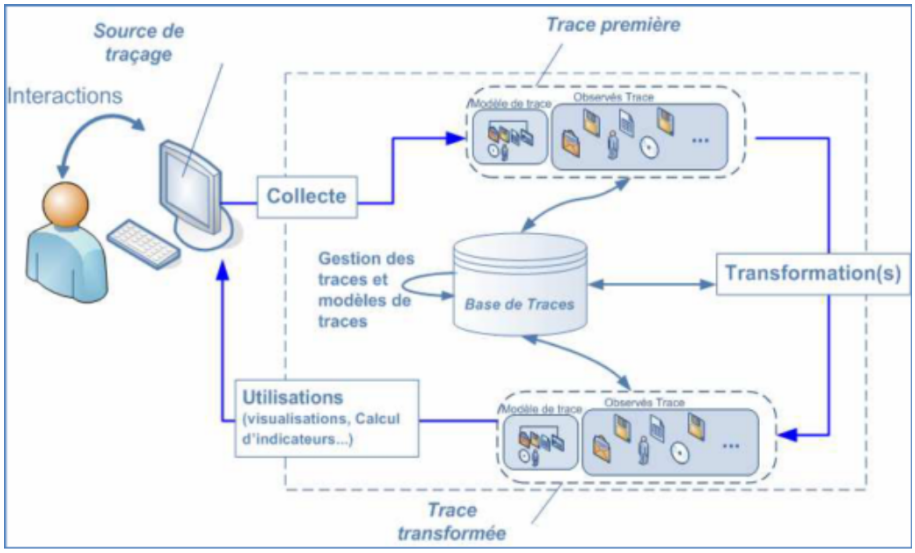

FIGURE 2.2 – Principe général d'un système à base de traces modélisées (Loghin, 2008)

2.4.2.1 Collecte de traces

La collecte des traces permet l'observation de l'utilisation d'un système à partir d'une ou de plusieurs sources de traçage. Elle sert à convertir ou à transformer des informations générées par l'interaction utilisateur/système pendant l'activité en une trace initiale, la trace première.

Une source de traçage est un fichier ou un flux de données dans un format explicite. Nous dirons que ce sont des ressources disponibles à tout moment dans le système à base de traces. Plusieurs sources de traçage, éventuellement différentes, peuvent être utilisées pour récupérer des traces (Settouti et al., 2006).

Un protocole de collecte de traces modélisées consiste à exploiter de façon manuelle ou automatique un ensemble de sources de traçage pour construire une trace modélisée. Pour cela, nous avons besoin de construire un ensemble d'*observés* associés à un modèle de traces défini ; les observés étant des données relatives à une observation faite d'une activité. Une trace modélisée issue du processus de collecte est appelée trace modélisée première ou m-trace première.

Il y a deux techniques de traçage :

— approche *ad-hoc* qui consiste à tracer tous les observés (l'ensemble des traces enregistrables) ce qui conduit généralement à une grande quantité de traces non directement interprétables par un utilisateur humain (par exemple, les traces collectées par un serveur web) ;

— approche par instrumentation de l'environnement de travail, qui consiste à collecter les traces par rapport à un objectif d'observation et qui permet de tracer un ensemble restreint d'observés jugé pertinent pendant la session d'observation.

Les traces premières, collectées depuis les sources de traçage, sont stockées dans la base

de traces modélisées. Cette base est l'ensemble des traces modélisées ou les m-traces qui sont manipulées par le système à base de m-traces.

2.4.2.2 Transformation des traces

La trace obtenue à l'issue de la phase de collecte n'est pas toujours exploitable directement. Il faut ensuite passer par une phase de transformation sur la trace en entrée dont le résultat est une nouvelle trace appelée trace transformée.

Une trace transformée est une trace obtenue après la transformation selon des intentions d'analyse particulière (Marty and Mille, 2009). Elle sert à assurer la capacité d'interprétation des traces recueillies. Cela permet à l'utilisateur de comprendre les éléments obtenus et permet de faciliter l'analyse de ces éléments selon des besoins différents. Les m-traces premières sont les seules m-traces non transformées d'un SBT. Par exemple, imaginons un système qui utilise des traces décrivant les activités effectuées lors des exécutions d'un processus, appelées traces d'activités. Si le système reçoit en entrée des traces d'IHM, il devra utiliser une étape de transformation pour faire passer les traces premières (les traces d'IHM) en traces transformées (les traces d'activités). Il existe deux catégories de transformations qui sont prises en charges par les SBT : la transformation manuelle et la transformation automatique utilisant des modèles de transformation.

Une transformation manuelle désigne toute création directe par l'utilisateur d'une trace à partir d'une trace existante. Ce type de transformation est réalisé par un utilisateur interagissant avec sa trace pour la mettre à jour ainsi que son modèle. Le SBT permet de garder la cohérence entre les traces et leurs modèles lors d'une transformation manuelle.

De plus, en utilisant certains modèles comme les modèles comportementaux (Muratet et al., 2018; Yessad et al., 2017), il est possible de comprendre les activités sous-jacentes aux interactions effectuées avec la machine (Mbatchou et al., 2018).

La transformation automatique est employée dans un SBT en définissant les modèles de transformation. Un modèle de transformation est l'ensemble des règles permettant de sélectionner ou de réécrire des motifs (un motif représente la séquence ou un sous ensemble d'observés de la trace). Nous pouvons distinguer trois types de transformations automatiques comme la montre la Figure 2.3 :

— Type de sélection : c'est la création d'une nouvelle trace contenant tous les observés selon un filtre donné. Ce type de transformation permet de séparer les observés pertinents de la trace et ceux qui sont considérés comme du bruit. Les filtres sont des contraintes sur les objets de la trace, le domaine temporel et les relations structurelles ;

— Type de réécriture de motifs : permet de remplacer un motif d'observés en un autre observé. Ces motifs peuvent être constitués d'observés qui ne se suivent pas directement. La transformation a pour but de réécrire un motif représenté par une succession d'obser-

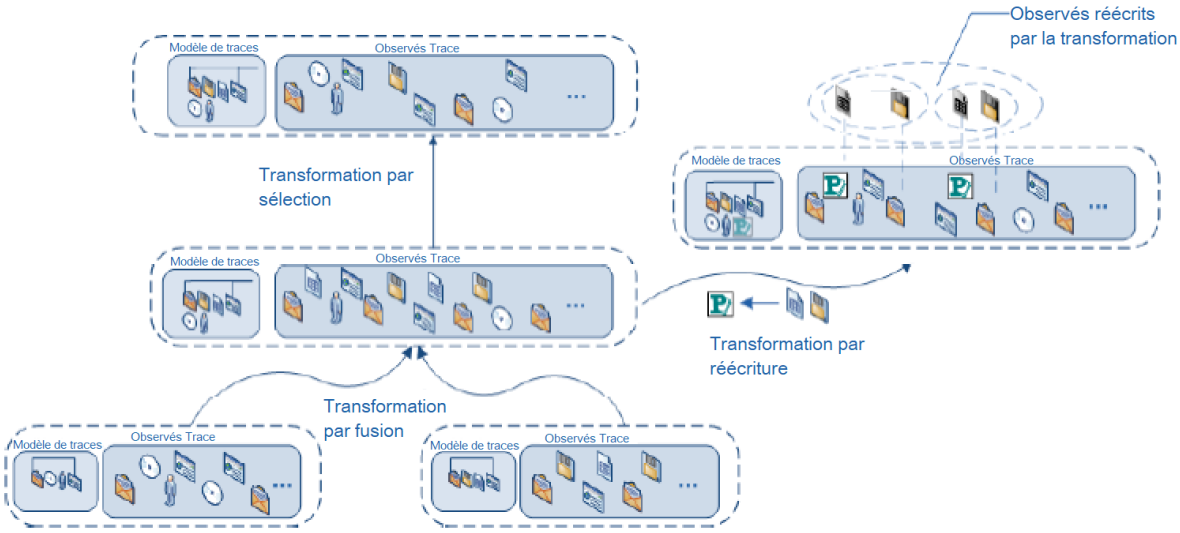

FIGURE 2.3 – Transformation de traces (Settouti et al., 2009)

vés ;

— Type de fusion : selon les domaines temporels, elle consiste à fusionner les observés de plusieurs traces en une seule trace en respectant la temporalité. Le modèle des traces issue de cette transformation est la fusion des modèles des traces qui ont été transformés. Ce type de transformation est possible à appliquer si les traces sont des extensions temporelles homogènes (permettant des translations et des conversions de temps).

Parmi les mécanismes permettant la transformation des traces, on peut citer le principe ETL (mentionné dans le paragraphe 2.4.1) du domaine des entrepôts de données (Kimball and Caserta, 2011) ou les modèles comportementaux (Downey et al., 2007; Muratet et al., 2018) du domaine des sciences humaines :

— un ETL consiste en un programme qui récupère des informations sous une forme donnée (modèle d'extraction) et les restitue sous une autre forme (modèle de transformation). Les données transformées sont réinjectées dans le système considéré (en général, sauvegardées dans une base dédiée pour une exploitation ultérieure) ;

— un modèle comportemental est un ensemble de relations entre les interactions d'un utilisateur et les activités sous-jacentes. Il permet de transformer les traces d'IHM en traces décrivant les activités effectuées en utilisant les observés des traces comme l'action effectuée (un clic ou un mouvement de souris) (Song et al., 2008; Han et al., 2007; Muratet et al., 2018).

2.4.2.3 Exploitation des traces

Une manière classique d'exploiter les traces est leur visualisation. Les traces obtenues peuvent être visualisées telles quelles ou bien servir à l'élaboration d'indicateurs plus complexes.

Le système de visualisation est l'ensemble des techniques de présentation des traces. La visualisation nécessite quelques transformations :

— extraction et sélection des données à partir des sources de traçage ;

— transformation pour en ressortir un modèle d'analyse comme une représentation de l'organisation des activités dans le processus ;

— transformation sous forme de primitives graphiques qui seront les briques pour présenter une visualisation des données aux utilisateurs ;

— génération de la visualisation graphique à partir des primitives.

Un indicateur est une variable à laquelle est attribuée une série de caractéristiques. Les valeurs de l'indicateur peuvent prendre des formes numériques, alphanumériques ou même graphiques. La valeur possède un statut, c'est-à-dire qu'elle peut être brute (sans unité définie), calibrée ou interprétée (Sehaba, 2014). Les indicateurs facilitent les adaptations à réaliser dans l'environnement numérique. L'obtention des indicateurs n'est possible que grâce à une meilleure analyse des traces.

Cependant, la visualisation n'est pas la seule manière d'exploiter les traces. Les techniques de fouille de données représentent des approches qui peuvent être utilisées pour l'analyse des traces numériques collectées. Elles sont appropriées pour l'analyse des grandes quantités de traces numériques. Dans ce cas, l'analyse des traces présente trois parties principales : le prétraitement des traces, la découverte des *patterns* (ou motifs) d'utilisation intéressants et l'analyse de ces *patterns*. En effet, l'approche de la fouille de données représente une alternative particulière et intéressante pour l'analyse des traces d'utilisation (Loghin, 2008).

La phase d'exploitation de traces nous permet de les utiliser comme une source d'informations abondante. Nous allons décrire quelques systèmes à base de traces existants pour identifier si l'un d'entre eux permet de traiter les traces représentant les événements de l'exécution d'un processus. Cela nous permettra de faire ressortir les caractéristiques nécessaires pour les systèmes à base de traces destinés à l'analyse des activités d'un utilisateur en vue de définir un compagnon pour l'exécution de processus métiers.

2.5 Systèmes à base de traces existants

Nous présentons ci-après quelques systèmes à base de traces existants. Notre objectif étant de définir un compagnon pour l'exécution de processus métiers, nous orientons notre étude sur

les systèmes à bases de traces utilisés pour l'apprentissage. Ceux-ci nous permettront d'avoir une vue globale des systèmes étudiés. Nous les comparons selon des critères en lien avec nos besoins pour la fouille de processus (qui seront présentés dans le chapitre 3).

2.5.1 Streaming Application Trace Miner (SATM)

SATM est un logiciel créé en 2015 par Oleg Iegorov au LIG de Grenoble pour l'entreprise STMicroelectronics qui en détient maintenant les droits (Iegorov, 2015). Ce système fait une analyse des traces d'exécution via l'utilisation de méthodes de fouille de données et d'approche statistique pour le débogage des flux de données des systèmes électroniques embarqués. Cela permet d'appliquer un aspect temporel au problème pour plus facilement détecter les anomalies comme les situations d'inter-blocage et les boucles infinies conditionnelles. Pour y parvenir, ce système utilise le *sequence mining*, une branche du domaine de la fouille de données, avec le *minimal contrast sequence mining algorithm* pour la détection des motifs recherchés. Néanmoins, cet algorithme utilise le temps pour ses analyses statistiques et détection de motifs, car il travaille avec des traces provenant uniquement de sources informatiques ou mécaniques. Cette approche est donc très intéressante mais difficilement utilisable dans notre contexte car la dimension temporelle dans les traces d'exécution des processus n'est pas obligatoire. De plus, cet algorithme recherche des motifs prédéfinis tandis que nous cherchons à modéliser un algorithme. De plus, son modèle de traces prend en compte des traces provenant uniquement de sources informatiques ou mécaniques qui n'acceptent pas complexité des activités des processus métier comme les activité du processus d'apprentissage. Pour finir, les algorithmes de *sequence mining* ne permettent pas la détection de certains motifs des processus métier, comme les motifs décrit dans la section 3.2.3.

2.5.2 Collaboration Analysis Tool (ColAT)

ColAT est un outil d'analyse de traces d'apprentissage indépendant de tout système d'apprentissage (Nikolaos et al., 2005). La nouvelle version de ColAT, ActivityLens, est un environnement développé par *Human-Computer Interaction Research Group* (HCI Group) de l'Université de Patras pour soutenir la recherche ethnographique et pour faciliter l'analyse des données produites à partir des études d'évaluation d'utilisabilité. ActivityLens est un outil d'analyse basé sur l'activité, spécialement conçu pour les chercheurs et les analystes qui sont impliqués dans les études ethnographiques. Son but est de les aider dans leur analyse et permet d'étudier les données provenant de sources multiples recueillies au cours de ces études. L'utilisation intensive du multimédia permet aux chercheurs de travailler avec des fichiers textes, graphiques, sons, vidéos, mais aussi des fichiers de log qui sont produits à partir de différentes études. Il offre une vaste gamme de fonctionnalités pour l'observation et l'analyse des données recueillies. Ainsi, ActivityLens permet une manipulation facile et une navigation dans les

sources multimédias, l'annotation des données, l'analyse statistique, le filtrage des données, la visualisation de données annotées, la génération de rapports, *etc*. Ne prenant pas en compte l'aspect processus, ce SBT n'est pas utilisable.

2.5.3 Aplusix

Aplusix, et maintenant Aplusix Neo, est un environnement d'aide à l'apprentissage de l'algèbre qui utilise les traces d'apprentissage pour aider les élèves à résoudre des exercices de calcul numérique (Nicaud, 1987). C'est un logiciel développé au LIG de Grenoble pour aider les élèves à apprendre l'arithmétique et l'algèbre. Il permet de renforcer les compétences des élèves, en diminuant leurs erreurs de calcul, et montre comment résoudre les exercices aux élèves qui en ont besoin. L'Aplusix permet d'enregistrer les productions des élèves, comme celles obtenues dans l'environnement classique papier, mais comportant des informations additionnelles, comme le temps, les hésitations, les corrections, *etc*. Ce travail produit une trace brute qui représente une modélisation comportementale des élèves. Le logiciel procède à une restriction, un découpage et une interprétation des traces brutes obtenues lors des anciennes manipulations des élèves et produit une trace enrichie contenant des règles algébriques expliquant les transformations des élèves et identifiant le contexte où elles apparaissent. Par ailleurs, Aplusix présente une modélisation des connaissances des élèves qui s'est appuyée sur la confrontation des analyses manuelles et automatiques d'expérimentions ayant eu lieu dans des établissements scolaires de différents pays (Chaachoua et al., 2007). Ne prenant pas en compte l'aspect processus, ce SBT n'est pas utilisable.

2.5.4 Modeling USEs and Tasks for Tracing Experience (MUSETTE)

MUSETTE est une approche générale de modélisation basée sur K-TBS (Zarka et al., 2013). Elle peut être implémentée grâce à divers langages ou formalismes de représentation (Champin et al., 2003). C'est un outil permettant et facilitant la manipulation des traces. Dans le cadre des recherches sur le traçage des interactions, le modèle MUSETTE a été développé dans le but de modéliser l'utilisation d'un système par un (ou des) utilisateur(s), à partir des traces, dans un objectif de réutilisation des traces. L'approche permet de modéliser l'expérience de l'utilisateur dans l'optique de réutiliser cette expérience comme une source de connaissances. Elle est basée sur le traçage des interactions, conformément à un modèle d'utilisation du système qui décrit les objets et les relations manipulés par l'utilisateur. Les traces peuvent être visualisées, intégrées à une base de connaissances non-orientée tâche, et peuvent être ensuite analysées grâce à des signatures de tâches. MUSETTE vise donc à représenter l'expérience concrète, en lien avec son contexte d'utilisation. La mise en œuvre de l'approche MUSETTE est développée et diversifiée dans les travaux de l'équipe CEXAS du LIRIS. Certains travaux s'attachent par exemple à étudier les possibilités de MUSETTE appliquée à de

multiples utilisateurs et au partage et à la réutilisation d'ontologies (approche multi-agents de MUSETTE, modèle MAZETTE) (Arana et al., 2004). D'autres travaux, s'appuyant également sur l'approche MUSETTE, visent l'étude des processus méta cognitifs dans le cadre de l'apprentissage (projet Clever@) ou l'assistance à la réutilisation de l'expérience dans un contexte coopératif de conception. Enfin, une approche dénommée MUSETTE-Analyse vise non plus à assister directement l'utilisateur, mais à fournir des outils à des analystes d'activités tels que la conduite automobile (projet INRETS), ou bien la manipulation d'outils informatiques pour les personnes âgées (projet MNESIS).

2.5.5 Trace Analysis Tool for Interaction ANAlysis (TATIANA)

TATIANA est un système qui sert aux chercheurs pour analyser des traces d'interactions (Dyke et al., 2010). C'est un logiciel orienté vers l'analyse de séances de travail collaboratif. C'est un environnement logiciel basé sur ce modèle et conçu pour aider les chercheurs à gérer, synchroniser, visualiser et analyser leurs données au travers de la création itérative d'artefacts qui leur permettent d'améliorer ou de concrétiser la compréhension qu'ils ont de leurs données. La figure 2.4 présente le schéma fonctionnel de TATIANA.

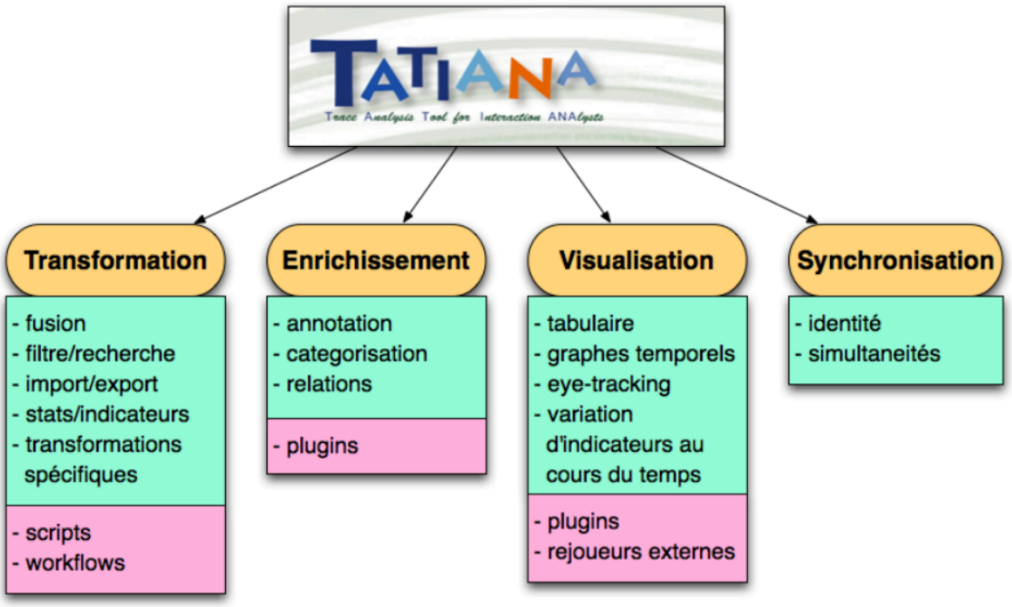

FIGURE 2.4 – Schéma fonctionnel TATIANA (Dyke et al., 2010)

2.5.6 VISU

VISU est une plate-forme de visioconférence à base de traces pour le tutorat synchrone (Clauzel et al., 2009; Bétrancourt et al., 2011). VISU intègre directement dans ses interfaces

les éléments de visualisation et de manipulation de traces. Il s'appuie fortement sur les traces modélisées pour représenter et enrichir les interactions homme-machine. VISU est un logiciel issu du projet ANR Ithaca [5]. Il permet aux tuteurs de préparer une séance, de la mener tout en posant des marqueurs. Il permet aux tuteurs et aux apprenants de revenir sur celle-ci *a posteriori* et d'en construire des bilans. VISU a été conçu, dans un premier temps, pour répondre à des besoins spécifiques de l'enseignement/apprentissage des langues en ligne. Par rapport à l'existant, VISU permet aux enseignants de disposer de trois fonctionnalités principales pendant la séance qui sont :

— un assistant de préparation qui permet de rentrer en amont de la séance des consignes, des mots-clés, des documents iconographiques ou vidéo, afin d'organiser l'activité pédagogique et rendre l'interaction plus facile pour les enseignants qui pourront rendre visibles ces éléments au moment choisi à leurs étudiants distants ;

— une zone de communication centrale qui concentre toutes les interactions, écrites, visuelles et orales et facilite le travail d'apprentissage en ligne ;

— une zone de suivi qui permet aux enseignants (et aux étudiants) de poser des marqueurs pour gérer le *feedback* apporté aux étudiants (précisions, corrections, *etc*.) sans perturber l'interaction orale qui reste la priorité principale.

Ne prenant pas en compte l'aspect processus, ce SBT n'est pas utilisable.

2.5.7 Système à base de traces pour le calcul d'indicateur dans Moodle (SBT-IM)

SBT-IM est un système à base de traces pour le calcul d'indicateur dans Moodle développé par le laboratoire LIRIS (Djouad et al., 2011). Il est composé d'un système de collecte de traces, un système de transformation des traces, un système de visualisation et un système de calcul d'indicateurs à base des traces. C'est un système à base de traces construit pour s'intégrer à la plateforme Moodle et respecte bien le principe général d'un SBT présenté dans la section 2.4. Ne prenant pas en compte l'aspect processus, ce SBT n'est pas utilisable.

2.5.8 Projet d'Intégration de l'eXpérience pour l'Enseignement à Distance

PIXED est un EIAH créé en 2002 intégrant un SBT (Heraud, 2002). Il donne un bon exemple de SBT pour les EIAH en utilisant un modèle de traces modélisées en 4-uplet avec :

— un ensemble de classes d'objets ou types d'objets décrivant les objets observés de la trace ;

5. `https://liris.cnrs.fr/ithaca/`

— un ensemble fini de types de relations entre classes ;

— un domaine temporel qui est l'ensemble fini des intervalles de temps apparaissant dans les traces ;

— un ensemble fini des éléments de la trace où, par exemple, un élément peut être une valeur d'une ligne dans le format CSV.

Le système PIXED propose aux apprenants consultant un cours en ligne de réutiliser le parcours d'apprentissage d'autres apprenants. PIXED permet de renvoyer à l'apprenti une possibilité d'exploiter des traces d'apprentissages comme source de connaissances pour assister son orientation dans la progression d'apprentissage dans un cours. La trace est considérée comme un cas composé d'une séquence d'actions que l'on peut interpréter selon des contextes différents. Une instanciation du modèle de trace du SBT Pixed permet de donner un ensemble composé de notions, activités éducatives, QCM, *etc*. Les observations représentent les données de la trace comme étant des instances de classe comme notions (observé, tableaux, *etc.*). Chaque observation est lié à un intervalle de temps appartenant au domaine temporel. Les relations entre observations sont contraintes par la notion/activité effectuée.

2.5.9 Étude comparative des SBT présentés

Le tableau 2.2 résume les propriétés recherchées chez les différents systèmes à base de traces présentés dans les paragraphes précédents.

La première colonne « Collecte de traces » définit si le SBT est capable de récupérer les traces depuis une source de traçage. La deuxième colonne décrit si le SBT est capable d'utiliser un modèle tel que le modèle d'événement ou le modèle de traces numériques (*cf.* paragraphe 2.4.1.1). La troisième colonne détermine si le SBT utilise des méthodes de transformation (algorithme ou logiciel) capables de récupérer une représentation de l'organisation du processus. La dernière colonne présente le domaine d'application du système à base de traces.

Il est possible d'observer que tous les systèmes à base de traces sont capables de collecter les données et que ColAT est capable de les collecter depuis plusieurs sources et même des sources hétérogènes. Cependant, la transformation pose problème car un seul SBT est capable de modéliser les traces afin qu'elles représentent les activités des exécutions d'un processus, il s'agit de MUSETTE. Au regard du troisième point, on constate qu'aucun système à base de traces n'est capable d'utiliser un modèle de traces compatible avec les logiciels ou algorithmes de fouille de processus.

Ainsi, il nous sera nécessaire de définir notre propre méthode de gestion des traces, basée sur les méthodes de gestion des traces similaires au SBT MUSETTE. Il est le plus proche de ce que l'on cherche à obtenir grâce à son utilisation du modèle de traces numériques. De plus, comme il s'agit d'un système complet et non d'une API, comme par exemple pour SBT-IM,

TABLE 2.2 – Comparaison des SBT

	Collecter les traces	Transformer les traces dans un modèle qui représente des exécutions d'un processus	Être adapté aux algorithmes de récupération de l'organisation des activités du processus	Domaine appliqué
SATM	Oui	Non	Non	Débogage des applications de streaming via les traces d'exécutions
COIAT	Collecte depuis de multiples sources	Non	Non	Recherche ethnographique
APLUSIX	Oui	Non	Non	Éducation (aide à l'apprentissage de l'algèbre)
MUSETTE (kTBS)	Oui	Oui, en traces numériques	Non	Approche générale, pouvoir développer et diversifier
TATIANA	Oui	Non	Non	Chercheurs dans une séance de travail collaboratif
VISU	Oui	Non	Non	Éducation (répondre aux besoins de l'enseignement apprentissage des langues en ligne)
SBT-IM	Oui	Non	Non	Moodle
PIXED	Oui	Non	Non	Éducation (apprentissage à distance)

nous ne pouvons pas le réutiliser dans nos travaux et devrons implémenter une approche de gestion des traces dédiée à nos besoin.

2.6 Conclusion

Nous venons de présenter un état de l'art sur les travaux dans le domaine des traces. Un système à base de traces définit la façon d'observer, de colleter et d'analyser les traces d'interactions entre un utilisateur et un environnement numérique. Avec l'explosion du « *Big Data* » les traces deviennent un enjeu stratégique et sont une source de connaissances importante.

Nous avons présenté les objectifs de l'utilisation des traces. Pour suivre les activités d'un utilisateur et en extraire des connaissances sur le comportement de l'utilisateur et définir ainsi son profil. Nous avons ensuite présenté les composants d'un système à base de traces : la collecte ; la transformation ; et l'exploitation. La collecte consiste à définir les « sondes » qui vont permettre recueillir de l'information. La phase de transformation correspond ensuite à modifier les informations collectées en données d'entrée pour la phase d'exploitation. Dans le

cadre de cette thèse, la phase d'exploitation est celle pendant laquelle nous allons extraire les processus des utilisateurs. Pour cela nous utiliserons les algorithmes définis dans le cadre de la fouille de processus que nous présentons dans le chapitre suivant (3).

Nous avons ensuite comparé des systèmes à base de traces existants. Nous nous sommes concentrés plus particulièrement sur les systèmes déployés dans le contexte de l'apprentissage car il s'agit des cas d'études que nous privilégions (qu'il s'agisse du parcours de l'apprenant ou pour construire sa connaissance dans un jeu sérieux). Nous avons conclu cette étude par un tableau qui synthétise les résultats en regard des caractéristiques vis-à-vis de nos besoins pour la fouille de processus. En effet la fouille de processus est basée sur des *event logs*, c'est-à-dire que l'on se base sur les traces et leur horodatage pour en extraire de la connaissance. En particulier il convient de rassembler les évènements qui correspondent à un cas pour en étudier les relations de précédence afin de construire le modèle de processus.

CHAPITRE 3 | Fouille de processus

Sommaire

3.1 Introduction 30
3.2 Fouille de processus (*Process Mining*) 31
3.3 Motifs courants dans les processus 33
3.4 Modélisation des processus 35
3.5 Les algorithmes de fouille 39
3.6 Propriétés du modèle miné 42
3.7 Limites 45
3.8 Principaux algorithmes 48
3.9 Conclusion 51

3.1 Introduction

Le système d'information représente l'ensemble de ressources et d'outils permettant d'organiser et de donner accès à l'information d'une organisation. Il est vu comme un système socio-technique incluant à la fois la structure organisationnelle, les processus métier ainsi que les données afférentes (Piccoli, 2007). La structure organisationnelle représente les règles de répartition de l'autorité, des tâches, de contrôle et de coordination. Elle permet notamment de définir les différents acteurs du système d'information, leurs rôles et leurs droits. Les processus métier représentent les activités ou les enchaînements d'activités relatives aux affaires de l'organisation concernée. Ainsi, un processus $\mathcal{P}$ est caractérisé par un ensemble ordonné d'activités a_i.

Ces processus et activités consomment et produisent des données gérées par le système d'information. Un processus peut correspondre à un assemblage de traitements réalisés par un opérateur humain (appelé aussi ressource humaine) ou par appel de services extérieurs, tels que les services web. De plus, dans certains contextes, les processus utilisent des données collectées par des sources variées (équipements, humains, capteurs...) et depuis différents supports. Tout ou partie des processus impliqués dans la vie d'un système d'information peuvent être déduits des différentes traces recueillies par l'observation et l'enregistrement des différents événements survenant dans le système d'information accompagnant ses activités (Van Der Aalst et al., 2015). L'ensemble des méthodes, techniques et outils permettant de déterminer et d'analyser les processus métier à partir des traces est regroupé sous le terme de « fouille de processus » (*process mining*)

Dans ce chapitre, nous allons présenter les différentes méthodes et algorithmes de la fouille de processus. Nous nous attacherons à introduire les principes et montrer les limites des approches actuelles afin de déterminer quelles méthodes peuvent être utilisées dans notre contexte particulier, permettant de définir une méthodologie pour proposer un compagnon à l'exécution de processus métier.

Nous commencerons, en section 3.4, par présenter les modèles manipulés dans le cadre de la fouille de processus. Nous rappellerons les définitions sur les réseaux de Petri pour commencer, puis décrirons les restrictions Workflow Net et Process Tree qui permettent la modélisation des processus métier. Nous présenterons aussi les Causal Net (réseaux de causalité) utilisés comme modèles intermédiaires pour décrire les relations de causalités entre activités.

Après une présentation des motifs les plus couramment rencontrés dans les processus métier en section 3.3, nous décrirons les techniques de fouille en section 3.5 où nous décrirons le principe de fouille des algorithmes par découpage, des algorithmes statistiques et des algorithmes génétiques. Nous aborderons ensuite les propriétés des modèles minés en section 3.6 et les limites du domaine en section 3.7. Nous finirons par une analyse des principaux algorithmes existants en section 3.8 avant une conclusion générale du chapitre.

3.2 Fouille de processus (*Process Mining*)

La fouille de processus a été développée par l'Université de Technologie d'Eindhoven qui a cherché à exploiter les traces d'exécution pour la modélisation de processus (Van Der Aalst and Weijters, 2004). Elle a été définie comme étant : « la méthodologie permettant de découvrir une définition de processus à partir d'un ensemble d'exécutions de processus réels, à l'aide de journaux d'activités provenant de systèmes d'informations prenant en compte les processus ». Comme présenté dans la figure 3.1, la fouille de processus peut être vue comme un lien entre la science des processus et la fouille de données (Van Der Aalst, 2016; Rebuge and Ferreira, 2012; Bozkaya et al., 2009).

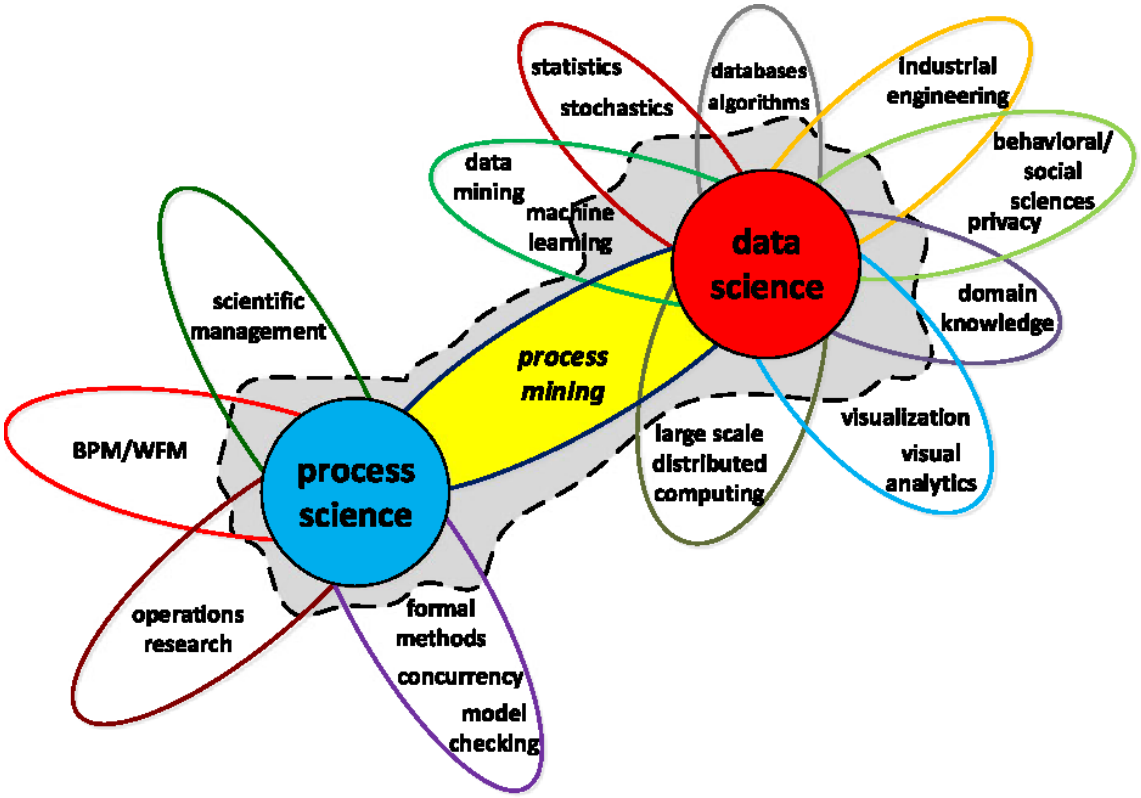

FIGURE 3.1 – Positionnement de la fouille de processus (Van Der Aalst, 2016)

Son objectif est de déduire le modèle des processus à partir des traces d'exécution. Ainsi, la fouille de processus permet d'établir le lien entre les processus réalisés et les processus modélisés afin de pouvoir les étudier.

3.2.1 Utilisation du process mining

Dans (Van Der Aalst, 2016), l'auteur considère trois domaines d'utilisation de la fouille de processus :

— la découverte des processus : produire un modèle à partir d'un journal des événements sans utiliser d'informations *a priori* ;

— le contrôle de conformité : analyser la conformité des traces vis-à-vis d'un modèle ;

— l'amélioration du modèle de processus : adapter le modèle existant aux nouvelles traces collectées ou le corriger en fonction de ces dernières.

Par exemple, Accorsi and Stocker (2012) proposent d'utiliser l'analyse de conformité pour réaliser un audit d'un processus de validation de crédit. Cette analyse permet d'identifier les processus qui dévient du modèle initial (par exemple, pour lesquels les validations ont été accordées alors que l'employé ne disposait pas des autorisations nécessaires ou les contraintes de temps n'ont pas été respectées). Comme autre exemple, Carmona et al. (2010) proposent une méthode basée sur l'analyse des régions permettant d'obtenir un réseau de Petri borné minimal qui est valide vis-à-vis des traces observées.

Dans le cadre de nos travaux, nous cherchons à guider l'utilisateur vers un objectif dans son processus d'utilisation du SI. Nous avons besoin d'un moyen pour modéliser les corrélations entre les activités afin de trouver une exécution du processus qui lui permettra d'atteindre son objectif. La fouille de processus nous permet de récupérer le modèle de processus d'usage à partir des traces. Modèle que nous utiliserons pour accompagner l'utilisateur dans son processus métier courant.

3.2.2 Organisation des données

Pour mener des analyses sur les données avec un objectif de fouille de processus, il est tout d'abord nécessaire de collecter les informations adéquates. Comme expliqué dans le chapitre 2, les logs constituent une bonne source de données et peuvent être considérés comme des traces brutes. Pour pouvoir les analyser, il est nécessaire de les transformer en un ensemble de traces modélisées portant les informations nécessaires, permettant notamment d'identifier les données représentant les activités. Une fois le modèle de traces défini, il faut ensuite sélectionner l'algorithme de fouille de processus le plus adéquat à ces traces pour obtenir le modèle de processus. Les traces modélisées suivant le modèle d'événement peuvent être représentées par le format XES (cf. paragraphe 2.4.1.2).

Après filtrage et transformation des traces brutes, les traces modélisées se présentent formellement comme un ensemble d'exécutions de processus (ensemble de traces d'événements) où chaque élément de l'ensemble est une séquence d'activités (aussi appelée « cas »). Par exemple, dans l'ensemble de traces d'événement $E = [\langle a, b, c\rangle^2, \langle b, a\rangle^3]$, $\langle a, b, c\rangle$ et $\langle b, a\rangle$ sont des représentations d'exécutions sous forme de séquence d'activités effectuant successivement les activités a, b et c pour la première séquence et b puis a pour la seconde séquence. Dans cette représentation, l'exposant décrit le nombre d'exécutions d'une séquence dans les traces. Ainsi,

il y a dans l'ensemble des traces E deux exécutions de la séquence d'activité $\langle a, b, c\rangle$ et trois exécutions de $\langle b, a\rangle$.

3.2.3 Principe général de la fouille de processus

La fouille de processus consiste à identifier les liens de causalité entre deux activités dans un ensemble de traces (Van Der Aalst et al., 2003). Soit un ensemble de traces $E = \{e_1, e_2, ..., e_n\}$ contenant des activités a_1, a_2 ... , a_n. Par exemple, $E = [\langle a, b, c\rangle^2, \langle b, a\rangle^3]$ qui contient 5 exécutions réparties en 2 séquences avec les activités possibles a, b ou c.

Quatre relations sont définies entre deux activités quelconques a_1 et a_2 :

— Succession directe (*direct succession*), notée $a_1 > a_2$ indique que a_1 est immédiatement suivi de a_2 dans certaines séquences (nous avons dans l'exemple $a > b$ pour la première séquence et $b > a$ pour la seconde) ;

— Séquence (*causality*), notée $a_1 \rightarrow a_2$, si a_1 est situé avant a_2 (par exemple nous avons $a \rightarrow c$) ;

— Parallèle, noté $a_1||a_2$, si on a à la fois $a_1 > a_2$ et $a_2 > a_1$;

— Choix (*choice*), noté $a\#b$, s'il n'y a jamais ni $a_1 > a_2$ ni $a_2 > a_1$

3.3 Motifs courants dans les processus

D'une manière générale, un processus peut-être représenté par un flux d'activités ou flux de travail (*workflow*). Les développements sur la modélisation et la conception de processus métier ont amené à définir un ensemble de motifs (*workflow pattern*) que l'on retrouve dans un processus.

Dans (Russell et al., 2006), les auteurs ont défini un cadre conceptuel pour la conception de processus. Le résultat de cette étude est un ensemble de vingt motifs de workflow orientés suivant le point de vue contrôle d'un processus.

Dans le cadre de la fouille de processus, il est intéressant d'identifier les motifs de processus que l'on est capable de miner. Nous retenons les processus élémentaires suivants [1] (illustrés dans la Figure 3.2 en utilisant les réseaux de Petri décrits au paragraphe 3.4.1) :

1. La séquence. C'est un motif désignant l'enchaînement de plusieurs transitions les unes à la suite des autres ;
2. Le départ en parallèle (*AND-split*). C'est un motif qui permet de désigner le début de franchissement de transitions en parallèle ;

1. Processus définis notamment par la *Workflow Patterns initiative* `http://www.workflowpatterns.com`

3. La synchronisation (*AND-join*). Ce motif permet de désigner la fin des franchissements de transitions en parallèle ;
4. Le choix exclusif (*XOR-split*). Ce motif désigne l'obligation de choisir entre plusieurs activités possibles à franchir. Toutes sont franchissables, mais il n'est pas possible d'en faire plusieurs ;
5. La fusion (*XOR-join*). Ce motif indique la fin d'un choix exclusif ;

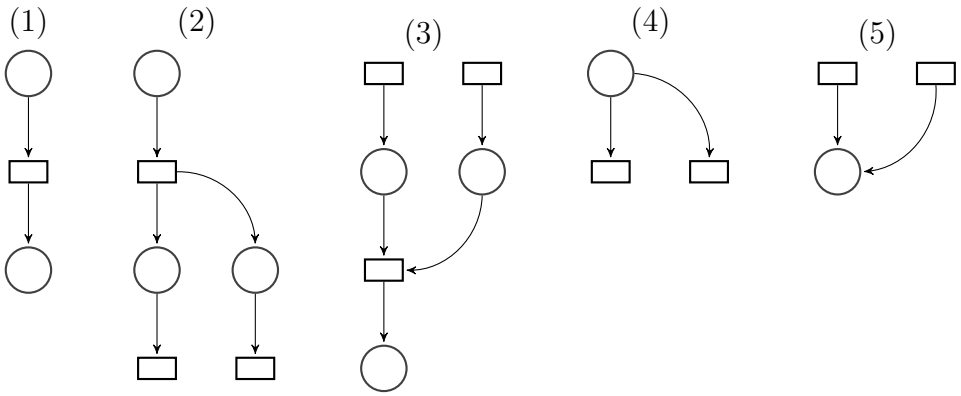

FIGURE 3.2 – Motifs élémentaires de processus

Dans le cadre de nos travaux, nous devons également considérer les boucles :

— boucles courtes (*short loop*) qui sont définies comme une activité unique utilisable zéro ou plusieurs fois (de Medeiros et al., 2004) (boucle au dessus de la séquence principale dans la Figure 3.3) :

— boucles mal formées (*arbitrary cycles ou unstructured loop*) qui apparaissent dans un cycle long (boucle en dessous de la séquence principale dans la Figure 3.3).

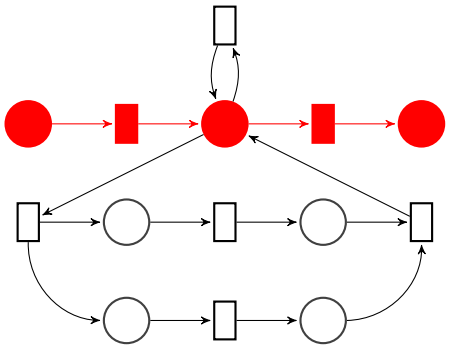

FIGURE 3.3 – Exemple de boucles

Le cas des boucles constitue une difficulté dans la modélisation des processus. La plupart des algorithmes de fouille de processus sont capables de traiter des cas simple de boucles

(de Medeiros et al., 2004). La découverte des boucles plus complexes nécessite d'utiliser des algorithmes plus évolués (comme les algorithmes de la section 3.5).

3.4 Modélisation des processus

Un modèle de processus fournit des outils permettant de représenter les processus sous une forme donnée. Ils fournissent aussi un ensemble d'outils mathématiques permettant de calculer l'évolution des exécutions du processus représenté. Par exemple, les réseaux de Petri permettent de représenter un processus sous la forme de transitions et de places (cf. paragraphe 3.4.1).

Les algorithmes de fouille de processus n'utilisent pas tous le même modèle. Il est donc nécessaire de présenter quelques modèles avant de présenter les algorithmes utilisés dans la fouille de processus. Nous présentons dans ce qui suit la présente section les formalismes les plus répandus permettant de représenter et de manipuler les processus dans le cadre de la fouille de processus. Nous commençons par définir le modèle général des réseaux de Petri, puis décrirons les restrictions Workflow Net et Process Tree qui permettent plus particulièrement la modélisation des processus métier. Nous présenterons aussi les Causal Net (réseaux de causalité), modèle utilisé dans plusieurs algorithmes pour décrire les relations de causalités entre activités.

3.4.1 Réseaux de Petri (RdP)

Un réseau de Petri est un modèle mathématique avec une représentation graphique qui permet d'exprimer des séquences d'activités avec des synchronisations et des partages de ressources (Murata, 1989). Ils sont utilisés dans de nombreux domaines : validation de protocole, planification d'atelier, supervision de processus, *etc*.

Un réseau de Petri est un graphe orienté composé de deux types de nœuds : des places et des transitions. Ces nœuds sont reliés alternativement par des arcs. Si un arc relie une place à une transition, il est dit « entrant » ; s'il relie une transition à une place, il est dit « sortant ».

Des jetons, ou marques, sont associés aux places. Ils décrivent l'état courant du système. Leur évolution dépend d'un ensemble de règles que décrivent les transitions.

Les réseaux de Petri sont définis formellement comme suit.

Définition (Réseau de Petri marqué)**.** *Un réseau de Petri marqué est le couple :* $N =< R, s >$ *où :*

- *R est un réseau de Petri défini par le quadruplet* $< P, T, Pre, Post >$ *avec :*
  - P *un ensemble fini de places.*
  - T *un ensemble fini de transitions.*

- $P \cap T = \emptyset$
- $\mathcal{F} \subseteq (P \times T) \cup (T \times P)$ *est l'ensemble des arcs reliant les places et les transitions.* Pre *est la matrice d'incidence avant, elle décrit l'ensemble des arcs entrant des transitions et* $Post$ *est la matrice d'incidence arrière, elle définit l'ensemble des arcs sortant des transitions.*

— s *est le marquage, c'est une application qui associe à chaque place du réseau un ensemble de jetons. Ainsi, dans chaque place, il peut y avoir un certain nombre de jetons, pouvant être nul. On note* s_0 *le marquage initial du réseau de Petri.*

Les jetons « se déplacent » dans le réseau de Petri en respectant des règles d'évolution. Une transition est franchissable si chacune de ses places d'entrée contient au moins un jeton ($\forall p \in P,\ s(p) \geq Pre(p,t)$), dans le cas le plus simple des réseaux pour lesquels aucun poids n'est attaché aux arcs.

Le franchissement (ou tir) d'une transition correspond à retirer un jeton de chaque place amont à la transition, et à ajouter un jeton à chaque place aval (place de sortie). Par exemple, la figure 3.4 illustre un exemple où le franchissement de la transition t_1 consomme un jeton dans les places A et D pour en produire un dans la place B puis le franchissement de la transition t_2 consomme un jeton de B pour en produire un dans C et un dans D. Le franchissement d'une transition correspond à l'occurrence d'un évènement. Une séquence de franchissement de transitions décrit l'exécution d'un processus.

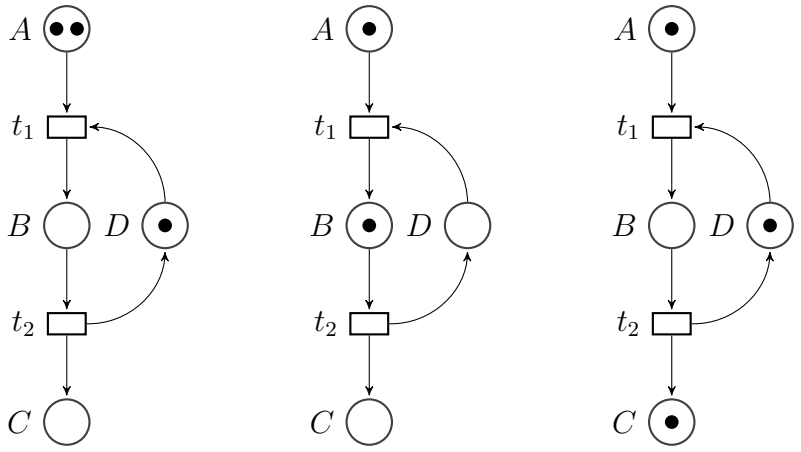

FIGURE 3.4 – Évolution du marquage d'un RdP

Les réseaux de Petri décrivent un système en intention. Un graphe dont les sommets correspondent à tous les marquages possibles (à condition qu'il y ai un nombre de marquages fini) et les arcs décrivent les évènements permettant de passer d'un marquage à un autre, est appelé graphe des marquages accessibles. Si on trace ce graphe, il est possible de déduire les propriétés du réseau Petri. Par exemple, on peut garantir qu'une place d'un réseau contient

un nombre maximal de jeton ou, si le graphe est fortement connexe, qu'un état reste toujours accessible.

3.4.2 Workflow Net

Les *Workflow Nets* constituent un sous-ensemble restreint des réseaux de Petri marqués. Ce sont des réseaux de Petri possedant une unique place marquée qui correspond au début du processus et une unique place de terminaison du processus (Figure 3.5). De plus, tous les nœuds du graphe appartiennent à un chemin allant de la place représentant le début du processus à la place représentant sa fin (Van Der Aalst, van Hee, ter Hofstede, Sidorova, Verbeek, Voorhoeve and Wynn, 2011). Formellement, les Workflow nets sont définis ainsi :

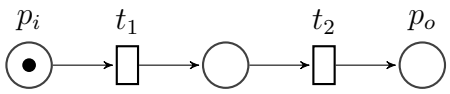

FIGURE 3.5 – Exemple de Workflow-net

Définition (Workflow Net). *Un réseaux de Petri R est un Workflow Net si et seulement si :*

1. *Initialisation :* $\exists\, p_i \in P : \forall t \in T, \nexists Post(t, p_i)$
2. *Terminaison :* $\exists\, p_o \in P : \forall t \in T, \nexists Pre(p_o, t)$
3. *Connexité : on ajoute une transition particulière* $\bar{t}$ *reliant* p_o *à* p_i *et on obtien ainsi* $\bar{R} = (P, T \cup \bar{t}, \mathcal{F} \cup \{(p_o, \bar{t}), (\bar{t}, p_i))\}$ *fortement connecté.*

Dans les travaux de l'Université de Technologie d'Eindhoven (Van Der Aalst, 1996; Dixit, 2019) l'auteur définit la notion de « soundness ». Un Workflow Net est dit *sound* si :

— $\forall_S([i] \xrightarrow{*} s) \Rightarrow (s \xrightarrow{*} [o])$, pour chaque marquage accessible depuis la place initiale, il existe une séquence de transitions qui mène au marquage final avec $[i]$ le marquage ne contenant que la place initiale p_i et $[o]$ le marquage ne contenant que la place finale p_o ;

— $\forall M([i] \xrightarrow{*} s \wedge s \geq [o]) \Rightarrow (s = [o])$, l'état final est le seul accessible depuis l'état initial ;

— le réseau est vivant.

Cela implique que quel que soit le marquage, il est possible d'atteindre le marquage final $[p_o]$.

Les Worfklow Nets peuvent être organisés en blocs, appelés Block-Structured Workflow Nets, qui constituent une représentation hiéararchisée ou chaque partie représente un workflow qui peut être divisé de la même manière récursivement (Leemans et al., 2013 a).

3.4.3 Process tree

Un *Process Tree* est une structure arborescente représentant un processus (Bose and Van Der Aalst, 2009; Leemans et al., 2013 a). L'arbre représente le processus complet. Les feuilles décrivent les activités du processus et les nœuds donnent les relations entre les activités.

Ils sont définis par :

Définition (Process Tree)**.** *Soit $\mathcal{A}$ un ensemble fini d'activités.*

- *a, avec $a \in \mathcal{A}$, est un process tree ;*
- *soient $M_1 ... M_n$ avec $n > 0$ des process tree et x un opérateur de process tree alors $x(M_1, ..., M_n)$ est un process tree ; x pouvant être une composition séquentielle ($\rightarrow$), un choix exclusif ($\times$), une composition parallèle ($\wedge$) ou une boucle ($\circlearrowleft$).*

Cette représentation est équivalente aux Block-Structured Workflow Net.

3.4.4 Graphe causal (Causal Net)

Un graphe causal est un graphe dont les sommets représentent des activités et les arcs des relations de causalités entre activités (Van Der Aalst et al., 2003; Van Der Aalst, 2016). Chaque activité possède des liaisons en entrée et en sortie. Ces liaisons permettent de traduire la synchronisation des activités.

Un Causal Net est formellement défini par :

Définition (Causal Net)**.** *Un Casual Net C est un tuple $C = (\mathcal{A}, a_i, a_o, D, I, O)$ avec :*

- *$\mathcal{A}$ un ensemble fini d'activités ;*
- *$a_i \in \mathcal{A}$ une activité de départ ;*
- *$a_o \in \mathcal{A}$ une activité de fin ;*
- *$D \subseteq \mathcal{A} \times \mathcal{A}$ une relation de dépendance entre activités ;*
- *$AS = \{X \subseteq \mathcal{P} | X = \{\emptyset\} \vee \emptyset \notin X\}$ définit les liaisons entre activités, où $\mathcal{P} = \{A' | A' \subseteq \mathcal{A}\}$ est powerset (l'ensemble des parties d'un ensemble) des activités de $\mathcal{A}$ où le powerset est l'ensemble des sous-ensembles de l'ensemble des activités. De plus, AS est un ensemble d'ensemble d'activités ;*
- *$I \in A \rightarrow AS$ définit l'ensemble des liaisons d'entrée potentielles pour chaque activité ;*
- *$O \in A \rightarrow AS$ définit l'ensemble des liaisons de sortie potentielles pour chaque activité.*

Avec :

- *$D = \{(a_1, a_2) \in A \times A | a_1 \in \bigcup_{as \in I(a_2)} as\}$;*
- *$D = \{(a_1, a_2) \in A \times A | a_2 \in \bigcup_{as \in O(a_1)} as\}$;*
- *$\{a_i\} = \{a \in A | I(a) = \{\emptyset\}\}$;*

— $\{a_o\} = \{a \in A | O(a) = \{\emptyset\}\}$;

— *toutes les activités du graphe* (A, D) *se trouvent sur le chemin allant de* a_i *à* a_o.

Avec ce modèle, il est difficile d'exprimer des séquences valides. En revanche, il est très utile pour décrire les relations de causalité entre activités. Ce modèle est utilisé dans les approches statistiques basées sur les fréquences d'occurrence des relations entre activités.

3.5 Les algorithmes de fouille

Il existe différents algorithmes pour effectuer la fouille de processus. Nous présentons ci-après les grands principes de ces algorithmes.

3.5.1 Algorithmes de base

Les techniques de découverte de processus ont pour but de modéliser les processus à partir des informations analysées dans les logs de l'application. Il n'est pas forcément nécessaire de connaître au préalable le processus analysé pour dégager un modèle cohérent. Il existe différentes méthodes pour aborder la génération de modèles depuis des logs. Certaines techniques se basent sur les relations entre les tâches alors que d'autres cherchent à découvrir les états possibles du système pour en déduire les relations de causalité.

Nous verrons ici les algorithmes se basant sur les relations entre les tâches. Nous verrons que l'on peut considérer les relations localement, entre chacune des tâches du modèle ou regrouper les tâches et trouver le lien entre les groupements de tâches et donc considérer les relations globales.

(Van Der Aalst et al., 2003) proposent un algorithme de fouille de processus nommé α Miner et basé sur les relations de causalités entre activités décrites précédemment.

L'algorithme prend en entrée un ensemble de traces noté T représentant les exécutions d'un processus donné. Une trace $t \in T$ représente une activité effectuée. Chaque exécution, notée σ, est un ensemble organisé par ordre d'apparition de traces, noté σ, et l'ensemble de toutes les exécutions est noté L. Il est à noter que chaque traces t de T appartient forcément à une exécution même si l'exécution ne doit contenir que t. Nous avons donc les règles suivantes :

$$T = \{t_0, ..., t_n\}, \sigma_i \subseteq T, L = \{\sigma_0, ..., \sigma_m\}, \forall t \in T \exists \sigma | t \in \sigma$$

L'algorithme est organisé en huit étapes :

1. $T_L = \{t \in T | \exists_{\sigma \in L} t \in \sigma\}$. On identifie les activités présentes dans les logs. Pour chaque activité on crée une transition.
2. $T_I = \{t \in T | \forall_{\sigma \in L} t = first(\sigma)\}$. On identifie l'activité de départ qui correspond à l'activité de départ de toutes les traces.

3. $T_O = \{t \in T | \forall_{\sigma \in L} t = last(\sigma)\}$. Idem pour l'activité de terminaison.
4. $X_L = \{(A, B) | A \subseteq T_L \wedge A \neq \varnothing \wedge B \subseteq T_L \wedge B \neq \varnothing \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_L b \wedge \forall_{a_1, a_2 \in A} a_1 \#_L a_2 \wedge \forall_{b_1, b_2 \in B} b_1 \#_L b_2\}$. On identifie les enchaînements d'activités (par paire). Puis, on réduit les paires pour identifier les places qui relient les activités. En effet, si deux activités (représentées par des transitions) se suivent, c'est qu'il existe une place permettant de les relier. Cet ensemble contiendra des doublons. Par exemple, prenons $a, b, c \in T$ avec $a \rightarrow b, a \rightarrow c$ et $b \# c$ alors $(\{a\}, \{b\})$ appartient à X et $(\{a\}, \{b, c\})$ également. L'objectif est de réduire les places afin de ne garder que celles reliant le plus de transitions (ici $(\{a\}, \{b, c\})$).
5. $Y_L = \{(A, B) \in X_L | \forall_{A', B' \in X_L} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$. On supprime toutes les paires (A, B) qui ne sont pas maximales.
6. $P_L = \{p_{(A,B)} | (A, B) \in Y_L\} \cup \{i_L, o_L\}$. On définit les places de départ et de terminaison des processus.
7. $F_L = \{(a, p_{(A,B)}) | (A, B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) | (A, B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) | t \in T_I\} \cup \{(t, o_L) | t \in T_o\}$. Identification des arcs. Les places (A, B) contenues dans Y ont été formées à partir de leurs arcs entrant (venant des transitions dans A) et de leurs arcs sortant (allant vers les transitions dans B).
8. $\alpha(L) = (P_L, T_L, F_L)$. On obtient alors le modèle, réseau de Petri, découvert.

3.5.2 Algorithmes statistiques

Les algorithmes statistiques se basent sur un seuil pour considérer une règle comme étant vérifiée. Cela permet de ne pas considérer les exceptions dans l'exécution comme des règles à inférer dans le modèle. Ce seul principe rend l'algorithme robuste au bruit, car il implique de ne pas considérer, dans le même contexte, les évènements ne survenant pas assez souvent.

Leemans et al. (2013 a) proposent un algorithme de découverte des processus à partir de traces d'exécutions incomplètes à l'aide de méthodes statistiques pour identifier les relations entre évènements.

Les algorithmes de découpage en régions étudient les relations entre activités pour construire un graphe de dépendance. L'objectif est d'exprimer les relations de causalité entre deux activités. Il se base sur le graphe de causalité, mais les relations sont déterminées de leurs probabilités d'existence. Pour cela on construit deux métriques : la fréquence des successions immédiates et la dépendance. À partir de seuils, ces métriques permettent de déterminer s'il y a réellement une relation de dépendance entre deux activités ou s'il s'agit d'une erreur.

Pour déterminer le graphe de dépendance, il faut calculer deux matrices décrivant les relations entre les activités. La première compte la fréquence de succession entre deux activités.

Par exemple à partir des traces $T = [\langle a, d, e\rangle^{10}, \langle a, d, d, e\rangle^{2}]$, la fréquence de succession immédiate entre a et d est 12, et 2 entre d et d.

La mesure de dépendance ($|a_i \Rightarrow_L a_j|$) entre deux activités a_i et a_j est donnée par

$$|a_i \Rightarrow_L a_j| = \begin{cases} \frac{|a_i >_L a_j| - |a_j >_L a_i|}{|a_i >_L a_j| + |a_j >_L a_i| + 1} \; si \; a_i \neq a_j \\ \frac{|a_i >_L a_i|}{|a_i >_L a_i| + 1} \; si \; a_i = a_j \end{cases}$$

Cette mesure est toujours comprise entre -1 et 1.

3.5.3 Algorithmes génétiques

L'algorithme génétique ou *genetic mining* utilise les principes de l'évolution des espèces pour miner un processus. Une population initiale de modèles est générée sous la forme de matrices de causalités. Ensuite, nous sélectionnons parmi cette population les individus qui représentent le mieux le processus recherché selon un critère choisi. Ces individus sont mélangés/fusionnés pour créer la génération suivante. Des mutations sont alors introduites dans notre population en enlevant ou en ajoutant arbitrairement des arcs et des nœuds à certains modèles. Nous pouvons ensuite sélectionner les meilleurs modèles parmi la population et recommencer ces étapes jusqu'à obtenir un modèle atteignant un seuil sur le critère de sélection (Weijters et al., 2006; Weijters and Ribeiro, 2011).

Création de la population La population initiale est construite aléatoirement sous la forme de matrice de causalités. Pour cela, on peut générer aléatoirement des relations entre les différentes activités ou utiliser des conditions de construction des modèles initiaux. L'avantage de définir des règles pour établir les premiers modèles permet d'accélérer le calcul nécessaire à la génération d'un modèle. En effet, plus la population courante est proche du résultat attendu plus il y a de chances que la population suivante contienne l'individu résultat.

Sélection des meilleurs modèles La première chose à faire est de définir le critère de qualité des modèles. Ce critère doit être mesurable afin de pouvoir calculer une distance entre chaque modèle et l'objectif attendu. En règle général, cette mesure est choisie parmi les mesures habituelles qualifiant la conformité d'un modèle vis-à-vis de traces.

Opérations génétiques On utilise l'élitisme, le croisement et la mutation pour générer de nouvelles populations s'approchant de plus en plus de l'objectif défini. Pour cela, tant qu'un modèle n'a pas été dégagé on génère les nouvelles populations.

Élitisme : À chaque nouvelle génération, on conserve un certain pourcentage des meilleurs modèles de la génération précédente, cela permet de ne pas régresser entre deux générations.

Croisement : On choisit aléatoirement un certain nombre d'individus dans la population, puis on sélectionne les deux meilleurs de ce groupe. Ensuite, on génère deux enfants depuis les modèles en combinant leurs structures.

Mutation : Elle consiste en la modification aléatoire de certains enfants selon un seuil de mutation. Pour ce faire, on génère aléatoirement un ou plusieurs arcs ou nœuds dans le modèle.

Condition d'arrêt L'algorithme s'arrête lorsque l'algorithme a généré n populations (n étant un paramètre de l'algorithme) ou lorsque le meilleur individu est le même depuis $n/2$ génération.

3.6 Propriétés du modèle miné

L'objectif du process mining est de produire automatiquement des modèles de processus décrivant avec précision la réalité observée en se basant sur les séquences produites par l'exécution dudit processus. Pour une meilleure compréhension des modèles générés, il est nécessaire de produire des rapports contenants les modèles produits, leurs indicateurs de conformité et une représentation des traces utilisées pour construire les modèles (Rozinat and Van Der Aalst, 2008).

Quatre indicateurs ont été définis et constituent les principaux indicateurs considérés par la communauté :

— justesse (*fitness*) : cet indicateur évalue dans quelle mesure le modèle de processus obtenu peut reproduire toutes les traces disponibles ;

— précision : un modèle est précis s'il ne permet pas trop de traces (séquences) différentes ;

— généralisation : elle quantifie dans quelle mesure le modèle obtenu sera capable de reproduire les futures traces du processus ;

— simplicité : cette valeur représente l'étendue structurelle du modèle de processus.

Les métriques peuvent être utilisées conjointement à un graphe représentant la temporalité des évènements pour fournir des indications supplémentaires sur la construction du modèle. Cela permet de comprendre certaines incohérences ou de valider certaines constructions. Chacune des quatre métriques qualifie un modèle de processus sur une valeur entre 0 et 1, avec 1 comme valeur optimale.

La détermination des indicateurs ci-dessus nécessite de pouvoir mesurer la distance entre les traces et le modèle. C'est le rôle des algorithmes d'alignement que nous introduisons ci-après. Les 4 indicateurs sont ensuite définis plus en détail.

3.6.1 Alignement du modèle et des traces

Les traces peuvent être incomplètes ou comporter des erreurs. La conséquence est que certaines traces ne correspondent à aucun processus défini par le modèle. On utilise alors le principe d'alignement, tel que défini par les algorithmes d'alignement, pour déterminer la distance entre une trace et le modèle (Van Der Aalst et al., 2012).

Pour aligner le modèle et les traces nous considérons des activités dans les traces et dans le modèle. Lorsqu'une activité dans la trace n'a pas d'équivalent dans le modèle, et réciproquement, on considère qu'il s'agit de l'activité nulle notée $\perp$.

Soit T_M l'ensemble des processus modélisés, T_L l'ensemble des traces et $A(T)$ l'ensemble d'activités apparaissant dans T, alors :

— (x, y) est une activité dans le modèle si $x \in A(T_M)$ et $y = \perp$

— (x, y) est une activité dans les traces si $x = \perp$ et $y \in A(T_L)$

— (x, y) est une activité à la fois dans le modèle et dans les traces si $x \in A(T_M)$ et $y \in A(T_L)$

— (x, y) est une activité illégale si $x = \perp$ et $y = \perp$

$A_{LM} = \{(x, y) \in (A(T_M) \times A(T_L)) | x \in A(T_M) \vee y \in A(T_L)\}$ est l'ensemble des activités légales. En associant un poids w à ces activités, on peut calculer la distance δ entre le modèle et les traces en faisant la somme des poids de chaque activité calculé à partir des alignements.

Soit $\gamma(t_L, t_M)$ l'alignement entre la trace $t_L \in T_L$ et le processus $t_M \in T_M$ alors la distance entre t_M et t_L est égale à : $\delta(\gamma(t_L, t_M)) = \sum_{(x,y)\in\gamma} w(x, y)$

Grâce à cette distance, nous pouvons chercher le meilleur alignement possible entre une trace et son modèle noté $\lambda(t_L)$. Il nous suffit de chercher le processus pour lequel δ est minimale. Une fois cet alignement réalisé, nous pouvons réaliser les calculs des indicateurs introduits précédemment.

3.6.2 Justesse

L'indicateur de justesse (*fitness*) mesure à quel point le modèle est représentatif des traces. On calcule, en rejouant les traces sur le modèle, la quantité d'exécution qui effectue un processus défini par le modèle dans sa totalité. Pour cela, on calcule la distance entre les traces et le modèle en alignant les traces sur les processus possibles du modèle. On rapporte ensuite la distance totale à la distance maximale possible δ_{max}.

Soit W un Workflow Net et T l'ensemble des traces générées par W alors la fitness $F(W, L)$ du modèle W devant la trace L se calcule ainsi : $F(W, L) = \frac{\sum_{t\in L} \delta(\lambda(t))}{\delta_{max}}$

3.6.3 Précision

La mesure de précision consiste à vérifier que le modèle ne permet pas de comportements n'existant pas dans les traces. Pour ce faire, il nous faut vérifier les comportements permis par le modèle à chaque instant et vérifier qu'ils apparaissent tous dans les traces. Il faut vérifier les activités possibles à chaque instant dans les traces et le modèle.

En considérant :

— ϵ l'ensemble des événements rencontrés dans les logs ;

— $enabled_L(e)$ les activités permises par le modèle ;

— $enabled_M(e)$ les activités observées dans les logs dans un contexte similaire.

Puis, il faut calculer l'indicateur de précision $P(L, M)$ tel que :

$$P(L, M) = \frac{\sum_{e \in \epsilon} \frac{enabled_L(e)}{enabled_M(e)}}{|\epsilon|}$$

Il est à noter qu'il est possible de découvrir à coup sûr un modèle de processus pouvant reproduire les traces en faisant un modèle de processus en fleur (Figure 3.6). Il s'agit d'un modèle ne contenant qu'une place (un état de repos) reliée à toutes les transitions par une boucle élémentaire. Ce genre de modèle peut reproduire n'importe quelle série d'activités à condition que celles-ci fassent partie de l'ensemble des activités observées. Cependant, ce modèle manquera de précision sur le comportement observé.

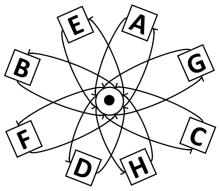

FIGURE 3.6 – Exemple de modèle de processus en fleur

3.6.4 Généralisation

La généralisation considère la fréquence avec laquelle chaque activité du processus est visitée. Plus les activités sont visitées, plus le modèle généralise le comportement modélisé. Un modèle décrivant un processus pour chaque trace possible possède un score de généralisation très faible mais un haut score en précision.

Si une activité est visitée souvent, cela nous indique que son comportement est correctement généralisé. Si, au contraire, certaines parties du modèle sont peu visitées alors la généralisation est mauvaise. Un modèle en fleur aura par exemple un bon score de généralisation dû au fait que n'importe quelle séquence d'activité est possible dans ce modèle.

Pour calculer cette métrique, il nous faut récupérer le nombre d'occurrences de chaque évènement et le nombre d'activités candidates à chaque état possible du système. Nous définirons donc une fonction $N(e)$ donnant le nombre de fois où le système se trouve dans l'état e et $O(e)$ donnant l'ensemble des évènements accessibles depuis l'état e. Ensuite, nous calculons la probabilité $P(e)$ que la prochaine visite à l'état e révèle une activité non vue précédemment.

$$P(e) = \begin{cases} si\ N(e) \geq O(e) + 2 \Rightarrow \frac{O(e)(O(e)+1)}{N(e)(N(e)-1)} \\ sinon \Rightarrow 1 \end{cases}$$

L'indicateur de généralisation est alors :

$$G(L, M) = 1 - \frac{\forall_{e \in \epsilon} P(e)}{|\epsilon|}$$

où, comme précédemment, ϵ désigne l'ensemble des événements rencontrés dans les logs.

3.6.5 Simplicité

La mesure de simplicité (Buijs et al., 2014) qualifie la complexité du modèle. La découverte de processus mène souvent à des modèles difficiles à interpréter de par leur nombre de transitions et/ou de places. La simplicité est mesurée en comparant la taille du modèle (nombre de places et de transitions présentes dans le réseau de Petri) avec le nombre d'activités dans les traces. Ceci est basé sur le fait que la taille d'un modèle de processus est le facteur principal pour la complexité perçue et l'introduction d'erreurs dans les modèles de processus.

$$S(L, M) = 1 - \frac{\#activités\ dupliquées + \#activités\ manquantes}{\#noeuds\ du\ modèle + \#classes\ d'événements\ des\ logs}$$

Ces quatre mesures nous permettent d'estimer la qualité du modèle découvert et ainsi la performance de l'algorithme utilisé vis-à-vis du jeu de données. Dans les paragraphes suivants, nous soulignons les difficultés auxquelles on fait face lorsque l'on utilise la fouille de processus.

3.7 Limites

Nous verrons ici les limites connues de la fouille de processus. La difficulté survient lorsque les modèles sont composés de structures complexes soulevant plusieurs des problèmes présentés ci-après. La combinaison de ces structures est la réelle limite de ce domaine de recherche (Van Der Aalst, 2016).

3.7.1 Source de données hétérogènes

Une des problématiques majeure de la fouille de processus ou de données est de devoir récupérer toutes les informations sur l'application et ses évènements pour permettre une analyse de meilleure qualité. Cependant, il est courant que les systèmes d'information possèdent plusieurs bases de données utilisant des formalisations et/ou technologies différentes voire disponibles dans des bases de données différentes. La complexité dans cette tâche indispensable à la fouille de processus et de fusionner l'ensemble les informations.

La solution souvent utilisée pour réduire la complexité de cette problématique et de créer un modèle pivot qui permettra de récupérer toutes les informations de manière normalisée.

3.7.2 Bruit

La plupart des algorithmes de fouille supposent que les informations contenues dans les traces sont correctes. Bien que cette hypothèse est vérifiée dans la plupart des cas, les traces peuvent contenir du bruit, c'est-à-dire, des informations mal écrites dans les logs.

Ces informations mal écrites peuvent créer deux types de problèmes : soit ajouter/retirer un évènement, ce qui peut induire une mauvaise modélisation, soit des modifications sur les informations, en particulier les dates des évènements. Par exemple, des évènements déclenchés en retard (à cause d'un défaut de réseau par exemple) vont inférer des règles de causalité faussées altérant le bon fonctionnement de l'algorithme de fouille.

C'est pour cela que l'algorithme de fouille a besoin d'être robuste au bruit : les relations causales ne doivent pas être basées sur une seule observation. Pour chaque observation, il faut pouvoir déterminer si c'est le chemin « normal » ou une exception.

Ce paramètre est particulièrement compliqué à régler, car si l'on se place dans une problématique de sécurité, on va chercher les « évènements rares » ou défaillances. Or ces évènements ont une fréquence d'apparition faible et peuvent être éliminés car étant considérés comme du bruit.

3.7.3 Traces incomplètes

Dans le cadre de systèmes complexes, ou d'observations trop réduites, il est peu probable que les traces contiennent tous les processus possibles. De même, certains processus peuvent avoir une faible probabilité d'apparaître dans les traces et ne seront probablement pas modélisés.

La complétude définit à quel point les traces sont représentatives du comportement permis par le modèle. Des traces sont complètes si elles représentent la totalité des comportements permis par le modèle ayant servi à les générer.

Les logs sont complets vis-à-vis d'un modèle au niveau des activités si toutes les activités présentes dans le modèle sont incluses dans les logs : $A(M) \subseteq A(L)$.

Cette limite est liée à la difficulté de trouver l'équilibre entre précision et généralisation.

3.7.4 Boucles

Dans un processus, il est possible d'effectuer la même tâche de multiples fois. Cela représente une boucle élémentaire dans le modèle. Les boucles contenant des structures simples (boucles élémentaires) peuvent être identifiées. Cependant, les boucles peuvent être utilisées pour revenir en arrière dans un processus (de Medeiros et al., 2004).

Si le processus est vraiment complexe, il peut être source d'ambiguïtés. Par exemple, comment distinguer une activité dupliquée et une boucle élémentaire.

3.7.5 Choix non-libres

Les constructions à choix non libre dans un réseau de Petri sont des constructions où deux transitions ne possèdent pas le même ensemble de places d'entrée mais où l'une d'entre elle possède une place d'entrée qui n'en est pas une pour l'autre (Van Der Aalst, Buijs and van Dongen, 2011). Par exemple, dans la figure 3.7 qui représente un choix non libre très simple, il est possible de voir un choix au niveau de la place C mais il n'est possible de fait t_3 que si t_1 a été effectué avant et pareillement avec t'_3 et t'_1.

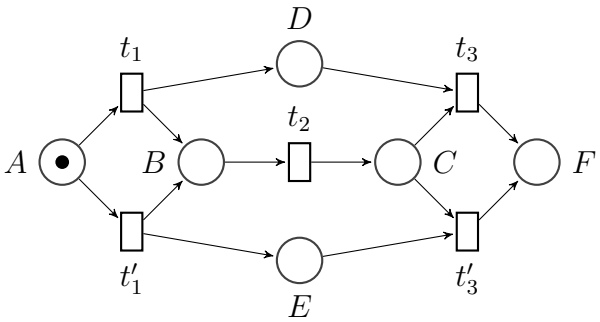

FIGURE 3.7 – Exemple de choix non libre

Les constructions à choix non-libres sont difficiles à modéliser, car elles représentent des choix contrôlés : les choix entre deux activités ne sont par exemple pas déterminés par un nœud du process tree mais par un choix qui a été fait en amont dans le processus. Les comportements non locaux ne sont pas évidents à miner et requièrent beaucoup d'observations.

3.8 Principaux algorithmes

Différents algorithmes existent pour la fouille de processus. Nous présentons les principaux ci-après.

3.8.1 Algorithme α

Cet algorithme a été proposé par Van Der Aalst et al. (2003) pour reconstruire les liens de causalité dans les Workflow Nets à partir des relations existantes dans les logs. Il ne peut repérer et modéliser ni les boucles, ni les choix non libres.

L'algorithme α (ou *Alpha miner*) génère des règles de précédence des activités depuis les traces, puis construit le modèle de processus correspondant aux règles générées. L'algorithme a une approche locale. Il ne considère que les relations entre chaque activité pour construire le modèle correspondant (voir section 3.5.1). Cependant, cette approche ne traite pas le bruit et connaît quelques difficultés à retrouver les dépendances éloignées temporellement. De plus, il l'algorithme ne peut pas traiter les boucles.

3.8.2 Algorithme $\alpha+$

de Medeiros et al. (2004) proposent une extension de l'algorithme α dans laquelle on qualifie les liens de causalité entre les tâches avec deux relations supplémentaires.

Soit T un ensemble de traces et, a et b deux activités présentes dans T. L'algorithme $\alpha+$ utilise les relations $\#$ et $>$ telles que définies dans le paragraphe 3.2.3 ;

Afin de découvrir les boucles, nous devons considérer deux relations supplémentaires :

— La relation de boucle simple (notée $\triangle$) représentant le fait qu'une activité se répète dans une boucle ; $a \triangle b \Leftrightarrow \exists t =< e_1, e_2, ...e_n >, t \in T \mid e_i = e_{i+2} = a \wedge e_{i+1} = b, 1 \leq i \leq n-2$

— La relation de boucle double (notée $\Diamond$) indiquant que deux boucles sont imbriquées ; $a \Diamond b \Leftrightarrow a \triangle b \wedge b \triangle a$

$\alpha+$ redéfinit les relations de parallélisme et de séquence pour considérer les boucles :

— $a \rightarrow b \Leftrightarrow b \wedge (b \not\succ a \vee a \Diamond b)$

— $a \parallel b \Leftrightarrow a \succ b \wedge b \succ a \wedge \neg(a \Diamond b)$

L'introduction de ces nouvelles règles permet de découvrir les motifs de type boucle de taille 2. Il est cependant nécessaire d'effectuer un pre-traitement sur les traces pour enlever les boucles de taille 1.

3.8.3 Algorithme $\alpha++$

L'algorithme $\alpha++$ étend l'algorithme $\alpha+$ dans le but de traiter les choix non libres (Van Der Aalst, 2016). Cet algorithme utilise les relations définies dans $\alpha+$ et en redéfinit certaines. $\alpha++$ ajoute la notion de relations implicites sous la forme de quatre règles :

— $\triangleleft$, précédence de choix. Si $a \triangleleft b$ alors il existe une activité qui une fois exécutée donne le choix entre a et b (XOR-split). $a \triangleleft b \Leftrightarrow a \sharp b \wedge (\exists c \in T | c \rightarrow a \wedge c \rightarrow b)$

— $\triangleright$, implication de choix. Si $a \triangleright b$ alors a et b sont en choix exclusif et il existe une activité qui succède à ce choix (XOR-join). $a \triangleright b \Leftrightarrow a \sharp b \wedge (\exists c \in T | a \rightarrow c \wedge b \rightarrow c)$

— $\gg$, deux activités sont en séquence sans se suivre directement. $a \gg b \Leftrightarrow a \not\succ b \wedge (\exists_{e \in t, t \in T} e \neq a \wedge e \neq b \wedge \neg(e \triangleright a \vee e \triangleleft a))$

— $\triangledown$, implication directe ou indirecte entre deux activités. $a \triangledown b \Leftrightarrow a \rightarrow b \vee a \gg b$

L'algorithme $\alpha++$ est décomposé en quatre étapes.

Étape 1 : modification des traces pour enlever les boucles de taille un.

Étape 2 : application de l'algorithme α sur ce nouvel ensemble de traces.

Étape 3 : calcul des dépendances implicites.

Étape 4 : calcul des arcs.

L'*Alpha miner* (et ses dérivés) a une approche bottom-up en minant les relations au niveau le plus fin, ce qui induit une partie des relations globales en reconstruisant le modèle. Il existe des techniques ayant une approche top-down en inférant les relations de plus petite granularité en partant de relations globales et en réduisant la granularité en subdivisant les logs.

3.8.4 *Heuristic Miner*

Cette technique est basée sur certaines règles de causalité de l'algorithme $\alpha++$. Elle calcule leurs fréquences d'apparition afin de définir la règle la plus adaptée. En effet, le score de chaque règle possible est fonction du nombre d'apparitions de la représentation de cette règle dans les traces. Cette technique peut détecter les séquences hors contexte mais cela nécessite une phase de configuration complexe (Weijters et al., 2006).

L'*Heuristic Miner* construit un graphe de dépendance en calculant, pour chaque activité, la probabilité qu'elle implique une autre activité du processus. Toutes les implications, ainsi calculées, dépassant le seuil choisi sont considérées pour construire le réseau de Petri correspondant. Il faut généralement effectuer une opération de réparation sur le modèle ainsi généré.

3.8.5 *Inductive Miner*

L'*Inductive Miner* est un algorithme qui permet de construire un *Process Tree* (Leemans et al., 2013 a). L'*Inductive Miner* a une approche globale sur la recherche des modèles, car il considère les relations entre des ensembles d'activités.

L'*Inductive Miner* cherche les relations entre évènements en découpant les traces récursivement en blocs et en générant les règles liant les sous-blocs ainsi découpés. Pour générer ces règles, l'*Inductive Miner* cherche quelle « découpe » utiliser en calculant la probabilité de chacune. La fonction de découpage change selon la variante de l'*Inductive Miner* utilisé.

3.8.6 *Inductive Miner Incompleteness*

Cette variante de l'*Inductive Miner* a pour but de parer au problème des séquences incomplètes dans les traces. Il n'est pas rare dans une application que certains processus ne soient pas terminés lorsque l'on procède à l'analyse (Leemans et al., 2014).

L'*Inductive Miner Incompleteness* calcule, pour chaque découpe, la probabilité de l'opérateur à appliquer en fonction du nombre d'apparitions de chaque évènement parmi les découpes possibles. Il applique ensuite la découpe ayant la plus forte probabilité d'être vraie.

Pour la sélection du découpage, l'*Inductive Miner Incompleteness* utilise les relations de l'*Inductive Miner* en complexifiant la relation de boucle pour mieux les identifier.

3.8.7 *Inductive Miner Infrequent*

L'*Inductive Miner Infrequent* filtre les traces sur les comportements observés peu fréquents selon un seuil choisi. Il utilise les mêmes opérateurs que l'*Inductive Miner* et découpe les traces de la même façon. À chaque étape, l'*Inductive Miner* va vérifier chaque évènement observé selon la fréquence d'apparition de celui-ci et ainsi pouvoir identifier les comportements peu fréquents. Ces évènements ne seront alors pas considérés pour décider de l'opérateur de découpe et ainsi éviter qu'un bruit produise une structure incohérente dans le modèle (Leemans et al., 2013 b).

L'*Inductive Miner Infrequent* se base sur l'algorithme *Inductive Miner Incompleteness* en appliquant en amont une série de filtres sur les traces. Le premier filtre s'applique directement sur les traces en entrée. Toutes les traces ayant un nombre d'apparitions trop bas comparé aux autres traces seront écartées. Le deuxième filtre s'applique sur le graphe de causalité. Celui-ci verra ses arcs pondérés selon leur nombre de représentation dans les traces, puis les arcs trop peu utilisés par rapport à l'utilisation des autres arcs seront supprimés.

Cet algorithme offre les propriétés les plus intéressantes vis-à-vis des traces que nous allons utiliser. En effet, outre le fait qu'il est reconnu comme étant le plus performant, il est capable de gérer des traces incomplètes et de filtrer en entrée les types de traces. Or notre approche

vise à extraire les processus des journaux d'évènement pour identifier les types de traces les plus performantes.

3.9 Conclusion

Dans ce chapitre nous avons présenté un état de l'art sur la fouille de processus. L'objectif de la fouille de processus est de découvrir les modèles de processus réalisés à partir des traces d'exécutions observées. Nous avons présenté les principes généraux, les modèles et algorithmes utilisés et les propriétés couramment utilisées pour caractériser la pertinence des modèles découverts.

Nous avons décrit les algorithmes de fouilles en commençant par donner les grands principes : identification des relations entre deux évènements (en séquence, parallèle, choix ou succession immédiate), et prise en compte de la fréquence d'apparition d'une relation de causalité. À partir de ces relations il est possible de bâtir des raisonnements pour construire les modèles de processus.

Nous avons ensuite présenté les propriétés des modèles minés afin d'évaluer leur qualité vis-à-vis des traces observées. En effet, le résultat de la fouille de processus est un modèle qui décrit au mieux le comportement observé. Les mesures classiques sont : la justesse qui exprime le fait que le modèle découvert correspond bien au logs ; la précision qui caractérise le fait que le modèle ne que les comportements observés dans les logs ; la généralisation qui, au contraire, décrit la capacité à prendre en compte d'autres comportements ; et la simplicité qui fournit un indicateur sur la complexité du modèle obtenu. Trouver un bon modèle ne consiste pas à obtenir le résultat qui maximise tous les critères, certains sont antagonistes, mais à trouver l'équilibre entre ces critères.

Fort de ces informations, nous avons présenté les limites de la fouille de processus. C'est-à-dire que nous avons présentés les conditions qui font qu'il est difficile d'obtenir un modèle, comme l'hétérogénéité des données, le bruit ou le fait que les cas observés ne représentent pas des processus en entier (on démarre et coupe toujours arbitrairement l'enregistrement des données et des processus peuvent avoir déjà commencé ou être en cours d'exécution). Et nous avons montrer les formes compliquées à extraire comme les choix non-libres et les boucles.

Nous finissons par présenter les principaux algorithmes utilisés dans le cadre de la fouille de processus. Nous avons commencé par présenter les algorithmes α et leurs dérivés. Il s'agit des premiers algorithmes qui ont le mérite de permettre une bonne compréhension des principes et limitations. Ces algorithmes sont basés sur les relations entre événements. Nous avons ensuite présenté les algorithmes basés sur le découpage récursive des traces. Il s'agit des algorithme de la famille « *Inductive Miner* ».

Cet état de l'art nous apporte une bonne connaissance du deuxième domaine que nous devons utiliser pour répondre à notre question de recherche. À savoir, est-il possible de définir

une méthodologie combinant trace et fouille de processus comme entrée d'un système de recommandation ? Pour cela nous devons analyser les caractéristiques des processus impliqués dans les systèmes que nous étudions (voir chapitre 5). Comme il s'agit de processus construits par l'utilisateur avec une grande souplesse laissée à ce dernier nous nous attendons à avoir des variations et de nombreux rebouclages.

S'il est difficile de caractériser *a priori* nos processus, nous pouvons quand même en déduire que, dans le cadre de cette thèse, l'algorithme *Inductive Miner Incompleteness* est celui qui répond le mieux aux caractéristiques de nos processus. Mais dans un contexte de généralisation, il est pertinent de sélectionner l'algorithme de fouille de processus en fonction des caractéristiques des traces comme dans Augusto et al. (2017); Jouck et al. (2018)

Nous allons présenter dans le chapitre suivant la problématique de la recommandation en général, en nous concentrant plus particulièrement sur la recommandation de processus.

CHAPITRE 4 | Recommandation

Sommaire

4.1 Introduction 54
4.2 Les systèmes de recommandation 54
4.3 Terminologie et concepts 55
4.3.1 Définitions 56
4.3.2 Entrées d'un système de recommandation 57
4.3.3 Sorties d'un système de recommandation 57
4.4 Classification des systèmes de recommandation 58
4.4.1 Recommandation collaborative ou par filtrage collaboratif 58
4.4.2 Recommandation basée sur le contenu 58
4.4.3 Filtrage hybride 60
4.5 Calcul de similarité 61
4.6 Entraves à la recommandation 62
4.7 Bilan 63
4.8 Conclusion 64

4.1 Introduction

L'utilisation de plus en plus grande de dispositifs informatiques génère un nombre important de traces. Une telle profusion d'information a aiguisé les appétits et de nombreux projets de recherche se sont développés afin d'exploiter ses données pour analyser/prédire un comportement.

Les systèmes de recommandation sont des outils permettant d'interagir avec des grandes masses de données. Ils fournissent une vue personnalisée de l'information en privilégiant les éléments susceptible d'apporter des connaissances supplémentaires à l'utilisateur. Ce domaine d'analyse a connu un essor important et donne lieu à une grande variété de technique d'intelligence artificielle (apprentissage automatique, extraction de données, modélisation utilisateur, *etc.*).

L'objectif du chapitre est de donner une vue générale des systèmes de recommandation. Nous commençons par les définitions de base, puis présentons les composantes des systèmes de recommandation, soit la composante de filtrage et la composante de calcul de similarité. Nous justifions ensuite ce que nous réutilisons tout en identifiant les composantes à développer pour nos travaux.

La suite du chapitre présente une synthèse de l'état de l'art dédié aux systèmes de recommandation. La section 4.3 définit les termes rencontrés, les données manipulées ainsi que les sorties. La section 4.4 présente l'ensemble des méthodes de filtrage des systèmes de recommandation et les méthodes d'hybridation. La section 4.5 résume les méthodes pour calculer la pertinence des objets recommandés. La section 4.7 compare des méthodes utilisables dans notre contexte.

4.2 Les systèmes de recommandation

Les systèmes de recommandations (SR) réduisent le temps de recherche de l'utilisateur et lui suggèrent des objets auxquels il aurait pu ne pas faire attention. Les SR ont été développés en parallèle du web. Ils étaient initialement basés sur des filtrages comme les filtres démographiques (comme l'âge, la nationalité ou le sexe des utilisateurs), à base de contenu (comme les résumés d'objets en vente, par exemple le SR de Amazon) ou collaboratif (comme les appréciations des autres utilisateurs, par exemple SR de Youtube) comme cela a été décrit dans (Pazzani, 1999).

Récemment, l'utilisation des SR dans Internet a fortement augmenté, ce qui a facilité leur utilisation dans divers domaines comme l'éducation, le commerce, les loisirs, *etc*. (Park et al., 2012) Des travaux de recherches dans le domaine de la recommandation peuvent être trouvés dans le domaine du cinéma (Carrer-Neto et al., 2012; Winoto and Tang, 2010) comme les SR de IMDB et de Rotten Tomatoes, de la musique (Nanopoulos et al., 2010; Tan et al., 2011;

Katarya and Verma, 2018) comme les SR de Deezer ou de Spotify, de la télévision (Yu et al., 2006; Barragáns-Martínez et al., 2010) comme les SR des chaînes VOD (*Video On Demand*), des livres (Núñez-Valdéz et al., 2012; Crespo et al., 2011; Mooney and Roy, 2000) comme les SR de LIBRA et de Gleeph, des documents (Serrano-Guerrero et al., 2011; Porcel et al., 2012) comme les SR de Hal et de Research Gate, du e-learning (Zaíane, 2002; Tarus, Niu and Mustafa, 2018; Bobadilla et al., 2009) comme les SR des MOOC (*Massive Open Online Course*), du e-service (Huang et al., 2007; Castro-Schez et al., 2011; Wu et al., 2015) comme les SR de Amazon ou de Ebay, des applications (Costa-Montenegro et al., 2012; Yang et al., 2016) comme les SR du Play Store et de l'Apple Store, des moteurs de recherche (McNally et al., 2011) comme les SR de Google ou des pages jaunes et encore beaucoup d'autres domaines. La recommandation a beaucoup de domaine d'application et nous nous intéressons, tout particulièrement, au domaine de l'accompagnement des utilisateurs. L'utilisation des algorithmes de filtrage ou des méthodes de calculs de similarité dans ce domaine permettent l'identification des profils utilisateurs (Herlocker et al., 2004) ou la pertinence des objets recommandés (Breese et al., 1998; Kluver et al., 2018).

Cependant, la complexité du domaine fait qu'il est parfois difficile de trouver un résultat pertinent en n'utilisant qu'une seule des méthodes du domaine (Ricci et al., 2015). Pour cela, il est important d'étudier les techniques hybrides, mixant plusieurs techniques de recommandation afin d'obtenir les avantages de chacune d'elles. Par exemple, il est possible de combiner deux types de filtrage comme le filtrage collaboratif et à base de contenue (Melville et al., 2002). Une étude approfondie portée sur les SR hybrides a été présentée dans (Burke, 2002).

Enfin, certaines recherches (Adomavicius and Tuzhilin, 2005; Ricci et al., 2015) présentent des synthèses du domaine des SR notamment les techniques à base de modèle. De telles techniques utilisent un modèle de données pour calculer des recommandations (Levandoski et al., 2012), comme par exemple un modèle de l'organisation des activités dans un processus. Dans (Lee et al., 2013), les auteurs construisent un modèle de données permettant de classer les informations extraites de sources de données hétérogènes afin d'effectuer des recommandations. Ce modèle de données est ensuite enrichi en fonction du type de recommandation à effectuer. Par exemple, si le modèle de données est un graphe où chaque noeud est une musique et où les liens sont les genres similaires entre les musiques. Lors d'une recommandation de musique, on enrichit le modèle avec les informations des films pour ajouter des liens exprimant l'apparition des musiques dans un même film. Cela permet de recommander des musiques en fonction de leur apparition dans un film.

4.3 Terminologie et concepts

Cette section présente la définition des concepts inhérents aux mécanismes de recommandation.

4.3.1 Définitions

Le domaine de recommandation possède un vocabulaire qui lui est propre permettant de définir les éléments qui sont utilisés.

Système de Recommandation : Ensemble d'algorithmes permettant d'associer plusieurs objets entre eux par similarité de façon personnalisée. Un système de recommandation est majoritairement constitué d'un système de filtrage, comme un filtrage collaboratif, et d'un calcul de similarité, permettant l'obtention d'un score caractérisant la ressemblance entre des objets. Un système de recommandation (SR) permet d'associer tous types d'objets (Linden et al., 2003).

Objet : Ce terme est la traduction de « *item* » en anglais qui, dans le contexte des systèmes de recommandation, désigne une entité qui est traitée par le système, comme un utilisateur, une ressource ou les articles d'un magasin (Linden et al., 2003).

Profil : Toute information concernant un objet tel que son nom, sa description ou toutes autres information propre aux sous-catégories d'objets. Dans ces sous-catégories, on peut trouver, tout aussi bien, les utilisateurs (on parle alors de profil utilisateur) ou les articles, comme des livres. D'un point de vue pratique, on peut voir le profil comme la caractérisation de l'objet considéré à l'aide d'un ensemble d'attributs organisés selon un modèle (Pazzani and Billsus, 2007).

Profil utilisateur : Il s'agit des caractéristiques et des préférences de l'utilisateur, de ce qui l'intéresse comme objet et ce qu'il n'aime pas. On peut aussi garder dans ce modèle les historiques de ce qu'il a effectué comme activités (ses achats par exemple, ou les pages qu'il a consultées le plus). Lorsque le système de recommandation est basé sur ce profil pour faire ses recommandations, on parle alors de système de recommandation à base de confiance (Shani and Gunawardana, 2011).

Attributs d'objets (document, article, ressource) : Tout comme l'utilisateur, un objet est défini par des caractéristiques. Par exemple, dans la recommandation de processus, les objets sont les exécutions du processus. Chaque exécution est définie par ses activités, l'ordre de celles-ci, les ressources utilisées, l'utilisateur, *etc*. Lorsqu'un système de recommandation est basé sur les objets pour ses recommandations alors on parle de système de recommandation à base de contenu (Pazzani and Billsus, 2007).

Note : Une note est une valeur numérique représentant l'appréciation d'un utilisateur pour un objet. Elle peut être récupérée implicitement ou donnée par l'utilisateur (Shani and Gunawardana, 2011).

Score : Le score est une valeur donnée à deux objets afin d'évaluer leur similarité. Par exemple, plus deux livres auront des genres similaires plus ils auront un score élevé. Il est souvent calculé par le système grâce à une méthode de calcul de similarité (Leblay, 2016).

4.3.2 Entrées d'un système de recommandation

La première étape pour une recommandation pertinente est la collecte de données. Elles sont extraites des actions observables par les systèmes qui ont une certaine relation avec ce que l'on cherche à recommander. Nous distinguons deux formes de collecte de données :

Collecte de données explicite : L'utilisateur effectue une action dont le but est de dire, soit au système, soit aux utilisateurs du système, ce qu'il pense d'un objet particulier. On demande à l'utilisateur par exemple de commenter, taguer, noter, « liker », suivre les objets qui l'intéressent. Beaucoup de systèmes utilisent une échelle de notation allant de 1 (je n'aime pas du tout) à 5 (j'aime beaucoup) étoiles.

Collecte de données implicite : L'utilisateur n'effectue pas d'action afin de communiquer ses préférences, mais nous pouvons apprendre quelque chose sur ses préférences en observant et analysant ses comportements sans rien lui demander. Par exemple, obtenir la liste des vidéos que l'utilisateur a regardées pour un système de recommandation de vidéos ou analyser le temps qu'il a passé dans l'exécution d'un processus pour la recommandation d'exécution de processus.

Les deux formes de collecte ont des avantages et des inconvénients. La collecte explicite recueille des informations précises sur ce qu'on veut savoir. Elle peut contenir le biais de la déclaration et impose une interrogation qui, si trop fréquente, pourrait agacer l'utilisateur. En revanche, dans la collecte implicite, toutes les informations sont collectées automatiquement mais ces informations ne sont pas forcément fiables parce qu'un utilisateur peut utiliser un objet qui ne le satisfera pas après, comme une exécution qui n'atteint jamais l'objectif du processus.

4.3.3 Sorties d'un système de recommandation

Un système de recommandation peut être utilisé pour différents objectifs.

La prédiction : une estimation de la satisfaction d'un utilisateur envers un objet, souvent réalisée à l'aide d'une échelle d'évaluation.

La suggestion : une suggestion des objets que l'utilisateur aime (ou qu'il aime peut-être d'après ses actions et son comportement), souvent organisée sous forme d'une liste top-k (les k meilleurs choix) présentée à l'utilisateur.

Une relation entre la suggestion et la prédiction est que la suggestion requiert le calcul prédictif des notes pour chaque objet potentiellement intéressant, et pas forcément la totalité des objets. Afficher les prédictions uniques des objets peut donc mener à un calcul de notes plus complexe qu'un affichage de suggestions.

4.4 Classification des systèmes de recommandation

Il est possible de classer les systèmes de recommandation de différentes manières. La classification la plus répandue est de distinguer la recommandation collaborative et la recommandation basée sur le contenu (Adomavicius and Tuzhilin, 2005; Bobadilla et al., 2013).

4.4.1 Recommandation collaborative ou par filtrage collaboratif

Le Filtrage Collaboratif (FC) (Sun et al., 2019; Ricci et al., 2011) est la technique la plus populaire et la plus répandue dans les systèmes de recommandation. Le principe général est de demander aux utilisateurs d'évaluer des ressources, de sorte à savoir ce qu'ils aiment le plus. Puis, quand une recommandation est demandée pour l'utilisateur courant, lui seront alors proposées des ressources que des utilisateurs, semblables à lui, ont aimé.

Ainsi le filtrage collaboratif fonde ses prédictions et recommandations sur les notes ou le comportement d'autres utilisateurs dans le système. L'hypothèse fondamentale est que les opinions des autres utilisateurs peuvent être sélectionnées et regroupées de manière à fournir une prédiction raisonnable des opinions des utilisateurs traités. Si plusieurs utilisateurs, qui ont un profil ou une liste d'opinions similaires avec l'utilisateur traité, ont une opinion sur un objet (musique, film, livre ou même un autre utilisateur) que l'utilisateur traité n'a pas encore vu/consulté alors il est possible que celui-ci ait la même opinion que les autres. Lorsque le FC est utilisé avec les objets plutôt que les utilisateurs, on l'appelle FC à base d'objets. Les objets sont considérés comme approchants si un utilisateur les a appréciés. Dans ce cas, la similarité entre objets est basée uniquement sur le jugement des utilisateurs (Linden et al., 2003). Pour le FC à base d'objets, la représentation d'un objet peut se limiter à son identifiant. Si on prend un exemple de recommandation de livres, le système de recommandation infère que deux livres sont voisins si les utilisateurs ont apprécié ces deux livres. Par ailleurs, les approches basées sur le FC sont, bien souvent, mixées avec d'autres techniques comme le filtrage à base de contenu (Alchiekh Haydar, 2014).

4.4.2 Recommandation basée sur le contenu

Dans le cas de la recommandation basée sur le contenu (Sun et al., 2019), le système recommande des objets qui sont similaires à ceux que l'utilisateur a aimé dans le passé. La similarité des objets est calculée en se basant sur les caractéristiques associées aux objets comparés. Par exemple, si l'utilisateur a noté positivement un film qui appartient au genre comédie, alors le système peut fournir des recommandations de films de ce genre.

La recommandation basée sur le contenu repose sur l'hypothèse que des objets ayant des contenus similaires seront appréciés pareillement (Schafer et al., 2007). Les objets qui peuvent être recommandés aux utilisateurs sont représentés par un ensemble de caractéristiques ou

méta-données : attributs, variables ou propriétés, *etc*. Par exemple, dans une application de recommandation de films, les attributs adoptés pour décrire un film sont : acteurs, réalisateurs, genres, sujet, *etc*. Un objet est représenté dans le système au moyen d'une donnée structurée. Plus formellement, cette donnée structurée est un vecteur $[x_1, ..., x_n]$ de n composantes. Chaque composante représente un attribut et peut contenir des valeurs binaires, numériques ou textuelles.

Ainsi, le processus principal réalisé par un système de recommandation basé sur le contenu consiste à faire correspondre les attributs d'un profil utilisateur avec les attributs des objets d'un contenu, le but étant de recommander à l'utilisateur de nouveaux objets intéressants pour lui.

Plusieurs mises en œuvre de la recommandation à base de contenu existent (Bobadilla et al., 2013). Nous présentons ci-après deux mises en oeuvre proches, au niveau des outils utilisés, de la méthodologie que nous proposons dans ce manuscrit : la recommandation basée sur un modèle et la recommandation basée sur un modèle de données enrichi.

4.4.2.1 Recommandation basée sur un modèle

Les systèmes basés sur les modèles sont un cas particulier des systèmes basés sur le contenu. Ils tendent à englober dans des modèles de données les informations dont ils disposent afin de remédier à certains manques des systèmes basés sur le contenu classique (Di Noia et al., 2012; Huang et al., 2002). Il existe plusieurs modèles sur lesquels se basent les SR à base de modèle comme les réseaux de Petri présentés dans le chapitre 3, les réseaux bayésiens ou les réseaux markoviens.

Les réseaux bayésiens (*Bayesian Networks*), par exemple, permettent de calculer la probabilité qu'un objet soit le suivant en fonction de ce que l'utilisateur a déjà traité. Il s'agit d'un graphe pondéré possédant la capacité de voir les actions déjà effectuées (He and Chu, 2010). L'avantage des réseaux bayésiens est qu'ils permettent de recommander des objets en prenant en compte les tendances de l'utilisateur. Par exemple, dans le cas de l'accompagnement d'un utilisateur dans un processus, selon les précédents choix d'activités que l'utilisateur a effectué, nous calculons la probabilité qu'il soit sur une trajectoire inadaptée afin de pouvoir lui recommander un recadrage le cas échéant. Il existe une forme de ces réseaux qui ne prend en compte que le dernier objet traité, on parle alors de réseaux markoviens (Breese et al., 1998).

4.4.2.2 Recommandation à base de contexte

Parfois utiliser un modèle classique n'est pas suffisant et il est nécessaire de l'enrichir avec des connaissances complémentaires pour l'adapter au cas d'application considéré. Les méthodes utilisant ces modèles enrichis sont appelées méthodes à base de contexte (*context-aware methods*) (Lee et al., 2010). Ces méthodes se servent des données du contexte de plusieurs façons, par exemple en récupérant un modèle de graphe pondéré où chaque arc est la proximité

entre les objets représentés par les nœuds de ses extrémités (Soualah-Alila, 2015). Ainsi, une méthode de recommandation utilisant un modèle comme un workflow-net serait considérée comme une méthode à base de modèle. Les outils pour calculer le score d'une recommandation pour ce type de méthode sont alors dépendants du modèle utilisé, comme détaillé dans la section 4.5.

Plusieurs recherches utilisent des méthodes d'enrichissement à base de connaissances pour prendre en compte des exceptions qui n'étaient pas prises en compte par le modèle de données. Pour ne citer que le plus similaire à notre approche, les recherches de Oak Ridge National Laboratory (ORNL) (Adomavicius and Tuzhilin, 2015) utilisent l'enrichissement pour associer, à un calcul de recommandations, des connaissances liées au contexte (localisation, date...). Cela permet d'influencer la recherche dans leur modèle de données hétérogènes en exploitant les connaissances associées au contexte des requêtes envoyées au système. Ainsi, il est possible d'effectuer un enrichissement d'un modèle de processus avec les connaissances qu'on a de ce processus. On peut par exemple identifier l'état du processus dans lequel l'objectif est atteint et les modifications apportées à l'état du processus par les différentes activités dans l'exécution de ce processus.

4.4.3 Filtrage hybride

Le filtrage hybride n'est pas une approche en soi, mais plutôt la combinaison de plusieurs approches de recommandation. Normalement, pour prédire une note, on calcule un score de similarité entre les objets qui quantifie leur rapprochement. En filtrage hybride, le calcul de similarité est effectué par toutes les approches considérées, ce résultat est appelé « score local ». Puis les résultats sont fusionnés pour donner le « score final ». Cela étant, il est aussi possible d'appliquer la méthode d'hybridation à la fonction de similarité. Malgré le grand nombre de méthodes, il est possible de les rassembler en sept groupes (Sun et al., 2019; Burke, 2002), comme suit :

1. *Weighted*, ou Pondérée, est une technique qui combine numériquement les scores venant de plusieurs techniques de recommandation (comme les méthodes de calcul de similarité). Elle est un moyen simple pour une hybridation car l'ajustement des paramètres de pondération permet de quantifier la contribution de chaque approche.

2. *Switch*, ou Alternance, est une technique qui exploite des paramètres qui lui sont donnés comme le profil de l'utilisateur ou les valeurs descriptives d'un produit, afin d'alterner entre plusieurs techniques de recommandation et de choisir la plus efficace dans le contexte considéré.

3. *Mixed*, ou Mixte, est une technique qui utilise simplement chaque méthode qui lui est associée pour générer une liste d'entités propices à être recommandées. Puis la liste

passe dans chaque système mixé pour obtenir différentes listes classées. Enfin, ces listes sont fusionnées pour obtenir un classement des entités potentiellement appréciées.

4. *Feature Contribution*, ou Assimilé, est une technique dont le but est de fusionner les SR. Chaque attribut qui n'est normalement pas pris en compte par le SR sera pris en compte par un autre SR, ce qui permet de prendre en compte un panel d'entités beaucoup plus large qu'avec un seul SR.

5. *Feature Augmentation*, ou Contributeur, est une technique où le but est d'ajouter des SR à un autre, qui sera désigné comme principal, pour l'aider dans sa décision. L'idée est de pondérer les valeurs d'entrées appréciées des contributeurs pour les exposer à la critique du SR principal.

6. *Cascade* est une hybridation faite pour les SR utilisant un score de sortie probabiliste. Le SR principal va déterminer la probabilité d'appréciation des entités et les SR secondaires vont se suivre les uns les autres pour affiner les résultats. Les SR secondaires ne peuvent pas changer le classement du SR principal.

7. *Meta-Level*, ou Méta niveau, est une approche assez semblable à l'approche *Feature Augmentation*. Plusieurs SR s'enchaînent pour créer une liste de valeurs acceptables pour être recommandées. Puis, le SR final récupère cette liste pour faire sa recommandation.

4.5 Calcul de similarité

Le calcul de similarité (Zhang et al., 2019) permet de créer un score qui quantifie la ressemblance entre deux objets compte tenu des caractéristiques observées. Il existe plusieurs méthodes pour trouver ce score. Comme par exemple :

— L'algorithme de Pearson permet de trouver la corrélation entre deux variables. Il s'agit d'un calcul de covariance sur le produit des variances et permet d'obtenir un score de similarité compris entre 1 et -1, avec 1 pour une ressemblance parfaite, 0 aucune corrélation et -1 une totale différence. L'algorithme de Pearson est souvent utilisé avec le filtrage collaboratif.

— La similarité cosinus (ou mesure cosinus) permet de calculer la similarité entre deux vecteurs à n dimensions en déterminant le cosinus de l'angle entre eux. La similarité entre deux vecteurs s'obtient par le produit scalaire et la norme des vecteurs.

— Le TF-IDF (*Term Frequency-Inverse Document Frequency*) est une méthode de pondération. Cette mesure statistique permet, par exemple, d'évaluer l'importance d'un terme contenu dans un document en calculant son nombre (ou sa fréquence) d'occurrences.

Il est possible de définir son propre calcul de score en fonction du modèle considéré. Par exemple, la recherche de chemin (*Path-Finding*) est une méthode permettant de trouver un chemin dans un modèle place-transition ou nœuds-arc respectivement comme les workflow-nets ou les graphes. Le score se calcule, soit en fonction de la pondération des arcs ou transitions, soit en fonction du nombre d'arcs ou de transitions franchis. Les algorithmes de Path-Finding ressortent une liste de chemins permettant, depuis un nœud ou une place de départ, d'atteindre un nœud ou une place d'arrivée. Les chemins de cette liste sont classés par leur nombre de sauts ou par la somme des pondérations des arcs ou des transitions qu'ils ont traversées. Il existe deux algorithmes très populaires pour le Path-Finding : l'algorithme A^* et l'algorithme du plus court chemin de Dijkstra (Heineman et al., 2016).

4.6 Entraves à la recommandation

Le problème majeur du système de recommandation est le manque de données. Pour le filtrage collaboratif par exemple, moins il y a d'utilisateurs moins les recommandations seront précises. Selon notre état de l'art, ce problème, appelé problème de démarrage à froid, peut se manifester de plusieurs manières comme détaillé plus bas. Il existe également d'autres problèmes que celui du manque de données, nous détaillons ci-après les problèmes les plus récurrents des différentes stratégies de recommandation.

Niches cross-genres : Le filtrage collaboratif se distingue par sa capacité à recommander à un utilisateur ce qui est hors du familier, c'est ce que Burke (2002) appelle niches cross-genres. En effet, un utilisateur peut se voir recommander des objets de genres différents. Par exemple, un utilisateur qui a des voisins similaires du point de vue des sports peut se voir recommander des recettes de cuisine si ces voisins aiment la cuisine et les recettes, même si cet utilisateur lui-même n'a jamais exprimé ce genre de favoris.

Problème de démarrage à froid, cas de nouveaux utilisateurs : Un problème commun au filtrage basé sur le contenu et au filtrage collaboratif est qu'un nouvel utilisateur qui n'a pas encore accumulé suffisamment d'évaluations ne peut pas avoir de recommandations pertinentes (Alchiekh Haydar, 2014).

Problème de démarrage à froid, cas d'un nouvel objet : C'est un problème qui concerne le filtrage collaboratif, et non pas le filtrage à base de contenu. Dans le cas du filtrage à base de contenu, il suffit d'introduire l'objet dans le système pour que celui-ci soit analysé et rentré dans le processus de recommandation. Dans le cas du filtrage collaboratif, il doit avoir suffisamment d'évaluations pour que celui-ci soit pris en considération dans le processus de recommandation (Zhang et al., 2010).

Problème de démarrage à froid, cas du système débutant : Le cas du système débutant survient lors du lancement d'un nouveau service de recommandation. Le système ne

possède alors aucune information sur les utilisateurs ni sur les objets. Les méthodes de filtrage collaboratif ne peuvent pas fonctionner sans ces informations. La solution consiste en général à trouver des informations descriptives des objets afin d'organiser le catalogue et inciter les utilisateurs à le parcourir jusqu'à ce qu'il y ait suffisamment de données pour permette de passer en mode collaboratif (Bobadilla et al., 2013).

Le *Gray Sheep* : Quand des utilisateurs ont des goûts atypiques (qui varient de la norme), ils n'auront pas beaucoup d'utilisateurs en tant que voisins. Cela mènera à des recommandations pauvres. Ce problème est également connu sous le nom de *gray sheep*. Il se produit fréquemment dans le filtrage collaboratif (Gras and Boyer, 2016).

Le détournement : C'est l'action malveillante d'influencer la recommandation. Par exemple, en créant de faux profils pour voter et favoriser/défavoriser certains objets (Bouraga et al., 2014).

4.7 Bilan

Pour les besoins de l'accompagnement des utilisateurs, plusieurs questionnements de recherches ont été identifiés :

— Quelles sont les contraintes pour que le modèle représentatif de l'organisation des activités présentes dans les traces soit utilisable pour calculer des recommandations permettant d'accompagner l'utilisateur dans un processus ?

— Comment garantir la pertinence des recommandations ?

L'utilisation des traces et de la fouille de processus, nous donne la possibilité de nous servir d'un modèle représentatif de l'organisation des activités dans le processus étudié pour baser nos recommandations. Pour utiliser ce modèle, nous utilisons les méthodes de recommandation à base de modèle ce qui n'est, à notre connaissance, pas faisable en utilisant les méthodes de filtrage collaboratif ou à base de contenu. Notre modèle étant fondé sur les réseaux places-transitions, nous utiliserons les outils du Path-Finding comme proposé dans (Van Der Aalst et al., 2012).

Plusieurs outils permettant de trouver un chemin dans un modèle de processus existent, comme l'algorithme A^* ou l'algorithme du plus court chemin de Dijkstra évoqués précédemment. L'algorithme de Dijkstra se positionne sur la pertinence de ses résultats où le premier résultat trouvé sera forcément le chemin le plus optimisé pour atteindre depuis un point de départ un point d'arrivée. L'algorithme A^* se positionne sur son temps de calcul, car pour un temps de calcul inférieur à celui de Dijkstra, celui-ci permettra de trouver une des meilleures solutions à condition qu'une heuristique soit définie pour la recherche d'une solution.

Dans notre cas, le temps de calcul nous importe moins que la pertinence de la recommandation comme cela a été défini dans le chapitre 1. Pour cette raison, dans la mise en place

de notre méthode, nous utiliserons l'algorithme de Dijkstra afin de garantir la pertinence des recommandations.

Lors de la recommandation, il est possible d'utiliser un modèle (section 4.4.2.1). Ainsi, il est possible d'utiliser un modèle de processus. Cependant, la représentation des processus possède plusieurs cas particuliers, comme les processus non ou faiblement structurés dans lesquels il n'y a aucune ou peu de corrélation entre les activités. Selon notre état de l'art, il existe des méthodes permettant de prendre en compte les modèles de processus qui représentent ces cas particuliers. Deux solutions s'offrent à nous : l'hybridation et l'enrichissement.

Hybridation : Par l'utilisation d'un système de recommandation hybride (Burke, 2007), comme le switch (*cf.* paragraphe 4.4.3), il est possible de mixer des méthodes de recommandation qui fonctionnent selon le type de processus (par exemple les processus non structurés) (Tarus, Niu and Kalui, 2018). Cela permet d'utiliser une méthode ou une autre en fonction du type de processus et ainsi essayer de rester le plus pertinent possible dans les recommandations.

Enrichissement : Via l'utilisation d'un enrichissement, il est possible d'ajouter au modèle des connaissances sur le processus, comme l'impact des transitions sur l'état du processus, afin de transformer le modèle workflow-net en un même modèle avec prédicat, comme les modèles prédicat-transition (Desel and Esparza, 2005). Une fois enrichi, le modèle permettra de représenter la majorité des types de processus y compris les processus non structurés.

4.8 Conclusion

Nous venons de faire une présentation des systèmes de recommandation. Nous avons commencé par introduire l'architecture globale de tels systèmes en précisant les termes utilisés. Nous avons ensuite proposé une classification des systèmes de recommandation en fonction des informations prises en compte pour faire une recommandation. Nous poursuivons ce chapitre par une présentation des mesures utilisés pour comparer des informations. Ce chapitre se termine par une présentation des limitations et difficultés de mise en œuvre.

Ce chapitre termine notre état de l'art. Nous avons étudié les systèmes à base de traces dans le chapitre 2. Nous avons vu les différentes phases : collecte, transformation et exploitation. Nous retenons les phases de collecte pour faire l'acquisition des informations nécessaires à notre étude et de transformation afin de construire des *event logs* (c'est-à-dire que nous regroupons toutes les informations d'un processus comme une suite d'événements). Ces *event logs* constituent l'entrée de notre phase d'exploitation : extraction des informations sur les processus métier en utilisant les algorithmes de la fouille de processus. C'est l'objet du chapitre 3. Nous y avons étudié le domaine de la fouille de processus afin d'étudier les algorithmes

couramment utilisés, les contraintes et les limites. Ces algorithmes ont été développés dans le cadre de systèmes d'information avec des processus métier fortement structurés et où peu de libertés est laissée à l'utilisateur.

Dans le chapitre suivant, nous proposons une méthodologie pour articuler ses différents domaines dans le but de construire un compagnon pour aider un utilisateur lors de l'exécution d'un processus. Notre objectif est de démontrer la faisabilité de bout en bout d'une telle approche.

CHAPITRE 5 | Proposition d'une méthodologie pour la recommandation d'activités dans des processus métier

Sommaire

5.1 Introduction 68
5.2 Principe de recommandation dans les processus 69
5.3 Méthodologie 71
5.3.1 Extraction des traces modélisées 71
5.3.2 Modèle de processus métier réalisé 74
5.3.3 Proposition de recommandation pour l'utilisateur 75
5.4 Validation de l'architecture proposée 76
5.4.1 Description du cas d'étude 77
5.4.2 Format des données 78
5.4.3 Application de la méthode 78
5.4.4 Discussion 79
5.5 Conclusion 81

5.1 Introduction

L'objectif de nos travaux est de proposer une méthodologie qui permet d'accompagner l'utilisateur durant l'exécution d'un processus. C'est-à-dire que dans le cadre de la supervision de processus, nous cherchons à proposer à un utilisateur l'activité (ou un ensemble d'activités) lui permettant d'atteindre son but en respectant des mesures de qualité. Nous nous plaçons dans le contexte de processus qui sont majoritairement pilotés par l'utilisateur. Comme par exemple l'achat via un site de commerce en ligne ou l'utilisation d'un site Web pour remplir un formulaire administratif. Nous sommes donc dans des cas où les processus sont définis par un expert et l'on cherche à encadrer les actions de l'utilisateur pour avoir une exécution satisfaisante.

Cette problématique ouvre de nombreuses questions de recherche et problématiques. Nous avons choisi de nous concentrer sur la définition d'une méthodologie pour faire des recommandations dans des processus métier. Plus particulièrement, nous avons choisi d'exploiter les traces d'exécutions pour mener des analyses sur les processus et proposer un mécanisme de recommandation à partir des processus observés. La question que nous cherchons à résoudre est : est-il possible d'extraire de la connaissance sur les processus métiers pour construire un raisonnement afin de faire une recommandation d'activité ?

Nous avons fondé nos travaux sur l'hypothèse qu'il est possible de construire une recommandation à partir des processus métiers observés. Nous avons donc construit notre état de l'art suivant trois directions : les systèmes à base de traces, la fouille de processus et la recommandation. Dans le chapitre 3 nous avons étudié la fouille de processus. Il s'agit de démarches et d'algorithmes qui permettent d'extraire de la connaissance sur les processus métier d'un système d'information. À partir de traces modélisées sous la forme d'*event logs* normalisé par le format *eXtensible Event Stream* (XES), abordé dans le paragraphe 2.4.1.2, un algorithme de fouille de processus construit un modèle des processus métier. Ce modèle est basé sur les réseaux de Petri ou sur une extension (*workflow net* ou *process tree*). La pertinence du modèle découvert est donnée par des mesures de qualité : la justesse, la précision, la généralisation et la simplicité. Nous avons également étudié les contraintes et limites d'utilisation des algorithmes de la fouille de processus. Cette étude nous permet d'identifier notre originalité vis-à-vis de ce domaine : les processus que nous étudions sont faiblement structurés et en forte interaction avec l'utilisateur.

Nous en avons donc déduit les informations que l'on doit posséder pour utiliser les algorithmes de la fouille de processus. Lors de notre étude des systèmes à base de traces, chapitre 2, nous avons vu comment mettre en place des sondes pour recueillir les données et comment les transformer en traces modélisées exploitables par les algorithmes d'extraction de l'information. En particulier, il est nécessaire d'avoir les événements et leur horodatage. Ainsi il est possible de déduire les relations de précédence entre les activités d'un processus.

Dans ce chapitre, nous proposons une méthodologie pour construire une recommandation à partir de la connaissance des processus d'usage. Nous nous plaçons dans un cadre où l'utilisateur déroule un processus métier conçu par un expert. Nous cherchons à savoir s'il est possible de définir un compagnon qui, à chaque terminaison d'une activité, est capable de proposer l'activité ou l'ensemble d'activités que doit suivre l'utilisateur pour atteindre une terminaison satisfaisante. Nous donnons le détail de cette méthodologie en commençant par décrire le système de traces que nous allons utiliser et le modèle de traces que nous allons utiliser dans la section 5.3.1. Puis, nous présentons comment nous réalisons la fouille de processus pour extraire les informations sur le processus métier réalisé, section 5.3.2, et nous donnons les principes pour réaliser une recommandation pendant l'exécution d'un processus métier, section 5.3.3.

Nous validons notre méthodologie et l'architecture logicielle qui en découle au travers d'un exemple simple, section 5.4. Il s'agit d'analyser les trajectoires d'étudiants d'IUT Informatique sur sept années, afin de construire notre base de connaissance, et de faire une proposition de recommandation sur la dernière année. Ce cas d'étude, nous permet de valider notre approche sur un exemple traité de bout en bout.

5.2 Principe de recommandation dans les processus

Notre recherche bibliographique nous a permis de prendre conscience de l'importance et de la difficulté d'organiser les traces brutes en traces modélisées. Nous avons étudié des méthodes qui permettent la supervision des activités d'un utilisateur dans le cadre de jeu ou de formation (Cordier et al., 2013; Ho et al., 2018; Toussaint and Luengo, 2015). Ces travaux se concentrent sur la prévention d'erreurs ou sur l'aide à la décision. Dans nos travaux, nous cherchons à définir une méthodologie qui, à partir des exécutions passées, permet de recommander à un utilisateur les activités à réaliser afin d'optimiser son processus. Nous cherchons à fournir un accompagnement pour améliorer les processus métier.

Nous observons deux situations. Dans la première, l'utilisateur se retrouve de plus en plus dans une situation où il est laissé en autonomie face au processus métier. Par exemple, beaucoup de services publics offrent des services sur des plate-formes en ligne, et l'usager doit assembler des morceaux de processus afin d'atteindre son but. La seconde situation correspond au cas des entreprises qui possèdent de nombreux processus métier (banque, assurance...) et qui utilisent le système d'information pour cadrer ces processus. Il en découle que les employés cherchent des alternatives pour réaliser contourner ce cadre, parfois rigide, en pratiquant leurs « propres processus métier ». Dans les deux cas, l'utilisateur se retrouve dans une situation où il est amené à prendre des décisions pour poursuivre son processus. Les décisions sont prises dans un contexte local, mais influent sur la performance globale du processus. Notre hypothèse de recommandation est que l'utilisateur doit recevoir une information sur la qua-

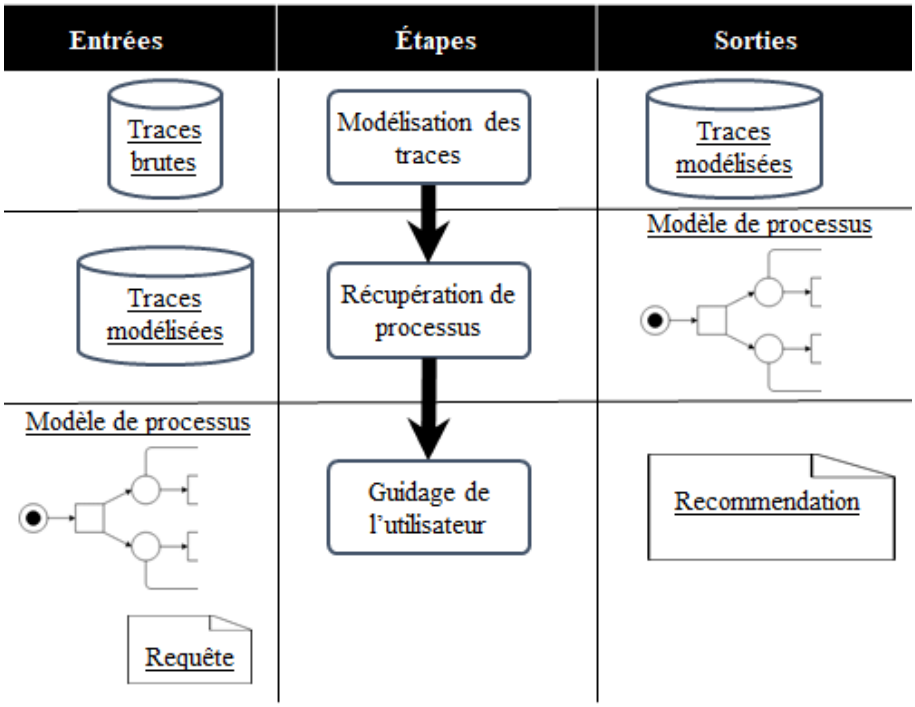

FIGURE 5.1 – Étapes de notre méthode de recommandation fondée sur les processus

lité de ses choix possibles (en termes de validité des processus) avec une mesure sur leurs performances.

Nous avons mis en place une méthodologie qui repose sur trois étapes, comme illustré par la figure 5.1. Il s'agit d'extraire des connaissances à partir des exécutions passées. Nous commençons par transformer les traces brutes en traces modélisées. Cette transformation est décrite dans la section 5.3.1. Puis, nous cherchons un modèle des processus d'usage sur lequel nous allons bâtir notre système de recommandation, rôle de le deuxième étape décrite dans la section 5.3.2. Cette recommandation se base sur la recherche de chemins optimaux dans un graphe et constitue la troisième étape de la méthode. Elle est décrite dans la section 5.3.3.

Ainsi, nous construisons un modèle qui représente le processus étudié et l'utilisons pour accompagner l'utilisateur en surveillant les activités effectuées. À chaque nouvelle activité, la méthode recommande une exécution à suivre pour atteindre l'objectif. Cependant, si l'utilisateur refuse notre solution, nous adaptons le processus proposé à ses choix tout en lui permettant d'atteindre son objectif, si cela est possible.

Deux phases sont nécessaires :

— récupérer le modèle de processus en appliquant la fouille de processus sur les traces (étape regroupant les 2 premières étapes ci-dessus), et,

— accompagner l'utilisateur en calculant une recommandation représentant l'exécution à effectuer pour atteindre l'objectif (3e étape ci-dessus).

5.3 Méthodologie

Nous décrivons en détail ci-après les trois étapes de la méthodologie proposée.

5.3.1 Extraction des traces modélisées

Dans cette étape, nous transformons les traces brutes en traces modélisées. Ce sont des données qui représentent le déroulement des exécutions du processus durant une période passée. Les traces sont structurées sous forme d'une table attribut-valeur ne véhiculant pas forcement la sémantique correspondante (la table 5.1 présente un exemple de telles traces brutes au format CSV avec 4 attributs). Il est nécessaire de les analyser pour identifier notamment l'activité effectuée dans l'exécution du processus.

La figure 5.2 illustre un exemple de transformation des traces brutes en traces modélisées. Les données des traces brutes qui sont identifiées en tant qu'activités sont ajoutées aux traces modélisées dans la colonne « Activité » et il en est de même pour les données représentant l'identifiant de séquence.

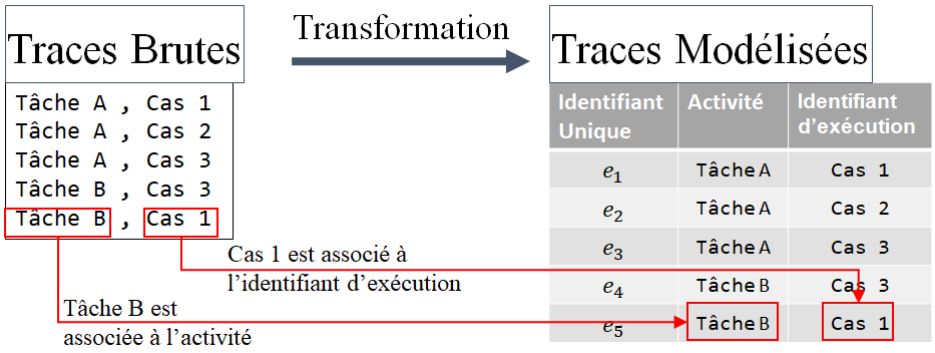

FIGURE 5.2 – Exemple de transformation des traces brutes en traces modélisées

Afin que cette étape se déroule correctement, les traces doivent contenir les informations permettant d'identifier dans une entrée donnée l'activité concernée et d'associer cette activité à une exécution.

Nous considérons l'ensemble des traces brutes noté T comme une table de base de données et nous utilisons l'algèbre relationnelle pour les définir (Roth et al., 1988). T est constitué des entrées $(t_1, ..., t_m)$ où chaque t_i est constituée de valeurs notées v_{ij}, $t_1 = (v_{11}, ..., v_{1n})$, avec i l'indice de l'entrée dans la table des traces et j l'indice de l'attribut concerné. Chaque v_{ij} est associée à un attribut noté r_j de domaine $\mathbb{D}_{r_j}$. L'ensemble de tous les attributs est noté R, $R = \{r_1, ..., r_n\}$ avec n le nombre d'attributs. Le domaine de définition des traces T est noté $\mathbb{D}_T$ et est équivalant à $\mathbb{D}_{r_1} \times ... \times \mathbb{D}_{r_n}$.

TABLE 5.1 – Exemple d'une table de traces brutes

Tâche A , Cas 1 , 2 , 1	Tâche D , Cas 2 , 1 , 4
Tâche A , Cas 2 , 2 , 1	Tâche E , Cas 5 , 1 , 1
Tâche A , Cas 3 , 2 , 1	Tâche C , Cas 4 , 2 , 2
Tâche B , Cas 3 , 2 , 2	Tâche D , Cas 1 , 1 , 4
Tâche B , Cas 1 , 2 , 2	Tâche C , Cas 3 , 2 , 3
Tâche C , Cas 1 , 2 , 3	Tâche D , Cas 3 , 1 , 4
Tâche C , Cas 2 , 2 , 2	Tâche B , Cas 4 , 2 , 3
Tâche A , Cas 4 , 2 , 1	Tâche F , Cas 5 , 1 , 2
Tâche B , Cas 2 , 2 , 3	Tâche D , Cas 4 , 1 , 4

TABLE 5.2 – Représentation des variables dans une table

T	r_1	...	r_n
t_1	v_{11}	...	v_{1n}
...		...	
t_m	v_{m1}	...	v_{mn}

Nous considérons aussi les fonctions de projection et de sélection de l'algèbre relationnelle. Soit :

1. $f_R(r_1, ..., r_k)$: la projection de T sur les attributs $r_1, ..., r_k | \{r_1, ..., r_k\} \subset R$ comme définie dans l'algèbre relationnelle,
2. $f_T(t_i)$: la sélection d'une entrée t_i de T comme définie dans l'algèbre relationnelle, avec $i \in \{1, ..., |T|\}$.

À partir des traces brutes, nous identifions ensuite les informations permettant, d'une part, de caractériser les exécutions et, d'autre part, les activités effectuées. Ces informations sont sous la forme d'un sous-ensemble des attributs présents dans les traces notés respectivement $R_{\mathcal{A}}$ et R_L :

— informations identifiant l'exécution $L = f_R(r_1, ...r_p), r_1, ...r_p \in R_L, R_L \subseteq R$.

— informations identifiant les activités $\mathcal{A} = f_R(r_1, ...r_q), r_1, ...r_q \in R_{\mathcal{A}}, R_{\mathcal{A}} \subseteq R$,

Le modèle d'événements ne nécessite que 2 informations : l'attribut activité et l'attribut identifiant d'exécution. Chaque entrée des traces suivant le modèle d'événements est appelée événement, noté $e = (a, l)$ avec $a \in \mathcal{A}$ l'activité de l'événement, et $l \in L$ l'identifiant de l'exécution. Par exemple, dans la table 5.1, le premier événement des traces modélisées (première ligne du tableau) est noté $e_1 = (\texttt{Tâche A}, \texttt{Cas 1})$. On note E la table ordonnée des traces d'événements, telle que $e_i \in E$. Le domaine de définition de E est le produit cartésien des domaines de définition des attributs de $R_{\mathcal{A}}$ et R_L. Ainsi, $E = (\mathcal{A}, L)$ avec $\mathcal{A}$ l'ensemble de toutes les activités observées dans les traces, L l'ensemble des identifiants d'exécution et toutes les entrées de E sont ordonnés temporellement. Il est possible de récupérer les informations d'une entrée e grâce aux deux fonctions suivantes :

$$f_a : E \to \mathcal{A} \qquad e \mapsto f_a(e) = a \qquad\qquad f_l : E \to L \qquad e \mapsto f_l(e) = l$$

Les traces modélisées obtenues représentent les exécutions du processus qui ont été enregistrées. Il est possible de déterminer une exécution σ en récupérant l'ensemble ordonné des événements e la constituant où chaque e possède le même identifiant d'exécution (ou de séquence) l. Nous notons cet ensemble ordonné par $\langle x, ..., y \rangle$ où x et y sont respectivement les événements de début et de fin de la séquence considérée. Ainsi, une exécution σ est donnée par :

$\sigma = \langle e_1, ..., e_n \rangle \Rightarrow \forall e, e' \in e_1, ..., e_n : f_l(e) = f_l(e')$

Par ailleurs, il n'est pas nécessaire que toutes les exécutions possibles soient présentes dans les traces. Néanmoins, plus il y en a et plus la méthode proposée aura de cas différents dans lesquels elle est capable de recommander une exécution terminant le processus.

L'étape de transformation ci-dessus permettant d'obtenir les traces modélisées E à partir des traces brutes T est résumé par l'algorithme l'algorithme 1.

Algorithme 1 Algorithme de transformation

Require: T, $R_{\mathcal{A}}$, R_L
Ensure: $F_T(T) = E$
  for all $t \in T$ **do**
    $e = (a, l)$
    $f_a(e) \leftarrow f_R(R_{\mathcal{A}}) \cap f_T(t)$
    $f_l(e) \leftarrow f_R(R_L) \cap f_T(t)$
    $E = E \cup e$
  end for
  return E

Exemple 1 : La table 5.3 montre les traces modélisées correspondant aux traces brutes de la table 5.1. La première colonne indique l'identifiant de l'entrée. Pour transformer une entrée des traces brutes t en entrée des traces d'événements e, nous associons l'activité à $R_{\mathcal{A}}$ et l'identifiant de séquences à R_L, sachant que $R_{\mathcal{A}}$ et R_L nous sont donnés. Puis, nous utilisons l'algorithme 1 afin de récupérer les traces modélisées représentées dans la table 5.3 dont la première ligne est $e_1 = (\texttt{Tâche A}, \texttt{Cas 1})$, avec :

- $a_1 = \texttt{Tâche A}$ est l'activité, soit $f_a(e_1) = \texttt{Tâche A}$,
- $l_1 = \texttt{Cas 1}$ est l'identifiant d'exécution, soit $f_l(e_1) = \texttt{Cas 1}$.

Pour simplifier, dans la suite du document, les activités de cet exemple seront renommées avec seulement la lettre qui leur est associée. Par exemple, l'activité Tâche A est renommée A.

TABLE 5.3 – Exemple de table de traces d'événements

Identifiant Unique	Activité	Identifiant d'exécution	Identifiant Unique	Activité	Identifiant d'exécution
e_1	Tâche A	Cas 1	e_{10}	Tâche D	Cas 2
e_2	Tâche A	Cas 2	e_{11}	Tâche E	Cas 5
e_3	Tâche A	Cas 3	e_{12}	Tâche C	Cas 4
e_4	Tâche B	Cas 3	e_{13}	Tâche D	Cas 1
e_5	Tâche B	Cas 1	e_{14}	Tâche C	Cas 3
e_6	Tâche C	Cas 1	e_{15}	Tâche D	Cas 3
e_7	Tâche C	Cas 2	e_{16}	Tâche B	Cas 4
e_8	Tâche A	Cas 4	e_{17}	Tâche F	Cas 5
e_9	Tâche B	Cas 2	e_{18}	Tâche D	Cas 4

Les deux ensembles $\mathcal{A}$ et L ne peuvent pas contenir de valeur qui n'a pas été observée dans T. Ce qui implique que ces ensembles sont comme suit :

- $\mathcal{A} = \{A, B, C, D, E, F\}$,
- $L = \{\texttt{Cas 1}, ..., \texttt{Cas 5}\}$

5.3.2 Modèle de processus métier réalisé

L'objectif de cette étape est de récupérer le modèle de processus représentatif de l'organisation des activités que l'on extrait à partir de l'analyse des traces. Pour cela, nous caractérisons le processus étudié et définissons nos attentes sur le modèle. Cela permet de déterminer les éléments structurels que la représentation doit satisfaire. Pour cela nous choisissons un algorithme de fouille de processus capable d'extraire de la connaissance sur le processus. Il existe de nombreux d'algorithmes permettant d'extraire le modèle de processus comme cela a été présenté dans l'état de l'art au paragraphe 3.5). Par exemple, dans le cas illustratif détaillé dans le paragraphe 5.4 du présent chapitre, l'algorithme utilisé est *Inductive Miner* (Leemans et al., 2013 a).

Parmi les modèles créés pour représenter les processus métier, nous avons choisi d'utiliser les *Workflow-Net*. Il est possible de calculer un chemin entre n'importe quel état atteignable, depuis l'état initial (constitué des places initiales) et l'état final, car nous savons que :

— Une activité est atteignable depuis une autre s'il existe au moins un enchaînement d'arcs qui va vers elle en provenant de l'autre.

— Une activité peut être franchie si l'ensemble des places en amont possèdent un jeton dans le marquage de l'état du modèle.

— Quel que soit le marquage s, si $p_o \in s$ alors $s = [o]$ où p_o est la place finale du modèle et $[o]$ le marquage ne contenant que la place finale).

.

Cela nous permet de déterminer si les séquences possèdent les caractéristiques suivantes :

— chaque exécution décrite dans les traces doit atteindre l'objectif du processus. En supposant que chaque exécution commence par un état initial et que l'objectif représente un état final, alors le modèle représentatif du processus, décrit dans les traces, est capable depuis n'importe quel état atteignable de l'état initial (lui-même inclus), de terminer le processus.

— sachant que chaque activité dans le modèle est issue des traces d'exécution et que chaque exécution commence par l'état initial pour atteindre l'état final, alors chaque activité peut être atteinte depuis l'état initial et chaque activité peut atteindre l'état final.

Le modèle de processus obtenu par l'utilisation de l'algorithme *Inductive Miner* (IM) (Leemans et al., 2013 a) suit les règles des *Block Structured Workflow-Nets* (décrits dans la section 3.5) et respectent les règles de *soundness*. La figure 5.3 représente un exemple d'un tel *Workflow-Net*.

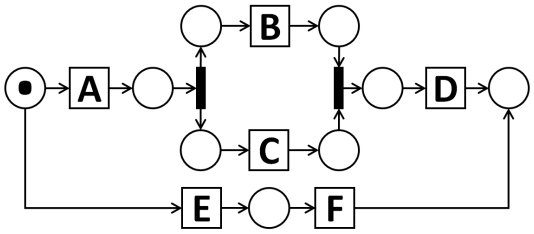

FIGURE 5.3 – Workflow-net correspondant aux traces modélisées

5.3.3 Proposition de recommandation pour l'utilisateur

Dans cette étape, chaque fin d'activité déclenche une recommandation. La progression d'un utilisateur est observée par l'apparition de traces décrivant une exécution non terminée. Cette étape a pour but de présenter à cet utilisateur une recommandation décrivant une exécution qui complète son exécution non terminée. On dit qu'une exécution en complète une autre lorsque franchir successivement les activités des deux exécutions depuis l'état initial du modèle, permet d'atteindre l'état final du modèle.

Afin d'identifier la progression de l'utilisateur, nous plaçons un déclencheur permettant d'exécuter l'étape de recommandation à chaque nouvelle trace : nouvelle entrée enregistrée dans les traces concernant une activité lors d'une exécution en cours.

À chaque recommandation, cette 3e étape fonctionne de la manière suivante (illusté par la figure 5.4) :

1. La progression de l'utilisateur dans le processus génère une trace ;

2. L'arrivée de la trace est observée par le déclencheur ;
3. Le déclencheur envoie une requête au système de recommandation portant sur l'exécution non terminée en cours ;
4. Le système de recommandation utilise le modèle pour calculer une exécution qui complète l'exécution en cours ;
5. L'exécution calculée par le système de recommandation est proposée à l'utilisateur.

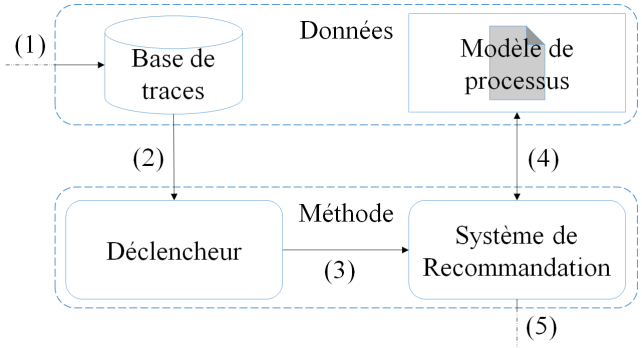

FIGURE 5.4 – Interactions entre les données et les composants durant l'étape de recommandation

Pour effectuer une recommandation, on doit calculer le dernier état connu de l'exécution non terminée σ_x. Pour cela, on calcule le marquage s atteint par l'exécution depuis le marquage initial $[i]$. Le marquage s de l'exécution σ_x est obtenu en franchissant dans l'ordre l'ensemble de ses transitions, tel que $(M, [i])[\sigma_x\rangle(M, s)$. Cependant, si $s = [o]$ alors l'étape se termine sans faire de recommandation, car l'exécution est terminée.

Pour calculer une exécution σ_r qui complète l'exécution non terminée σ_x, il existe plusieurs méthodes (voir section 4.5). Nous utilisons un algorithme de *Path Finding* qui nous permet de calculer un chemin depuis un état donné, soit le marquage s, jusqu'à l'état voulu qui est l'état final. Pour simplifier, nous partons de l'hypothèse que, dans un processus, le chemin possédant le moins d'activités est l'optimal à recommander.

Nous notons cette étape PF, permettant depuis une exécution non terminée σ_x de calculer le dernier état connu de σ_x représenté par le marquage s et, depuis ce marquage, de calculer une exécution σ_r, tel que $PF(\sigma_x) = \sigma_r$.

5.4 Validation de l'architecture proposée

Pour valider notre méthodologie, nous avons choisi de l'appliquer sur un cas d'étude illustratif de la progression académique des étudiants en Institut Universitaire de Technologie (IUT). Nous avons eu accès à sept années de décisions de jurys du Diplôme Universitaire

de Technologie (DUT) en Informatique. Ce cas d'étude permet de démontrer l'applicabilité de la méthode proposée sur un exemple de complexité maîtrisable dont nous avions le jeu de données complet (Champagnat et al., 2016; Leblay et al., 2018).

L'IUT délivre le DUT en deux ans. Celui-ci nécessite la validation de quatre semestres. À l'issue de chaque semestre un jury statue sur la réussite des étudiants au semestre en question. Dans le cadre de nos travaux, nous considérons le processus de validation des semestres suivis par les étudiants. Cependant, il est à noter qu'on pourrait adopter un point de vue plus fin en nous intéressant à la validation des Unités d'Enseignement (UE) voire des Éléments Constitutifs (EC), matières ou modules, des UE.

Ainsi, nous considérons les décisions des semestres comme des activités afin d'avoir une vision globale de la réussite (ou de l'échec) scolaire des étudiants. Le parcours académique de l'étudiant durant le DUT est alors vu comme le processus utilisateur suivi et la recommandation de processus devient alors le conseil et l'orientation de l'étudiant durant sa scolarité.

5.4.1 Description du cas d'étude

À l'IUT de La Rochelle, certains étudiants ont des difficultés à obtenir leur DUT et ce pour diverses raisons qui pourraient être liées à une mauvaise connaissance des règles d'obtention du diplôme. Notre premier objectif est d'utiliser et d'analyser les données des jurys des semestres pour identifier les chemins d'échec et les chemins de réussites, en particulier le chemin de réussite nominal.

L'IUT a normalisé le format pour les décisions des jurys. Chaque semestre (numéroté de 1 à 4) peut être :

- V pour la validation automatique si les résultats des étudiants répondent aux exigences académiques ;
- $C-$ ou $C+$ pour une validation avec compensation entre deux semestres successifs ($C-$ désigne la compensation avec le semestre précédent et $C+$ avec le semestre suivant) ;
- N pour une non-validation causée par des résultats académiques insuffisants ;
- J pour une validation accordée par le jury lorsque les résultats de l'étudiant ne permettent pas une validation automatique mais correspondent aux exigences académiques satisfaisantes pour le jury ;
- E si l'étudiant échoue et n'est pas autorisé à poursuivre, quelle qu'en soit la raison.

Le code ci-dessus est suivi du semestre concerné. Par exemple, $V//S1$ signifie que l'étudiant a eu des résultats suffisants et que le premier semestre a été automatiquement validé. Deux codes supplémentaires sont également possibles : REO et DEM qui désignent respectivement la réorientation et la démission.

Les données considérées dans le cas d'étude ont été fournies, après anonymisation, par le département Informatique de l'IUT de La Rochelle et concernent setp années de décisions du

jury (2007-2013). Chaque année, environ cent étudiants commencent le premier semestre en septembre. Pour chaque semestre, les études sont organisées en deux UE comprenant chacune plusieurs EC. Pour valider un semestre, un étudiant doit avoir une moyenne générale d'au moins 10/20 et une note moyenne d'au moins 8/20 dans chaque UE.

5.4.2 Format des données

Les traces représentent les résultats scolaires et la décision du jury par étudiant et par semestre. Ainsi, les traces ont les attributs suivants :

— `LearnerId` est l'identifiant de l'étudiant attribué après l'étape d'anonymisation. Cet identifiant généré automatiquement sert juste au suivi du parcours d'un même étudiant et aucun lien avec l'identité réel du l'étudiant n'est possible ;

— `Decision` est la décision du jury avec l'une des valeurs dans $\{V, J, N, C+, C-, A, E, DEM, REO\}$;

— `Semester` est le semestre concerné ;

— `Date` est la date de la prise de décision du jury ;

— `EvalAvr` est la note de moyenne générale ;

— `EvalUE1` et `EvalUE2` sont les notes moyennes dans les unités d'enseignements du semestre considéré [1] ;

— `EvalEC`xy sont les notes détaillées dans chaque EC avec x le numéro de l'UE et y l'index de l'EC dans l'UE en question.

Les traces brutes sont transformées en traces modélisées selon le modèle *EventLog* (*cf.* 5.3.2). Le modèle a besoin de *l'identifiant de l'exécution* et de *l'activité*. Nous instancions l'identifiant de l'exécution avec l'attribut `LearnerId` identifiant l'étudiant et l'activité avec le couple `Decision, Semester` correspondant à une décision donnée dans un semestre donné. Les attributs restants ne sont pas utiles à notre expérimentation et ne sont pas considérés.

5.4.3 Application de la méthode

Une fois les traces transformées, elles sont utilisées pour obtenir le modèle de processus en faisant la fouille de processus sur les données transformées avec l'algorithme *Inductive Miner*. La fouille de processus est effectuée en utilisant l'outil *ProM* [2]. Ce dernier prend en entrée des

1. Dans le Programme Pédagogique National (PPN) du DUT Informatique en vigueur sur la période considérée (2007 à 2013), tous les semestres étaient composés de 2 UE.

2. ProM est un environnement extensible regroupant divers outils dédiés à la fouille de processus (`http://www.promtools.org`)

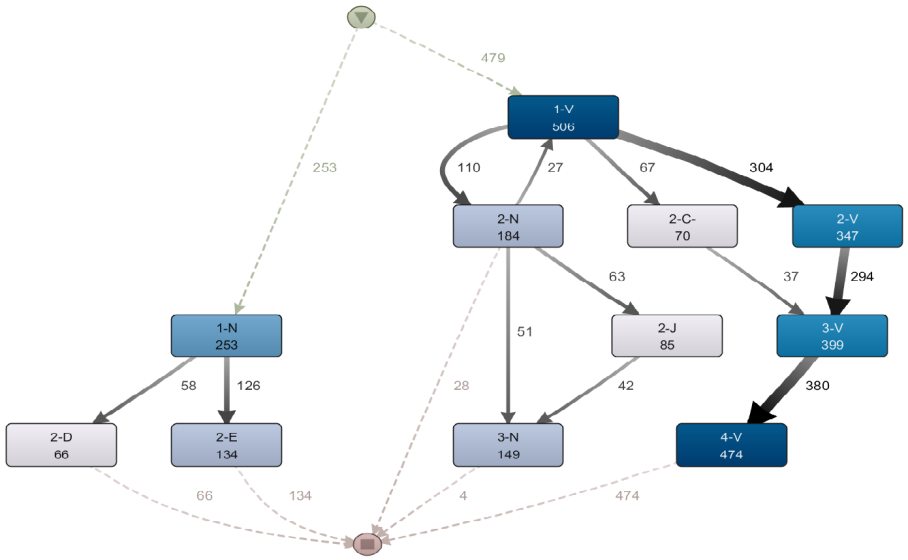

FIGURE 5.5 – Représentation schématique des exécutions des traces

données au format *XES* (présenté au paragraphe 2.4.1.2) et produit un modèle dans différents formats possibles, notamment au format *Workflow-Net* et ses dérivés.

Le modèle obtenu à la fin de cette étape représente l'organisation des décisions du jury dans les exécutions. La figure 5.5 donne un aperçu de ce résultat avec un point de vue général agglomérant les places et transitions les moins fréquentes pour améliorer sa lisibilité. La granularité la plus fine affiche un graphe complet, précis mais très complexe. Chaque transition est une décision du jury et chaque transition atteignable est une décision pouvant être prise au jury suivant. Ainsi, le modèle obtenu peut être vu comme les règles du jury de passage pour l'obtention du DUT.

La figure 5.5 montre les liens observés dans les exécutions entre les activités précédentes et suivantes dans les cas les plus fréquents. Chaque carré de la figure représente les décisions les plus fréquemment rencontrées et le nombre à l'intérieur détermine combien de fois cette activité a été observée dans les traces. Chaque arc indique l'activité la plus probable après celle considérée et elle est pondérée par le nombre de fois où cet enchaînement a été observé. Compte tenu du niveau de granularité choisi, toutes les exécutions n'y apparaissent pas.

Nous utilisons ce modèle pour calculer la recommandation comme expliqué dans le paragraphe 5.3.3. Dans cette expérimentation, nous nous sommes contenté de générer des recommandations pour chaque étudiant n'ayant terminé que le premier semestre. Chaque recommandation décrit une exécution terminant le processus.

5.4.4 Discussion

La figure 5.5 montre la fréquence des activités utilisées dans le processus étudié. Nous pouvons voir que les activités de validation sont les plus fréquentes. Cependant, les décisions

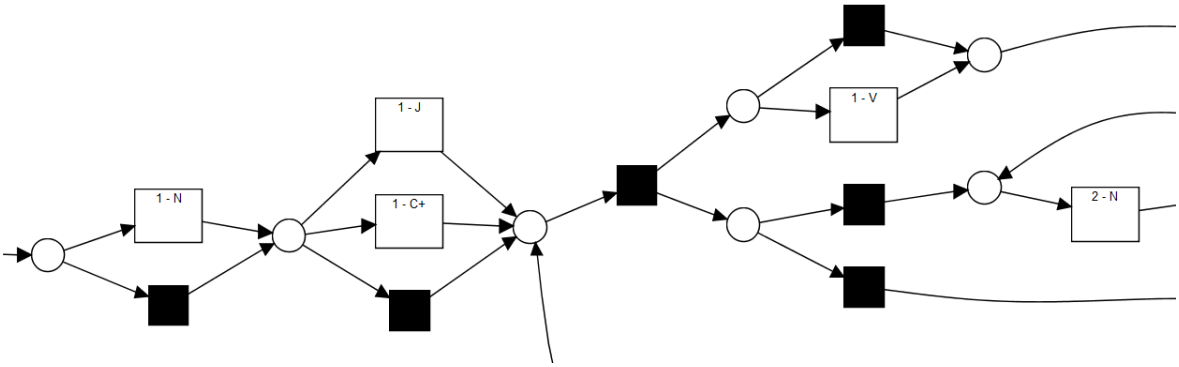

FIGURE 5.6 – Extrait du modèle représentant l'enchaînement des décisions du jury

de validation des semestres deux et trois sont moins fréquentes que les première et quatrième. Cela signifie qu'il existe d'autres moyens d'obtenir le diplôme et signifie également qu'une attention particulière devrait être apportée au parcours d'apprentissage des étudiants au cours de ces deux semestres. Nous pouvons également observer les chemins défaillants les plus fréquents et le fait que les étudiants échouent principalement après le premier ou le deuxième semestre. Un accompagnement devra être fourni par le département d'informatique pour aider les étudiants faibles du premier semestre.

La figure 5.6 présente un extrait du modèle *Workflow-Net* obtenu. On peut y voir les règles de précédence entre les différents semestres ainsi que les retours en arrière dus aux redoublements. Cependant, la complexité structurelle du modèle fait qu'il est difficilement utilisable par un humain et rend une automatisation de son utilisation nécessaire.

Dans l'expérimentation, nous identifions automatiquement les déviations potentielles en vérifiant si l'exécution diffère du chemin recommandé. Si c'est le cas, nous effectuons une recommandation. Cette dernière décrit les décisions du jury que l'apprenant doit obtenir afin d'avoir son diplôme et ce malgré l'écart du chemin recommandé avec le chemin courant. L'exécution décrite dans la recommandation devient ensuite le chemin recommandé.

Chaque recommandation a été vérifiée afin de savoir si l'exécution décrite arrive toujours à :

— respecter les règles de l'IUT, et,

— atteindre l'objectif.

Cependant, le modèle n'arrive pas à identifier l'absence totale de solutions. Il propose toujours une recommandation même si celle-ci est basée sur un cas atypique à partir du moment où il a été observé ne serait ce qu'une fois. Par exemple, quand nous cherchons une recommandation pour un étudiant ayant effectué $N2$ puis $N3$ et ayant déjà redoublé, notre méthode proposera un second redoublement car cela a déjà été observé alors que ce cas était une autorisation

exceptionnelle de redoubler une seconde fois pour une raison médicale. Ce système est donc limité par sa connaissance de l'organisation des règles ; il n'intègre que les règles observées et non pas les règles réelles. Par conséquent, il est nécessaire de considérer une étape supplémentaire à notre modèle qui interviendrait avant l'étape de recommandation et qui consisterait à enrichir le modèle en associant des connaissances supplémentaires. Cette étape permettrait de contextualiser le modèle obtenu pour le rendre plus précis. Cette étape sera expliquée en détails dans le paragraphe 6.3.3.

Il est à noter que dans cette expérimentation, nous nous sommes concentrés sur la mise en place de la méthode. Les données ont donc été transformées pour correspondre au format Event Log. Les traces ont été traitées par l'algorithme de fouille de processus *Inductive Miner*. Le *Workflow-Net* obtenu a été utilisé pour créer des recommandations. Il est possible par la suite de corréler les notes des étudiants avec le modèle pour donner des recommandations plus précises. La mise en place de la méthode dans le cadre des données de l'IUT de La Rochelle permet de montrer l'utilisabilité de la méthode sur un cas concret.

Nous sommes bien conscients que cette expérimentation, un peu naïve dans son analyse et éloignée des processus métiers d'entreprise, nous permet néanmoins de valider la faisabilité de notre méthodologie. Nous montrons qu'il est possible de construire une recommandation (choix de l'activité suivante à réaliser) à chaque fin d'activité en se basant sur un modèle de processus découvert à partir des traces d'exécutions passées.

5.5 Conclusion

Dans ce chapitre, nous avons proposé une méthodologie pour produire des recommandations lors de l'exécution de processus. Nous nous plaçons dans le cas où l'utilisateur dispose de beaucoup de liberté pour réaliser un processus, en fonction de son objectif, en composant des sous-processus ou séquences définies par un expert. Nous proposons donc une démarche et une architecture logicielle qui permettent, à la fin de chaque activité, de proposer l'activité la plus pertinente, vis-à-vis d'un objectif donné, à réaliser.

Notre méthodologie se compose de trois étapes. La première consiste à collecter les données des exécutions puis faire une transformation afin d'obtenir des traces modélisées au format *eXtensible Event Stream*. La deuxième étape consiste à extraire des connaissances sur les processus passés réalisés par les autres utilisateurs. Pour cela, nous utilisons les algorithmes de la fouille de processus. Cela nous permet de construire un modèle des processus métier. Ces deux étapes se font en amont de l'exécution. La troisième étape se fait en cours d'exécution et consiste à déterminer, à partir du modèle des processus et de la séquence réalisée jusque là par l'utilisateur, quelle activité doit être réalisée pour finir le processus en optimisant des critères préalablement définis.

Nous terminons ce chapitre par un exemple. Il s'agit de vérifier, à partir de données re-

cueillies sur les jurys d'IUT Informatique, que la méthodologie permet de produire des recommandations après avoir réalisé les transformations et extractions de connaissances nécessaires. Toutefois, cette expérimentation ne nous permet pas de nous confronter à des processus où l'utilisateur dispose de beaucoup de liberté. De plus, nous souhaitons également que l'architecture que nous proposons soit suffisamment générique pour permettre d'en substituer des éléments afin de les adapter à un autre contexte.

Dans le chapitre suivant, nous allons appliquer la méthodologie sur un cas d'étude pour lequel aucun processus métier n'a été prédéfini par un expert. Nous allons devoir formuler notre recommandation à partir des informations extraites des exécutions passées. Nous montrons ainsi comment l'on peut adapter notre méthodologie à ce contexte.

CHAPITRE 6 | Application au cas d'étude « Tamagotchi »

Sommaire

6.1 Introduction 84
6.2 Description du cas d'étude Tamagotchi 85
6.2.1 Motivations 85
6.2.2 Principe du jeu 86
6.2.3 Attributs du Tamagotchi 86
6.2.4 Critères de vie du Tamagotchi 87
6.2.5 Activités proposées au joueur 90
6.2.6 Format des traces 91
6.3 Adaptation de la méthode au cas d'étude Tamagotchi 93
6.3.1 Formalisation de l'approche proposée 93
6.3.2 Impact des activités sur l'évolution des critères 95
6.3.3 Formalisation du contexte dans le modèle de processus 96
6.3.4 Recommandation à partir du modèle de processus enrichi 98
6.4 Expérimentation 100
6.4.1 Paramètres et protocole expérimental 101
6.4.2 Présentation des résultats expérimentaux 102
6.5 Bilan 104
6.6 Conclusion 105

6.1 Introduction

Nous avons proposé une méthodologie pour la recommandation dans les processus métier dans le chapitre précédent. Nous avons terminé ce chapitre en mettant en place notre architecture sur un cas d'étude simple. Cette étude nous a permis de démontrer sa faisabilité. C'est-à-dire qu'à partir de traces, nous avons formulé une recommandation qui optimise un critère sur la réalisation du processus métier. Cette recommandation se base uniquement sur les aspects contrôle (l'ordre d'exécution des activités) et ne tient pas compte des données associées ni des données contextuelles.

Dans ce chapitre, nous traitons d'un exemple avec une complexité qui se rapproche des cas que nous souhaitons aborder, à savoir les processus « externalisés par les entreprises [1] », les jeux vidéo et l'apprentissage. La particularité de ces applications est que les processus ne correspondent plus à l'exécution d'une procédure rigide, mais offrent de la souplesse à l'utilisateur. Par conséquent, les processus produits risquent de varier et de faire apparaître des boucles qui traduisent les hésitations ou méconnaissances de l'utilisateur. De tels processus présentent un caractère non-structuré. Nous allons donc voir comment adapter notre méthodologie à ce type de processus à travers l'exemple d'un jeu, et présenter comment prendre en compte les données associées et/ou contextuelles.

Le cas d'étude que nous traitons porte sur un jeu inspiré du jeu Tamagotchi. Il s'agit de faire grandir un avatar et faire en sorte qu'il trouve un équilibre entre ses besoins primaires et ses interactions avec les autres. La section 6.2 détaille le fonctionnement de notre jeu, la dynamique associée aux attributs du Tamagotchi (santé, sociabilité et maturité) et le système de traces que nous mettons en place. Nous poursuivons par la présentation de l'expérience que nous allons développer. Le but de notre étude est de valider la robustesse de notre méthodologie dans le cas d'un système où l'utilisateur développe seul un/des processus métier. Nous nous plaçons donc dans une situation d'apprentissage où l'apprenant dispose d'un simulateur et met en œuvre des stratégies afin d'acquérir des connaissances. Dans la section 6.3, nous décrivons comment nous adaptons notre méthodologie à cette étude de cas. Nous ajoutons des annotations au modèle, à partir des informations contextuelles, pour l'enrichir. Nous présentons, ensuite, comment nous adaptons le système de recommandation, section 6.3.4.

Nous concluons ce chapitre par la section 6.4 qui présente l'expérimentation que nous avons conduite. Nous y présentons les critères qui permettent de mesurer la qualité de l'exécution. Puis, nous présentons notre protocole expérimental. Nous présentons le jeu aux participants en leur donnant comme objectif d'arriver à cinq victoires en cherchant à minimiser le temps de jeu. Nous avons réalisé notre expérimentation en deux temps avec deux groupes de testeurs différents. Le premier groupe ne bénéficie pas de recommandations. Il constitue ainsi

1. Par processus externalisés, nous entendons des processus définis par morceaux par des applications Web, où l'utilisateur doit construire son processus en composant des sous-processus.

notre groupe témoin et nous permet de collecter les traces qui vont servir à construire notre connaissance pour le second groupe. Ce second groupe se voit proposer des recommandations.

6.2 Description du cas d'étude Tamagotchi

Tamagotchi est un jeu interactif dans lequel le joueur doit prendre soin d'un animal virtuel appelé Tamagotchi. Le joueur doit utiliser les activités « soigner », « nourrir », « faire dormir », « faire jouer », *etc* pour assurer le bien-être du Tamagotchi le plus longtemps possible. Dans le cas contraire, il mourra. Chaque activité peut être faite à n'importe quel moment et se succéder dans n'importe quel ordre. Ainsi, le processus du jeu Tamagotchi est un processus non structuré. La description en détails de ces activités sera abordée dans la suite du chapitre.

6.2.1 Motivations

En plus de valider l'efficacité de notre méthode, l'étude de cas doit être illustrative et doit nous permettre de valoriser le deuxième objectif de la thèse concernant la recommandation à partir des informations sur un processus. Il est nécessaire d'avoir un exemple suffisamment complet permettant de récupérer et d'exploiter les traces, et les informations, sur les processus de l'application. Les traces sont générées par l'utilisateur et l'application pendant leurs interactions. L'étude de cas doit répondre aux caractéristiques suivantes :

— les processus composant l'application doit être non structurés ;

— l'application doit posséder assez d'interactions entre les différents acteurs afin d'avoir suffisamment de traces générées à exploiter. Chaque trace doit être récupérable dès sa génération afin qu'une requête puisse, par la suite, être envoyée à l'étape de recommandation ;

— l'application doit être de complexité maîtrisable afin de pouvoir illustrer tous les aspects de la méthodologie proposée. Les processus inhérents doivent pouvoir être décomposés en un nombre réduit mais suffisant d'activités. Un nombre trop important d'activités nécessite un trop grand temps de calcul et un nombre trop bas d'activités n'est pas significatif.

Par conséquent, nous devons fabriquer une application simple, connue, interactive et génératrice de traces interprétables. Cette application doit être :

— simple en termes de fonctionnalités implémentées et d'objectifs à atteindre ;

— connue dans le sens où les fonctionnalités de l'application sont ouvertes et implémentables par un développeur ;

— interactive pour permettre les échanges actions-réactions entre l'utilisateur et l'application ;

— capable de générer des traces qui relatent les interactions entre l'utilisateur et l'application et qui correspondent au processus suivi par l'utilisateur.

6.2.2 Principe du jeu

Nous considérons la vie du Tamagotchi à partir du moment où il sort de son œuf. À partir de ce moment, lorsque le Tamagotchi a besoin de quelque chose, un petit signal va apparaître pour avertir l'utilisateur d'intervenir. L'utilisateur a accès à l'état complet du Tamagotchi à travers divers indicateurs afin d'agir de façon appropriée : donner à manger, jouer, entretenir, éduquer, *etc*. La Figure 6.1 présente le modèle du processus nominal de l'application. Nous avons 7 activités qui seront enchaînées tout au long des exécutions du jeu. Elles sont : Manger, Jouer, Dormir, Soigner, Entretenir, Socialiser et Éduquer. La description en détail de ces situations sera abordée plus loin dans le chapitre.

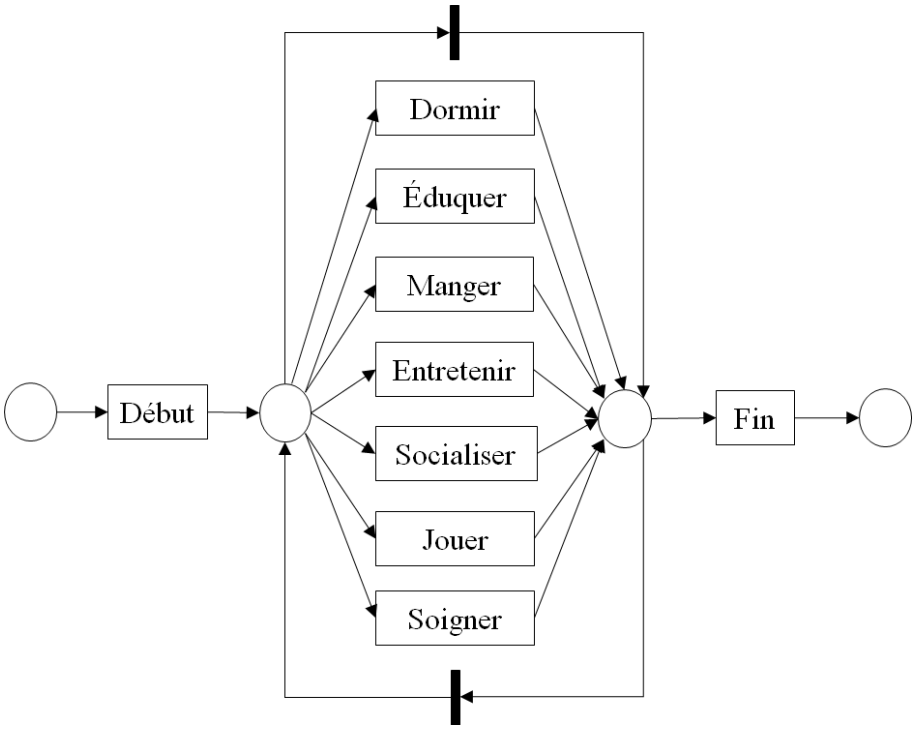

FIGURE 6.1 – Modèle réel du jeu Tamagotchi

6.2.3 Attributs du Tamagotchi

L'état du Tamagotchi est décrit par plusieurs attributs. Chaque attribut est une propriété décrivant une partie de son état global. Les différents attributs sont listés dans le tableau 6.1. Chaque attribut est une valeur numérique. La valeur de 1 indique un état optimal compte tenu de l'attribut considéré. Lorsque la valeur descend vers 0 l'intervention du joueur devient de

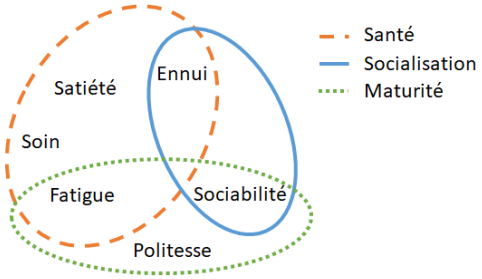

FIGURE 6.2 – Croisement des attributs et des critères de vie

plus en plus nécessaire et à 0 elle devient urgente car le Tamagochi risque la mort. Au delà de 1, l'attribut est en excès avec un risque d'overdose à 2. Par exemple, pour l'attribut Satiété, 1 indique que le Tamagotchi est rassasié, 0 qu'il est affamé et à 2 qu'il est gavé. Les deux extrêmes peuvent engendrer des conséquences graves pour le Tamagotchi comme la mort.

Ces attributs permettent de déterminer les critères de vie du Tamagotchi. Chaque attribut caractérise un ou plusieurs critères de vie.

6.2.4 Critères de vie du Tamagotchi

Il existe 3 critères de vie pour le Tamagotchi : la *santé*, la *socialisation* et la *maturité*. Chaque critère est défini par plusieurs attributs comme définis précédemment.

La Figure 6.2 schématise les liens entre les trois critères et les attributs les concernant. La santé se rapporte à quatre attributs : la satiété, le soin, la fatigue et l'ennui. La socialisation se rapporte à deux attributs : l'ennui et la sociabilité. La maturité est associée à la politesse, la fatigue et la sociabilité. Ainsi, certains attributs interviennent dans plusieurs critères : l'ennui (santé, socialisation), la fatigue (santé, maturité), la sociabilité (socialisation, maturité).

L'objectif global du jeu est que le Tamagotchi atteigne un niveau suffisant sur les critères (maturité, Socialisation et Santé) :

- La santé permet de savoir si le Tamagotchi est :
  - affamé, assez alimenté ou suralimenté ;
  - malade, sain ou sur-médicamenté ;
  - fatigué, reposé ou hypersomniaque ;
  - ennuyé, diverti ou hyperactif.
- La sociabilité permet de savoir si le Tamagotchi est :
  - seul, amical ou nuisible ;
  - ennuyé, diverti ou hyperactif.
- La maturité permet de savoir si le Tamagotchi est :
  - rustre, poli ou trop collant pour une vie en société ;

TABLE 6.1 – Les attributs du Tamagotchi

Attribut	Valeur	Description
Satiété	[0; 2]	La satiété du Tamagotchi admet une valeur réelle. Lorsque celle-ci est : • à 0, le tamagotchi a faim, seuil critique bas ; • entre 0 et 1 sa satiété augmente ; • à 1 sa satiété est optimale ; • entre 1 et 2 le tamagotchi commence à avoir trop mangé ; • à 2 il est gavé, seuil critique haut ;
Propreté	[0; 2]	La propreté du Tamagotchi admet une valeur réelle. Lorsque celle-ci est : • à 0, le tamagotchi est sale, seuil critique bas ; • entre 0 et 1 son hygiène augmente ; • à 1 le tamagotchi est propre ; • entre 1 et 2 le tamagotchi commence à détruire sa flore cutanée ; • à 2 la flore cutanée du tamagotchi n'est plus équilibré, seuil critique haut ;
Fatigue	[0; 2]	La fatigue du Tamagotchi admet une valeur réelle. Lorsque celle-ci est : • à 0, le tamagotchi est fatigué, seuil critique bas ; • entre 0 et 1 son repos augmente ; • à 1 le tamagotchi est reposé ; • entre 1 et 2 le tamagotchi commence à avoir trop dormi ; • à 2 il dort trop, seuil critique haut ;
Ennui	[0; 2]	L'ennui du Tamagotchi admet une valeur réelle. Lorsque celle-ci est : • à 0, le tamagotchi s'ennui, seuil critique bas ; • entre 0 et 1 son amusement augmente ; • à 1 le tamagotchi est diverti ; • entre 1 et 2 le tamagotchi commence à être en sur-activité ; • à 2 il est en sur-activité, seuil critique haut
Soin	[0; 2]	Le soin apporté au Tamagotchi admet une valeur réelle. Lorsque celle-ci est : • à 0, le tamagotchi est malade, seuil critique bas ; • entre 0 et 1 il commence à aller mieux; • à 1 le tamagotchi en bonne santé ; • entre 1 et 2 il commence à être sur-médicamenté ; • à 2 il est sur-médicamenté, seuil critique haut ;
Sociabilité	[0; 2]	La sociabilité du Tamagotchi admet une valeur réelle. Lorsque celle-ci est : • à 0, le tamagotchi seul, seuil critique bas ; • entre 0 et 1 il a quelques amis ; • à 1 le tamagotchi est bien entouré ; • entre 1 et 2 il commence à être trop présent dans la vie de ses amis ; • à 2 il est nuisible pour les autres et lui même, seuil critique haut ;
Politesse	[0; 2]	La politesse du Tamagotchi admet une valeur réelle. Lorsque celle-ci est : • à 0, le tamagotchi est impoli, seuil critique bas ; • entre 0 et 1 il commence à réfléchir quand il communique ; • à 1 le tamagotchi est poli; • entre 1 et 2 les gens commencent à trouver qu'il parle comme un livre ; • à 2 il n'est plus adapté à une communication informelle, seuil critique haut ;

— fatigué, reposé ou hypersomniaque ;

— seul, amical ou nuisible.

Il faut maintenir le Tamagotchi en vie le plus longtemps possible. Pour cela, il faut optimiser les trois critères calculés depuis l'état du Tamagotchi.

6.2.4.1 Santé

Si un Tamagotchi est en vie, nous pouvons observer sa santé. Tant que sa santé est bonne, il pourra vivre. Sinon, le Tamagotchi peut mourir ou l'utilisateur devra le soigner. Le critère santé est obtenu par :

$$\texttt{Santé} = \alpha_{sat}(\texttt{Satiété}) + \alpha_{soi}(\texttt{Soin}) + \alpha_{fat}(\texttt{Fatigue}) + \alpha_{enn}(\texttt{Ennui}) \quad (6.1)$$

Les fonctions α sont définies pour déterminer l'impact des attributs dans le calcul de la santé du Tamagotchi. Elle permet de représenter le manque ou l'excès des attributs.

Nous définissons aussi une règle concernant la santé du Tamagotchi pour fixer le seuil de *bonne santé*. En effet, l'objectif est de maximiser la valeur des facteurs tout en respectant des contraintes sur la santé. Pour cela, nous avons fixé des seuils déterminant l'état *santé insuffisante* à 0 et l'état *bonne santé* à $3,6$.

Une santé inférieure au seuil de *santé insuffisante* engendre la mort du Tamagotchi, car celui-ci aura soit des manques dans tous les attributs associés à sa santé ou au moins un manque conséquent dans un attribut. Comme le maximum atteignable sur la santé est 4, le seuil de l'état *bonne santé* a été placé à 90% du maximum pour mieux impliquer le joueur dans l'entretient de la santé du tamagotchi, ce qui implique une augmentation du nombre d'interactions. Cela permet de rendre le jeu suffisamment complexe pour générer assez de traces nécessaires à la création du modèle de processus.

6.2.4.2 Socialisation

Pour évaluer la socialisation du Tamagotchi, nous proposons d'utiliser les attributs sociabilité et ennui qui influencent directement l'aspect social. Nous introduisons la mesure suivante (où les fonctions β déterminent l'impact de chaque attribut sur le critère considéré) :

$$\texttt{Socialisation} = \beta_{soc}(\texttt{Sociabilité}) + \beta_{enn}(\texttt{Ennui}) \quad (6.2)$$

Le seuil maximum atteignable sur la socialisation est 2. Nous définissons une règle concernant la Socialisation du Tamagotchi pour fixer les seuils de *socialisation suffisante* et *socialisation insuffisante* à $1,8$ et à 0 respectivement, de la même manière que pour le critère santé.

6.2.4.3 Maturité

Le troisième critère vise à l'évaluation de la maturité du Tamagotchi combinant les attributs : sociabilité, fatigue et politesse.

Les règles de construction de cette fonction peuvent s'appuyer sur les cas suivants. Plus le Tamagotchi est fatigué, plus il fera des actions impolies. Par ailleurs, la sociabilité influence aussi la maturité du Tamagotchi car plus celui-ci est sociable plus il apprendra des autres Tamagotchis. En revanche, il peut faire des actions polies même s'il est fatigué, dans ce cas, il peut obtenir une bonne valeur de maturité. S'il n'est pas trop fatigué mais s'il fait des actions impolies, la maturité diminue rapidement. Nous proposons de calculer la maturité comme suit :

$$\texttt{Maturité} = \delta_{soc}(\texttt{Sociabilité}) + \delta_{fat}(\texttt{Fatigue}) + \delta_{pol}(\texttt{Politesse}) \quad (6.3)$$

Les fonctions δ indiquent l'impact de chaque attribut. Toujours de la même façon que pour les critères précédents, le seuil maximum atteignable sur la maturité est 3 et nous définissons les seuils de *maturité suffisante* et *maturité insuffisante* respectivement à $2,7$ et 0.

6.2.4.4 Déroulement

Au début du jeu, l'utilisateur découvre le Tamagotchi à l'état initial *Naître* qui correspond à la naissance du Tamagotchi. Dans cet état, la valeur de chaque attribut est placée à $0,2$. Cette valeur est suffisamment éloignée des seuils de *bonne santé*, *maturité suffisante* et *socialisation suffisante* pour garantir une durée et un nombre de traces suffisants par partie. De plus, les valeurs sont suffisamment éloignées des seuils de *santé insuffisante*, *maturité insuffisante* et *socialisation insuffisante* pour permettre aux joueurs de réagir avant la mort du Tamagotchi. Ainsi, le but du jeu pour le joueur est d'effectuer les actions permettant au Tamagotchi d'améliorer ses critères de vie et d'atteindre les niveaux optimaux pour les trois critères. Le jeu s'arrête soit lorsque cet objectif est atteint soit lorsque le Tamagotchi meurt pour cause de mauvais entretient.

6.2.5 Activités proposées au joueur

Le jeu comprend sept activités que l'utilisateur peut réaliser avec le Tamagotchi pour en prendre soin. Des actions inadéquates ou l'absence d'actions mèneront à sa mort (par exemple, s'il n'est pas bien soigné, ou s'il est gravement malade).

— Manger : donner à manger ou à boire au Tamagotchi ;

— Entretenir : nettoyer ou entretenir l'environnement où le Tamagotchi vit ;

— Jouer : permettre au Tamagotchi de jouer au jeu choisi pour distraire le Tamagotchi ;

— Soigner : si le Tamagotchi est malade, il a besoin d'être soigné ;

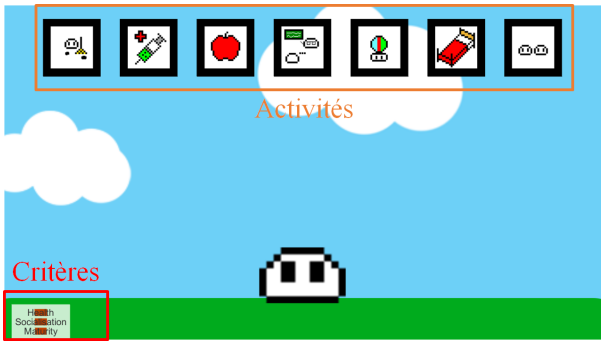

FIGURE 6.3 – Interface de l'application Tamagotchi

— Dormir : il faut faire dormir le Tamagotchi quand il est fatigué ;

— Socialiser : aider le Tamagotchi à se sociabiliser avec les autres ;

— Éduquer : apprendre au Tamagotchi à vivre en autonomie.

La figure 6.3 illustre la fenêtre principale de l'interface graphique du jeu. Elle donne accès aux activités possibles (icônes en haut de la fenêtre) et indique en permanence l'état des critères de vie (coin en bas à gauche).

Chaque activité a un effet positif ou non sur un ou plusieurs attributs. Le tableau 6.2 récapitule l'influence de chaque activité sur les attributs concernés. Une activité est utilisable pendant une certaine durée. Cette durée est calculée en unité de temps propre au jeu Tamagotchi. Chaque modification sur le tableau ($+1$, -1, ...) est calculée dans cette unité. Par exemple, la valeur de l'attribut satiété, lors de l'utilisation de l'activité manger pendant une unité de temps, augmentera de 6. Cela signifie que manger pendant une unité de temps gavera le Tamagotchi et influera négativement sur sa santé.

À chaque instant et même pendant l'utilisation d'une activité, chaque attribut diminue de 1 pour chaque unité de temps de jeu. Cela permet de représenter les effets du temps sur l'état du Tamagotchi et rendre le jeu plus propice à l'utilisation d'une stratégie.

6.2.6 Format des traces

À chaque activité effectuée (action de l'utilisateur), une trace est générée. Les traces de Tamagotchi sont sauvegardées au format *CSV*. Elles représentent les attributs avant l'utilisation de l'activité, l'activité effectuée et les attributs après l'activité. Ainsi, nous avons :

— `Id` : l'identifiant unique de l'utilisateur ;

— `gameNumber` : l'identifiant unique de la partie de Tamagotchi ;

— `c1a`, `c2a`, `c3a`, `c4a`, `c5a`, `c6a`; `c7a` : respectivement, les valeurs de la satiété, de la propreté, de la fatigue, de l'ennui, du soin, de la sociabilité et de la politesse avant l'utilisation de l'activité ;

TABLE 6.2 – Influence des activités sur les attributs

Activités	Attributs
Manger	Satiété +6 Ennui +1 Soin +2 Propreté -4
Entretenir	Propreté +6
Dormir	Satiété +0,5 Ennui +1 Fatigue +6 Soin +1 Politesse +1 Propreté +1
Jouer	Ennui +6 Fatigue -1 Soin -1
Soigner	Satiété -1 Ennui -1 Soin +6 Propreté +1
Socialiser	Ennui +6 Sociabilité +6
Éduquer	Politesse +6

— `activity` : l'activité effectuée ;

— `c1b`, `c2b`, `c3b`, `c4b`, `c5b`, `c6b`; `c7b` : respectivement, les valeurs de la satiété, de la propreté, de la fatigue, de l'ennui, du soin, de la sociabilité et de la politesse après l'utilisation de l'activité ;

— `duration` : la durée de l'activité en unité de temps propre au jeu Tamagotchi.

La table 6.3 donne un extrait du fichier *CSV* obtenu.

TABLE 6.3 – Exemple des traces de Tamagotchi

```
id,gameNumber,c1a,c2a,c3a,c4a,c5a,c6a,c7a,activity,...
-29,0,0.75,0.55,0.75,0.15,0.45,0.75,0.75,feed,     ...
-29,0,0.79,0.55,0.75,0.16,0.45,0.75,0.72,heal,     ...
```

```
...c1a,c2a,c3a,c4a,c5a,c6a,c7a,duration
...0.79,0.55,0.75,0.16,0.45,0.75,0.72,0.0063
...0.78,0.54,0.74,0.18,0.44,0.74,0.72,0.0045
```

6.3 Adaptation de la méthode au cas d'étude Tamagotchi

Dans cette section, nous appliquons la méthode proposée à l'étude de cas du Tamagotchi. Nous mettons à profit cet exemple applicatif complet pour présenter les détails des différentes étapes de la méthode. De plus, comme nous l'avons vu au chapitre précédent (*cf.* paragraphe 5.4.4), il est nécessaire de rajouter une étape avant l'étape de recommandation pour injecter de la connaissance métier. Cette étape d'enrichissement du modèle sera abordée dans le paragraphe 6.3.3.

La figure 6.4 présente la version modifiée de notre méthodologie. Les 2 premières étapes sont inchangées par rapport à la version initiale présentée dans la figure 5.1, nous les avons par conséquent grisées. La première étape récupère les traces de l'application et les transforme en traces modélisées afin d'y identifier les informations nécessaires à la fouille de processus comme l'activité et l'exécution (*cf.* paragraphe 5.3.1). La deuxième étape permet de construire le modèle de processus en utilisant les méthodes de fouille de processus sur les traces modélisées (*cf.* paragraphe 5.3.2). Cependant, le modèle de processus obtenu ne permet pas encore de représenter les processus non structurés car il n'intègre pas les critères contraignant l'objectif ni les moyens de modifier leurs valeurs. C'est le rôle de la nouvelle étape d'enrichissement du modèle (détaillée au paragraphe 6.3.3), insérée à la suite des 2 premières étapes. Elle permet de fournir en entrée à la dernière étape, étape de recommandation, un modèle de processus enrichi et donc plus précis.

Enfin, pour calculer les recommandations depuis le modèle de processus, nous utilisons une version modifiée de l'étape de recommandation décrite précédemment. Cette étape n'apparaît qu'en partie grisée car l'objectif reste le même alors que les données d'entrée ne sont plus les mêmes, ce qui implique que la façon d'obtenir l'exécution recommandée a été modifiée. Nous utilisons le modèle de processus enrichi pour proposer une recommandation en utilisant toutes les informations des processus non structurés. Cette dernière étape est décrite en détails au paragraphe 6.3.4.

Pour présenter notre méthode, il est nécessaire de décrire formellement les données manipulées.

6.3.1 Formalisation de l'approche proposée

Ci-après, nous travaillons sur la formalisation des informations du processus non structuré utilisés tels que les informations sur les critères de décision ou la façon de modifier leur valeur. Parmi les informations utilisées, nous ne nous attarderons pas sur les informations déjà formalisées au chapitre précédent comme l'identification des activités ou des séquences.

Les processus non structurés sont des processus possédant des critères contraignant l'ob-

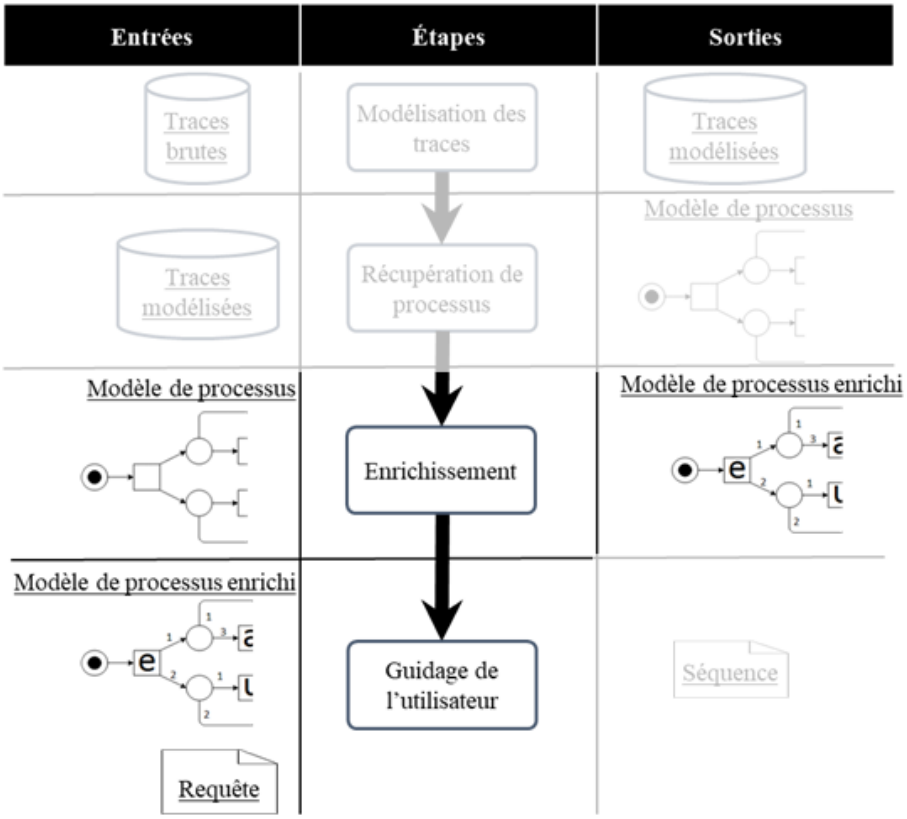

FIGURE 6.4 – Mise en évidence des modifications à l'approche proposée

jectif, dont la valeur change à chaque activité. Ces activités sont également très peu corrélées. Un exemple simple de processus non structuré se trouve dans le processus de composition de document avec un logiciel de traitement de texte. Diverses activités sont possible, telles que la rédaction, la mise en forme, le suivi des modifications, *etc*. Effectuer chaque activité une seule fois ne permet pas d'atteindre l'objectif ; il est souvent nécessaire d'effectuer plusieurs fois les activités et dans un ordre non déterminé pour finaliser un document. Ainsi, en observant les traces d'exécutions de ce processus, il est possible d'observer tout ou partie des activités dans un ordre quelconque et avec ou sans répétitions.

Nous utilisons les informations des processus non structurés pour accompagner l'utilisateur jusqu'à atteindre le résultat escompté. Elles deviennent les entrées de la méthode proposée. Cependant, toutes les informations disponibles ne sont pas utiles. Nous ne nous intéressons qu'aux informations sur :

— les critères permettant de déterminer si l'objectif est atteint. Il est nécessaire de contre les valeurs à l'état initial du processus et celle correspondantes à l'atteinte de l'objectif ;

— l'impact des activités sur le calcul des critères ci-dessus.

On considère un critère quelconque c comme un doublet avec une valeur à l'état initial du processus $c[init]$ (valeur au marquage initial du modèle de processus $[i]$) et une valeur qui

contraint l'objectif $c[fin]$ (valeur qu'il est nécessaire d'avoir lors de l'atteinte du marquage finale $[o]$). Ces deux valeurs sont respectivement appelées *valeur initiale* et *valeur finale* pour le reste du document.

Par exemple, dans le cas Tamagotchi, les critères sont des métriques propres au jeu représentant les critères de vis (Santé, Sociabilité et Maturité). Chacun est défini sur $\mathbb{R}$ et leurs valeurs initiales sont respectivement à $0,8$, $0,4$ et $0,6$ car les valeurs initiales de tous les attributs est $0,2$ (*cf.* paragraphe 6.2.4). Ces critères contraignent l'objectif car il est nécessaire d'atteindre une valeur respectivement de $3,6$, $1,8$ et $2,7$ pour terminer le processus.

Il est à noter que chaque critère peut être défini sur un domaine de définition différent à partir du moment où il permet de calculer une distance d entre les valeurs du critère considéré avec les propriétés suivante :

— $d(x, y) = d(y, x)$ (symétrie) ;

— $d(x, y) = 0 \Leftrightarrow x = y$ (séparation) ;

— $d(x, z) \leq d(x, y) + d(y, z)$ (inégalité triangulaire).

Le domaine de définition d'un critère $c_i \in C$ est noté $\mathbb{D}_{c_i}$ et le domaine de définition de l'ensemble des critères C est noté $\mathbb{D}_{c_1} \times ... \times \mathbb{D}_{c_m}$ avec m le nombre de critères ou plus simplement $\mathbb{D}_C$. Dans notre cas d'étude, par soucis de simplification, nous considérons que chaque critère est défini sur $\mathbb{R}$.

6.3.2 Impact des activités sur l'évolution des critères

La réalisation d'une activité peut avoir un impact positif ou négatif sur un ou plusieurs critères ce qui reflète le rapprochement ou l'éloignement de l'objectif global final. Ainsi, formellement, la valeur de chaque critère c_i est affectée, pour chaque activité a_j effectuée dans l'exécution, par une fonction g_{a_j}. Avec $c_i \in C$ un critère et $a_j \in A$ une activité. L'ensemble de toutes ces fonctions est noté $G = \{g_{a_1}, ..., g_{a_n}\}$ avec n le nombre d'activités. Par conséquent, après chaque activité a_j, les valeurs de tous les critères sont mis à jour :

$$\begin{aligned} &\forall a_j \in A, \forall c_i \in C \\ &g_{a_j} : \mathcal{R} \ \rightarrow \mathcal{R} \\ &\qquad c_i \ \mapsto g_{a_j}(c_i) \end{aligned}$$

Il est possible de déterminer la valeur des critères en utilisant, dans l'ordre, les fonctions associées aux activités de l'exécution depuis les valeurs initiales. Ainsi, pour une exécution $\sigma = \langle a_1, ..., a_n \rangle$, avec a_1 la première et a_n la dernière activités effectuées, les valeurs des critères peuvent être obtenues en composant les fonctions via l'opérateur $\circ$ tel que :

$$\forall c_i \in C, c_i[fin] = g_{a_n} \circ ... \circ g_{a_1}(c_i[init]) = g_{a_n}(g_{a_{n-1}}(...(g_{a_1}(c_i[init])...))$$

6.3.3 Formalisation du contexte dans le modèle de processus

Les informations sur les critères et les fonctions définissant l'impact des activités sur leur évolution sont utilisées pour enrichir le modèle de processus (un *Workflow-Net* dans notre cas). Quand nous parlons d'état du processus dans ce modèle, nous parlons du marquage. Maintenant, nous ajoutons à cet état un ensemble de valeurs pour chaque critère afin de déterminer si, à un état où le marquage final est atteint, les contraintes sur l'objectif sont satisfaites. Dans le but de représenter la connaissance de l'expert, nous y ajoutons une association entre les activités et les fonctions, nous y définissons les valeurs initiales et finales des critères depuis le marquage et nous y modifions la règle de franchissement.

Les critères sont considérés comme un ensemble de valeurs en plus du marquage. Étant donné que le marquage initial représente les éléments d'entrée du processus, les critères sont à leur valeur initiale. Ainsi, pour toute exécution σ_0 n'ayant aucune activité effectuée, tel que $\sigma_0 = \langle\emptyset\rangle$, chaque critère est à sa valeur initiale. Il est possible de représenter la valeur des critères de l'exécution σ_0 par le vecteur $\big[c_1[init], ..., c_m[init]\big]$. Chaque modification du marquage par le franchissement d'une transition (la réalisation d'une activité) modifie les valeurs de ce vecteur via la fonction de l'activité associée à la transition.

Puisque notre modèle se base sur les Workflow-Nets, il est alors impossible d'être a l'état initial autrement qu'en commençant une exécution (*cf.* le paragraphe 3.4.2). Ainsi, chaque critère c est à sa valeur initiale dans un *Workflow-Net* marqué $(\mathcal{M}, [i])$ qui commence alors l'exécution avec un vecteur de valeurs des critères $\big[c_1[init], ..., c_m[init]\big]$. De la même manière, on peut représenter les valeurs à atteindre par un vecteur d'arrivée $\big[c_1[fin], ..., c_m[fin]\big]$ regroupant les valeurs cibles pour chaque critère.

Nous intégrons les valeurs finales à atteindre pour terminer le processus dans le *Workflow-Net* en considérant que, quel que soit le marquage $(\mathcal{M}, s)$, si le franchissement d'une transition correspondant à une activité a, permet d'atteindre le marquage final $[o]$, tel que $(\mathcal{M}, s)[a\rangle(\mathcal{M}, [o])$, alors ce franchissement ne peut être effectué que si la fonction $g_a()$ associée à a permet d'atteindre les valeurs finales sur l'ensemble des critères : $g_a(\big[c_1, ..., c_m\big]) = \big[c_1[fin], ..., c_m[fin]\big]$. Ainsi, les valeurs des critères d'un *Workflow-Net* marqué $(\mathcal{M}, [o])$ seront forcément égales à $\big[c_1[fin], ..., c_m[fin]\big]$.

Nous ajoutons à la définition du paragraphe 3.4.1 le fait qu'à chaque franchissement la fonction associée à l'activité soit utilisée sur le vecteur des critères $\big[c_1, ..., c_m\big]$ avant franchissement. Ainsi, quel que soit un *workflow-Net* marqué $(\mathcal{M}, s)$, si une activité a est franchie, tel que $(\mathcal{M}, s)[a\rangle(\mathcal{M}, s')$, alors le nouveau vecteur des critères sera $g_a(\big[c_1, ..., c_m\big])$.

Une fois enrichi, le modèle de processus est toujours défini sur un ensemble de places P, un ensemble d'activités $\mathcal{A}$ et un ensemble d'arcs $\mathcal{F}$ comme un *Workflow-Net* classique. Le vecteur des valeurs des critères, tout comme le marquage, est considéré comme en tuple avec le modèle. Il est indissociable du marquage car il est défini depuis ce dernier et on note un

Occurrences	Exécutions
20	$\langle b, b, b, c, e, f \rangle$
19	$\langle a, b, b, b, b, c, e, f \rangle$
14	$\langle g, b, b, c, d, b, c, e, f \rangle$
11	$\langle g, a, b, b, b, c, d, b, c, e, f \rangle$
8	$\langle g, b, c, d, b, g, c, d, b, c, e, f \rangle$

TABLE 6.4 – Traces collectées à partir de processus non structurés

modèle de processus enrichi marqué par $(\mathcal{M}, s, [c_1, ..., c_m])$.

Exemple 1 : Soit le modèle de la figure 6.5 obtenu après les deux premières étapes de la méthode appliquées sur les traces du tableau 6.4.

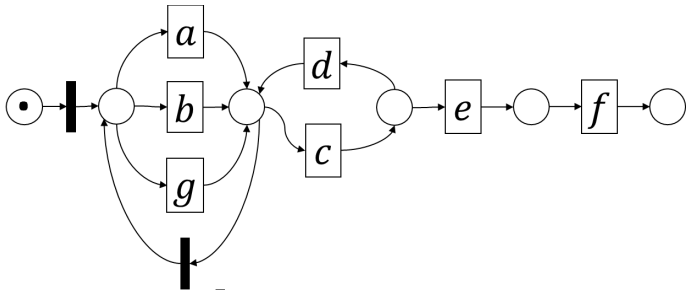

FIGURE 6.5 – Modèle de processus découvert depuis les traces du tableau 6.4

Pour enrichir le modèle, nous considérons deux critères $C = \{c_1, c_2\}$ avec :

- pour c_1, la valeur à atteindre est 0 et la valeur initiale est 3,
- pour c_2, la valeur à atteindre est 0 et la valeur initiale est 0.

Ainsi, le vecteur des valeurs initiales est $q_i = [3, 0]$ et celui des valeurs finales est $q_o = [0, 0]$. Puisque, nous avons sept activités $\mathcal{A} = \{a, b, c, d, e, f, g\}$ et deux critères $C = \{c_1, c_2\}$, nous avons donc besoin de 14 fonctions composantes agissant sur les critères (7 pour chaque critère). Pour un critère c_i et une activité x, la fonction associée est g_{x1}.

Si on définit pour chaque critère les fonctions composantes suivantes :

- Pour c_1 :

$$g_{a1}(c_1) = c_1 + 1$$
$$g_{b1}(c_1) = c_1 - 1$$
$$g_{c1}(c_1) = g_{d1}(c_1) = g_{e1}(c_1) = g_{f1}(c_1) = g_{g1}(c_1) = c_1$$

- Pour c_2 :

$$g_{a2}(c_2) = g_{b2}(c_2) = g_{c2}(c_2) = g_{e2}(c_2) = g_{f2}(c_2) = c_2$$
$$g_{d2}(c_2) = c_2 - 1$$
$$g_{g2}(c_2) = c_2 + 1$$

Et si on considère une fonction m-dimensionnelle pour chaque activité, avec $m = 2$, nous obtenons : $g_a \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} g_{a1}(c_1) \\ g_{a2}(c_2) \end{pmatrix} = \begin{pmatrix} c_1 + 1 \\ c_2 \end{pmatrix}$ $g_b \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} g_{b1}(c_1) \\ g_{b2}(c_2) \end{pmatrix} = \begin{pmatrix} c_1 - 1 \\ c_2 \end{pmatrix}$

$g_c \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} g_{c1}(c_1) \\ g_{c2}(c_2) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ $g_d \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} g_{d1}(c_1) \\ g_{d2}(c_2) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 - 1 \end{pmatrix}$

$g_e \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} g_{e1}(c_1) \\ g_{e2}(c_2) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ $g_f \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} g_{f1}(c_1) \\ g_{f2}(c_2) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$

$$g_g \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} g_{g1}(c_1) \\ g_{g2}(c_2) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 + 1 \end{pmatrix}$$

6.3.4 Recommandation à partir du modèle de processus enrichi

Pour faire la recommandation lors d'une exécution, nous appliquons la même méthode qu'au chapitre précédent. Pour cela, nous identifions le meilleur chemin à suivre dans le cas considéré. Puis, nous déterminons comment calculer le dernier état connu de l'exécution non terminée contenue dans la requête de recommandation. Enfin, nous décrivons le déroulement de l'approche. Pour calculer une exécution permettant d'atteindre l'objectif, nous utilisons l'algorithme de Dijkstra.

Cet algorithme a besoin d'un point de départ dans le modèle pour calculer un chemin vers le point d'arrivée. Pour cela, nous calculons le dernier état connu de l'exécution non terminée σ_x, tel que :

$$(\mathcal{M}, [i], q_i)[\sigma_x\rangle(\mathcal{M}, s, q) \quad | \quad q = a_n \circ ... \circ a_1(q_i), a_i \in \sigma_x$$

avec n le nombre d'activité dans σ_x, q le vecteur des valeurs des critères $\left[c_1, ..., c_m\right]$ et q_i le vecteur initial des valeurs des critères.

Pour simplifier, nous partons de l'hypothèse que dans un processus, le chemin possédant le moins d'activités est l'optimal à recommander.

Nous notons cette étape modifiée PF'. Elle permet depuis une exécution non terminée σ_x de calculer le dernier état connu de σ_x représenté par le marquage s et l'ensemble de valeurs des critères q puis, depuis cet état, de calculer une exécution σ_r, tel que $PF'(\sigma_x) = \sigma_r$.

Exemple 2 :

Cherchons, par exemple, à faire une recommandation depuis le modèle de l'exemple 1 pour une requête contenant la séquence $\sigma = \langle g, b, b, b\rangle$. Pour calculer l'ensemble des valeurs des critères q de l'exécution, nous utilisons successivement les fonctions associées aux activités de l'exécution, telles que :

$$q = g_b \circ g_b \circ g_b \circ g_g(q_i) = g_b \circ g_b \circ g_b \circ g_g \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

$$q = g_b \circ g_b \circ g_b \begin{pmatrix} g_{g1}(3) \\ g_{g2}(0) \end{pmatrix} = g_b \circ g_b \circ g_b \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

$$q = g_b \circ g_b \begin{pmatrix} g_{b1}(3) \\ g_{b2}(1) \end{pmatrix} = g_b \circ g_b \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$q = g_b \begin{pmatrix} g_{b1}(2) \\ g_{b2}(1) \end{pmatrix} = g_b \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$q = \begin{pmatrix} g_{b1}(1) \\ g_{b2}(1) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

On calcule aussi le marquage s de l'exécution, tel que $(\mathcal{M}, [i])[\sigma\rangle(\mathcal{M}, s)$

Par la suite, on cherche une séquence permettant d'atteindre le marquage $[o]$ et les valeurs à atteindre des critères représentés par un vecteur final $q_o = [0, 0]$ sur le modèle $\mathcal{M}$ représenté sur la figure 6.5. Ainsi nous utilisons l'algorithme *path-finding* évoqué précédemment :

Iteration 1 Au début, la liste des chemins candidats ne contient que $\langle g, b, b, b\rangle$, soit la séquence de la requête. Les transitions franchissables sont a, b, c et g et sont toutes franchissables depuis l'exécution $\langle g, b, b, b\rangle$. Les exécutions résultantes possibles sont $\langle g, b, b, b, a\rangle$, $\langle g, b, b, b, b\rangle$, $\langle g, b, b, b, c\rangle$ et $\langle g, b, b, b, g\rangle$. Toutes ont la même taille et nous choisissons aléatoirement $\langle g, b, b, b, c\rangle$, qui est ensuite ajouté à la liste des chemins candidats. $[o]$ n'est pas atteint et l'ensemble des valeurs des critères n'est pas le même que l'ensemble des valeurs du vecteur à atteindre. Donc l'algorithme continue ses itérations.

Iteration 2 La liste des chemins candidats contient $\langle g, b, b, b\rangle$ et $\langle g, b, b, b, c\rangle$. Les transitions franchissables sont a, b, g qui sont franchissables depuis $\langle g, b, b, b\rangle$, et d et e qui sont franchissables depuis $\langle g, b, b, b, c\rangle$. Les exécutions issues du franchissement d'une transition depuis $\langle g, b, b, b, c\rangle$ possèdent plus d'activités que celles issues de $\langle g, b, b, b\rangle$. Ainsi, parmi ces dernières, $\langle g, b, b, b, b\rangle$ sera choisie aléatoirement car elles ont la même taille. $[o]$ n'est toujours pas atteint. $\langle g, b, b, b, b\rangle$ est donc ajoutée à la liste des chemins candidats. L'algorithme continue ses itérations.

Dans l'exemple considéré, le vecteur des critères cible q_o est atteint après la 59^{e} itération. La séquence recommandée obtenu est $\sigma_r = \langle c, d, c, e, f\rangle$. Cette séquence est proposée à l'utilisateur comme suite de l'exécution car elle atteint $[o]$. Ainsi :

$$(M, [i])[g, b, b, b\rangle(\mathcal{M}, s)[c, d, c, e, f\rangle(M, [o])$$

et les valeurs des critères atteignent les valeurs de q_o, tel que :

$$q = \underbrace{g_f \circ g_e \circ g_c \circ g_d \circ g_c}_{g_{\sigma_r}} \circ \underbrace{g_b \circ g_b \circ g_b \circ g_g}_{g_\sigma} \begin{pmatrix} 3 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

À la fin de l'étape de recommandation, l'utilisateur a le choix de suivre ou non la séquence σ_r proposée. Cependant, quelle que soit l'activité suivante choisie, l'étape de recommandation n'est refaite que si l'utilisateur dévie de la séquence proposée.

6.4 Expérimentation

Nous utilisons le cas Tamagotchi pour réaliser nos expérimentations. Nous observons comment les utilisateurs terminent les processus, toutes les activités d'une exécution donnée d'un utilisateurs sont enregistrées dans les traces. Et nous définissons des métriques pour comparer l'efficacité des exécutions des différents utilisateurs, soit :

— durée : durée prise par un utilisateur pour gagner une partie, la plus courte sera le mieux. La durée est calculée en nombre de secondes ;

— échec : nombre de parties perdues entre deux parties successives remportées ;

— bruit : pourcentage des activités n'ayant pas fait progresser l'utilisateur pour atteindre l'objectif, soit le rapport entre le nombre d'activités bruitées et le nombre d'activités totales.

Une activité est considérée comme bruitée si la distance entre les valeurs des critères à atteindre et les valeurs des critères avant l'utilisation de l'activité est plus petite que la distance entre les valeurs des critères à atteindre et les valeurs des critères après l'activité. Par souci de simplicité, nous utilisons une distance Euclidienne entre les valeurs des critères $x \in q$ et les valeurs des critères à atteindre $y \in q_o$, soit :

$$d(q, q_o) = \sqrt{\sum_{1}^{m} (x - y)^2}$$

Cependant, pour l'utiliser, il est nécessaire de normaliser les valeurs. Ainsi, après normalisation, chaque valeur est contenue entre 0 et 1 car la normalisation divise chaque valeur par la valeur maximale atteignable sur le critère. Cette valeur maximale est considérée comme donnée à notre méthode dans les informations sur les critères.

Dans la suite de cette section, nous décrivons la mise en œuvre de la méthode proposée pour le cas Tamagotchi et présentons les résultats obtenus. Le paragraphe 6.4.1, décrit l'expérimentation et son déroulement. Le paragraphe 6.4.2 est dédié à la présentation et à la discussion des résultats.

6.4.1 Paramètres et protocole expérimental

Nous avons mis en œuvre l'architecture illustrée dans la figure 6.6 correspondant à la méthode proposée. L'ensemble des composants du cadre *Application Tamagotchi* a été développé dans le cadre des travaux précédents au laboratoire L3i ((Ho Hoang and Ho, 2015)) tandis que les éléments du cadre *Implémentation de la méthode* ont été développés dans le cadre des travaux de la présente thèse. L'étape de gestion des traces utilise l'ensemble des données de la base des traces de l'application. Puis, l'étape de fouille de processus l'utilise pour la récupération de modèle de processus. L'étape d'enrichissement transforme le modèle en modèle de processus enrichi. Enfin, l'étape de recommandation est déclenchée à l'apparition d'une nouvelle trace dans la base de traces de l'application. À chaque nouvelle trace, l'étape de recommandation calcule et envoie une recommandation vers l'application où elle sera proposée à l'utilisateur.

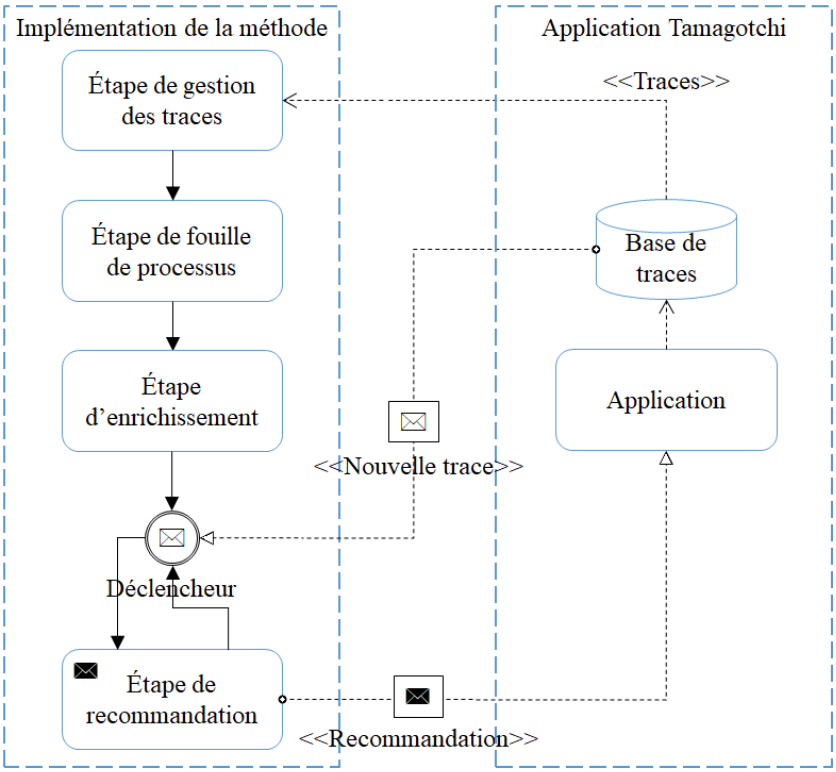

FIGURE 6.6 – Mise en œuvre de la méthode proposée sur le cas Tamagotchi

Les traces ont été récupérées de deux groupes de joueurs. Le premier groupe, constitué de 17 joueurs, est un groupe de joueurs non accompagnés durant leurs utilisations de l'application. Le second groupe, constitué de 8 joueurs, a quand à lui été accompagné par la méthode définie durant le jeu. Chaque joueur a effectué des parties de Tamagotchi jusqu'à obtenir 5 victoires, afin d'avoir suffisamment de traces pour les traitements. Le but est que les joueurs découvrent par eux-mêmes le processus sous-jacent au jeu Tamagotchi. Il a été fourni à chaque

joueur une notice expliquant comment utiliser l'application. Il a été notamment indiqué qu'il fallait réaliser 5 victoires en tentant de minimiser le temps de jeu. Le modèle de processus construit lors de la mise en place de notre méthode est issu des traces d'exécutions des victoires du premier groupe d'utilisateurs.

Pour identifier si une activité a fait progresser l'utilisateur, nous calculons la distance entre les valeurs des critères de vie du Tamagotchi à atteindre pour gagner la partie et leurs valeurs actuelle.

Pour calculer la distance d à l'objectif, nous avons utilisé une distance Euclidienne entre les vecteurs $(x_1, ..., x_m)$ et $(y_1, ..., y_m)$, soit :

$$d((x_1, ..., x_m), (y_1, ..., y_m)) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

avec m le nombre de critères.

6.4.2 Présentation des résultats expérimentaux

Afin de comparer les groupes, nous avons agrégé les valeurs des métriques pour obtenir :

— La durée moyenne et l'écart-type des exécutions pour les deux groupes ;

— Le nombre moyen et l'écart-type des défaites avant une réussite pour les deux groupes ;

— La moyenne et l'écart-type du pourcentage de bruits dans les exécutions pour les deux groupes.

Nous avons filtré les données afin d'enlever les valeurs aberrantes correspondant aux joueurs qui n'ont pas respecté les règles de l'expérimentation et ceux qui refusaient de suivre les recommandations.

Les résultats obtenus montrent qu'en utilisant notre accompagnement, les joueurs ont diminué la durée de leur exécution de $62,4\%$, leurs échecs de $66,7\%$ et le bruit qu'ils produisent de $26,1\%$ par rapport aux parties sans accompagnement.

La figure 6.7 présente la durée moyenne et l'écart-type des exécutions des victoires. La moyenne est présentée par une barre d'histogramme tandis que l'écart-type est présenté par un intervalle à son sommet.

La durée moyenne des exécutions diminue tout au long des victoires des joueurs qu'ils soient accompagnés ou non passant de $5,7$ unités de temps pour la première victoire du premier groupe à $2,1$ pour la cinquième du même groupe. On estime que cela est dû à la découverte du processus par l'utilisateur. Cependant, il est possible de voir que la moyenne de la durée d'une victoire avec accompagnement est significativement inférieure à la moyenne de la durée de la victoire qui lui est associée sans accompagnement avec une diminution de la moyenne de $62,4\%$ et une diminution de l'écart-type de $90,3\%$. Cela s'explique par le fait que les

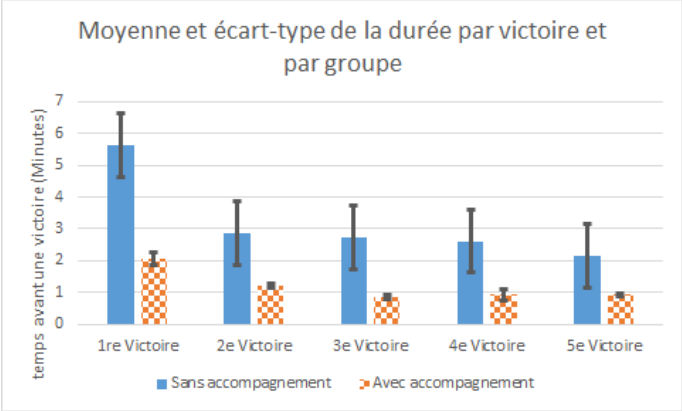

FIGURE 6.7 – Moyenne et écart-type de la durée par victoire et par groupe

recommandations proposées cherchent à optimiser la durée. De plus, comme les utilisateurs suivent majoritairement les recommandations qui leurs sont proposées l'écart-type est réduit.

La figure 6.8 présente le nombre moyen de défaite avant une victoire et l'écart-type de cette moyenne. Il est possible de remarquer que les défaites n'apparaissent qu'avant la première victoire des joueurs qu'ils soient accompagnés ou non. La raison à cela est qu'après la première victoire d'un joueur, celui-ci connaît le processus et cherchera seulement à l'améliorer par la suite. Cela explique aussi la diminution de la durée dans la figure 6.7. Il est aussi possible de voir que, même si la moyenne et l'écart type entre les deux groupes s'entrecroisent, le nombre moyen de défaites avant une victoire est le plus faible quand les joueurs sont accompagnés, avec une diminution globale de $66,7\%$. Cela est dû au fait que la méthode leur présente une stratégie pour gagner qu'ils assimilent en l'exécutant.

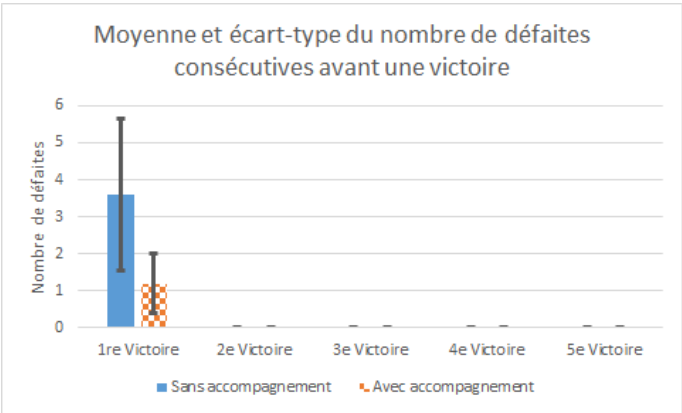

FIGURE 6.8 – Moyenne et écart-type du nombre de défaites consécutive avant une victoire

La figure 6.9 présente la quantité moyenne de bruit pour une victoire et son écart-type. Il est possible de remarquer que, comme la durée, le bruit diminue tout au long des victoires passant de 55% à la première victoire à 39% à la cinquième victoire pour le premier groupe. Entre les

groupes, le bruit a été significativement diminué passant de 55% pour le premier groupe à 47% pour le second groupe durant les premières victoires, avec une diminution globale de $26,1\%$. Cela représente le fait que les joueurs suivent la recommandation et sont moins susceptibles de faire des erreurs.

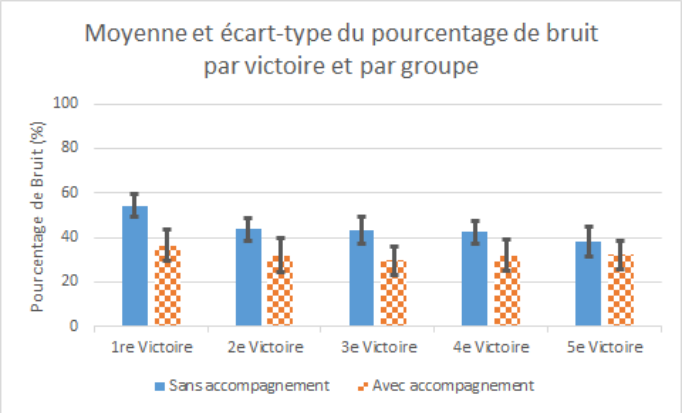

FIGURE 6.9 – Moyenne et écart-type du pourcentage de bruit par victoire et par groupe

Cependant, il est possible de voir qu'il reste une certaine quantité de bruit dans les traces, et ce pour deux raisons :

— un chemin constitué des activités les plus pertinentes à chaque instant ne guidera pas obligatoirement à l'objectif et donc afin d'optimiser le temps, il est parfois nécessaire d'ajouter du bruit. Par exemple, un chemin dont chaque activité est la plus optimisée à chaque étape, n'est pas forcément le chemin le plus optimisé du point de vue global. Pour rejoindre le chemin le plus optimisé, il est parfois nécessaire de faire des activités qui sont considérées, par notre système, comme du bruit. Ainsi, même notre assistance ajoutera du bruit si nécessaire mais cela garantira une quantité minimum de bruit. Il est donc actuellement impossible que le joueur arrive à faire moins de bruit que nos recommandations ;

— les activités peuvent avoir un effet négatif, si elles sont utilisées en excès. Par exemple, les activités des recommandations sont calculées pour une durée précise, alors, si l'utilisateur dépasse le temps recommandé, il est possible de produire du bruit.

6.5 Bilan

Dans cette expérimentation, nous cherchions à démontrer l'efficacité de notre méthode en comparant les exécutions des utilisateurs accompagnés avec les exécutions d'utilisateurs non accompagnés. Les résultats montrent que, sur les métriques définies, les exécutions des utilisateurs accompagnés sont meilleures : plus rapides, moins bruitées et ont plus tendance à atteindre l'objectif.

Toutefois, il est à noter que notre expérimentation se limite à une application simple ayant un nombre limité d'activités et que notre population de test est aussi limitée. Il serait alors intéressant d'étudier le comportement de notre approche dans un contexte où le nombre d'activités et d'utilisateurs est élevé :

— expérimenter notre approche dans le cadre d'une application plus complexe comme dans le cas d'un système d'information. Cela implique le passage à l'échelle de notre approche ;

— expérimenter d'autres heuristiques dans un cas où les ressources, en terme de puissance de calcul, sont limitées.

6.6 Conclusion

Dans ce chapitre, nous validons l'intérêt de notre méthodologie pour un environnement dans lequel l'utilisateur, l'apprenant, développe son propre parcours d'apprentissage et construit ainsi son processus métier. Nous montrons comment il est possible d'adapter notre architecture logicielle pour prendre en compte le contexte d'exécution. Pour cela nous avons construit un cas d'étude basé sur une simulation d'un micro-monde inspiré du jeu Tamagotchi.

Nous finissons ce chapitre par la présentation d'une expérimentation que nous avons menée à partir du jeu Tamagotchi. Après avoir présenté nos critères d'évaluation, nous avons décrit notre protocole expérimental. Dans celui-ci, nous demandons à deux groupes d'utilisateurs de réaliser des parties avec pour objectif d'obtenir un état de satisfaction haut pour le Tamagotchi, le plus rapidement possible. Le premier groupe nous sert de groupe témoin et nous permet de développer de la connaissance sur les processus d'usage de notre jeu. Le second bénéficie de notre outil de recommandation. Nous constatons une réduction de la durée de jeu et moins de dispersion de la part des joueurs. Ainsi ils passent moins de temps à explorer l'environnement et vont directement à l'essentiel.

Nous avons ainsi montré qu'il était possible de mettre en place une architecture pour la recommandation de processus qui construit sa connaissance à partir de l'analyse des traces en utilisant les algorithmes de la fouille de processus.

CHAPITRE 7 | Conclusion

Sommaire

7.1 Synthèse des travaux 108
7.2 Bilan 110
7.3 Perspectives 111

7.1 Synthèse des travaux

Au cours de cette thèse nous avons abordé le problème de la recommandation pendant l'exécution d'un processus métier. Nous nous sommes questionnés sur l'apport de la fouille de processus pour faire une recommandation dans un tel contexte. Nous avons plus particulièrement étudié la mise en place d'une architecture logicielle pour réaliser une telle recommandation. Nous nous sommes donc concentrés sur les aspects informatique. Cela nous a amené à nous interroger sur la possibilité de construire une suite de traitements dans la perspective de construire un compagnon pour la conduite de processus métier.

Nous nous sommes placés dans le cadre de processus métier « *faiblement structurés* ». Les applications visées par ce type de processus concernent les systèmes d'informations où l'on externalise des fonctionnalités aux utilisateurs (site Web marchand, saisie d'informations pour une administration, formation).

Dans le chapitre 2 nous avons réalisé un état de l'art sur les systèmes à base de traces. En effet un système à base de trace défini la façon d'observer et de collecter des données issues d'interactions entre l'utilisateur et le système. Avec le développement du *big data* et de l'*intelligence artificielle* les traces numériques ont attisé les convoitises et ont permis le développement de techniques d'observation et d'extraction de l'information. Nous avons présenté les grands principes des systèmes à base de traces et nous avons proposé une étude comparative de ces systèmes appliqués au contexte de l'enseignement. Cela nous a permis d'identifier nos besoins pour la fouille de processus. En effet la fouille de processus se base sur des événements, c'est-à-dire qu'une donnée tracée doit correspondre à un événement et doit être horodatée.

Nous avons poursuivi notre état de l'art, dans le chapitre 3, par une analyse du domaine de la fouille de processus. À partir des traces recueillies dans un système d'information, la fouille de processus vise à déterminer et analyser les processus métiers qui sont mis en œuvre. Nous avons présenté les grands principes de la fouille de processus. Il s'agit d'étudier les relations de causalité entre les activités enregistrées pour un processus. À partir de ces relations, des algorithmes ont été développés afin d'extraire des connaissances sur les processus d'usage. Nous avons présenté, ensuite les mesures utilisées pour déterminer la qualité d'un modèle miné. Cette étude nous a permis de constater que les algorithmes de la fouille de processus n'étaient pas construit pour traiter de processus faiblement structurés. Toutefois nous avons pu observer que l'algorithme *Inductive Miner* était suffisamment performant pour être utilisé sur les cas d'études que nous avons étudié dans cette thèse.

Nous finissons notre état de l'art par le chapitre 4 qui aborde le domaine de la recommandation. Nous avons présenté les systèmes de recommandation et leurs composants. Nous avons proposé une classification des systèmes de recommandation basée sur les actions des utilisateurs possédant des caractéristiques proches ou basée sur le contenu. Puis nous avons décrit les

mesures de similarité couramment utilisées. Cette étude nous a permis de déduire les éléments à mettre en place pour fabriquer notre système de recommandation.

Les deux derniers chapitres sont consacrés aux apports de cette thèse. Dans le chapitre 5 nous présentons une méthodologie afin de produire des recommandations lors de l'exécution de processus. Nous avons proposé une démarche et une architecture logicielle qui permet de suggérer une activité à l'utilisateur à la fin de chaque activité. L'objectif étant de proposer l'activité qui permet d'optimiser un critère donné. Notre méthodologie est basée sur trois étapes. La première consiste à collecter les données des exécutions puis à réaliser une transformation afin d'obtenir des traces modélisées au format *eXtensible Event Stream*. La seconde étape consiste à extraire des connaissances sur les processus passés réalisés par les autres utilisateurs en utilisant l'algorithme *Inductive Miner*. Cela nous permet de construire un modèle des processus métier. Ces deux étapes se font en amont de l'exécution. La troisième étape se fait en cours d'exécution et consiste à déterminer, à partir du modèle des processus et de la séquence réalisée jusque là par l'utilisateur, quelle activité doit être réalisée pour finir le processus en optimisant des critères préalablement définis.

Nous avons validé notre architecture sur un exemple. Il s'agit d'analyser des parcours d'étudiants en IUT Informatique et de produire une recommandation. Cette étude nous permet de s'assurer que notre système fonctionne en construisant une analyse de bout en bout. C'est-à-dire que nous partons des données accumulées sur sept années et nous rejouons la dernière année en simulant des recommandations. Cette étude est satisfaisante pour établir les relations entre les différentes étapes de notre recommandation. Toutefois, il ne s'agit pas de processus faiblement structurés.

Le chapitre 6 termine cette thèse en abordant un cas d'étude plus ambitieux pour lequel aucun processus métier n'a été prédéfini par un expert. Nous souhaitions voir s'il est possible d'identifier des comportements et/ou stratégies d'utilisateurs vis-à-vis d'un système. Nous nous sommes placés dans un contexte d'apprentissage où l'apprenant est impliqué dans une simulation d'un micro-monde. Ce cas d'étude nous a permis de montrer comment adapter notre méthodologie et comment prendre en compte les données contextuelles.

Ce cas d'étude a donné lieu à une expérimentation où deux groupes ont utilisé notre simulateur. Le premier sans recommandation, ce qui nous a permis de constituer un ensemble de traces d'exécution qui ont servi à extraire les connaissances nécessaires sur nos processus métier. Le second groupe a bénéficié de notre système de recommandation. Nous avons observé que dans ce dernier groupe le critère de performance était amélioré et que la dispersion était plus faible. Les phénomènes d'essais/erreurs se trouvent considérablement réduits.

7.2 Bilan

Pour conclure ce manuscrit, nous pouvons dire que nous avons traiter une problématique autour du pilotage de processus métier qui met en jeu trois domaines de recherche : les systèmes à base de traces, la fouille de processus et la recommandation. Ainsi dans notre première contribution, nous avons construit une méthodologie qui s'appuie sur ces trois domaines et, en deuxième et troisième contributions, nous avons montré qu'elle était applicable dans différents contextes. En particulier nous avons construit un cas d'étude d'une application pour laquelle aucun processus métier n'était défini *a priori*. Nous avons pu extraire des connaissances sur les comportements des utilisateurs et construire un modèle des processus d'usage qui nous a servi à mettre en place une recommandation.

Notre contexte concerne les processus métier. Toutefois nous nous sommes restreint à des domaines où l'interaction avec l'utilisateur est importante. En particulier nous nous focalisons sur des systèmes pour lesquels les choix de l'utilisateur influent sur le déroulement du processus. Le concepteur n'étant, en amont, là que pour définir des activités et pour les placer dans des séquences que l'utilisateur ordonne. Nous nous sommes plus particulièrement concentrés sur le domaine de l'apprentissage dans le cadre des simulations de micro-monde. Il s'agit de laisser l'utilisateur expérimenter des stratégies pour résoudre un problème et ainsi acquérir des compétences. Avec notre système, il devient possible de faire bénéficier l'utilisateur de l'expérience des apprentissages passés des autres utilisateurs.

L'objectif de cette thèse était d'explorer la problématique du pilotage de processus afin de donner des pistes pour la fabrication d'un compagnon qui guiderait l'utilisateur dans sa phase de découverte des processus. Nous avons concentré nos efforts sur les aspects faisabilité informatique. Nous avons ainsi chercher à montrer que l'on pouvait combiner trois domaines de recherche pour construire notre raisonnement. Nous avons proposé une méthodologie et nous avons démontré sa faisabilité en fabriquant deux démonstrateurs. Le deuxième a donné lieu à une expérimentation qui nous a permis de juger de sa pertinence en terme d'accompagnement de l'utilisateur.

Nous avons montré qu'il est possible de construire un système de recommandation basé sur des connaissances issues de la fouille de processus. Toutefois si notre architecture est robuste de nombreux détails peuvent être améliorés, en particulier sur les aspects fouille de processus et recommandation. En effet nous avons utilisé les algorithmes et mesures classiques de la fouille de processus, or nous nous plaçons dans un contexte d'utilisation différent de celui pour lesquels ils ont été défini. Il en est de même pour la recommandation où nous ne nous sommes servi que de résultats très élémentaires.

L'expérience acquise au cours de cette thèse nous pousse à orienter nos travaux vers l'aide à la personnalisation de parcours d'apprentissage. En particulier avec la définition des formations en blocs de compétences, la prise en compte du profil de l'apprenant, qu'il s'agisse de

ses connaissances acquises aussi bien que de ses stratégies d'apprentissage, conduit à fabriquer une trajectoire d'apprentissage, et donc une sélection de blocs de formation, qui doit être personnalisée. La méthodologie que nous avons proposée constitue une brique pour fabriquer un tel écosystème.

Cependant, nous avons bien conscience que les aspects liés au domaine de la recommandation n'ont été que trop peu approfondis. Il en résulte que nous n'avons pas cherché à prescrire un chemin à un utilisateur, en lui automatisant les étapes qu'il doit suivre pour atteindre son objectif, nous nous contentons de l'inciter fortement à suivre un chemin que nous avons précalculé sur un modèle de processus découvert.

7.3 Perspectives

Nos travaux portent sur la question de l'accompagnement des utilisateurs lors de l'utilisation de processus métier. Nous avons orientés nos cas d'étude sur des problématiques liées à l'e-éducation. Ce domaine est riche car le marché de la connaissance est varié (formation initiale sur des notions élémentaires, formations initiales sur des compétences avancées, formation tout au long de la vie) et les attentes nombreuses. Il est facile d'imaginer un environnement informatique qui personnalise le parcours d'un apprenant en lui proposant des activités, et donc des stratégies d'apprentissage, en fonction de son profil observé à partir des traces. Cependant, dans un tel système il ne faut pas considérer l'apprenant comme une personne en particulier voire un profil unique mais comme un groupe de personnes différentes et que l'on doit faire apprendre ensemble. La connaissance des profils et comportements permet à l'outil d'envisager la mise en place d'un système d'apprentissage collaboratif.

Pour mettre en place un système d'apprentissage collaboratif il faut bien entendu avoir un profil précis des apprenants ainsi qu'une bonne connaissance des processus d'apprentissage. Cette perspective, à la lumière de nos travaux fait apparaître plusieurs questions de recherche.

Développer une recommandation qui s'appuie sur la connaissance des processus métier. En effet avec le modèle des processus métier nous pouvons développer des stratégies pour permettre à un utilisateur d'atteindre son objectif. Toutefois, dans un tel contexte se pose la question de savoir comment lui prescrire le chemin à suivre et que faire en cas de remise en cause ?

Développer des algorithmes de recommandation capables de traiter le cas des traces avec du bruit et des légères variations. En effet les processus que nous étudions sont collectés à partir d'application pour lesquelles peu voire pas de processus ont été imaginés. S'il est possible d'envisager de retrouver des comportements typiques en fonction du profil de l'utilisateur, il est peu probable que ces derniers se comportent de manière identique. Nous devons donc présenter un modèle de processus qui accepte des varia-

tions. L'utilisation d'algorithme de classification ou de recherche de motifs sont des pistes de recherche qui semblent intéressantes de suivre.

Développer des mesures pour caractériser les comportements des utilisateurs. Il s'agit de développer un ensemble d'indicateurs qui vont permettre d'identifier des comportements, et donc des formes particulières dans les modèles de processus, et de pouvoir les utiliser pour comparer des comportements par rapport à des comportements types.

Les cas des processus faiblement structurés est un cas particulier de notre méthode. Nous avons traité le cas d'un processus structuré et le cas d'un processus non structuré. Quid des processus faiblement structurés ? Comme les extrêmes ont été traités, est-il possible de faire varier les configuration de notre méthode entre la configuration pour les processus structurés et celle des non structurés, afin de traiter les processus faiblement structurés ? Des expérimentations permettraient de répondre à ces questions.

Bibliographie

Accorsi, R. and Stocker, T. (2012), On the exploitation of process mining for security audits : the conformance checking case, dans « 27th Annual ACM Symposium on Applied Computing », ACM, Riva del Garda (Trento), Italy, 1709–1716.

Adomavicius, G. and Tuzhilin, A. (2005), « Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions », IEEE Transactions on Knowledge and Data Engineering **17**(6), 734–749.

Adomavicius, G. and Tuzhilin, A. (2015), Context-aware recommender systems, dans « Recommender systems handbook », Springer, 191–226.

Alchiekh Haydar, C. (2014), Les systèmes de recommandation à base de confiance, thèse de Doctorat, Université de Lorraine.

Arana, J., Hassas, S. and Prié, Y. (2004), MAZETTE : Multi agent MUSETTE for sharing and reusing ontologies, dans « On the Move to Meaningful Internet Systems 2004 : OTM 2004 Workshops », Springer, Agia Napa, Cyprus, 741–752.

Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F., Marrella, A., Mecella, M. and Soo, A. (2017), « Automated discovery of process models from event logs : Review and benchmark », IEEE Transactions on Knowledge and Data Engineering **PP**, 1–20.

Barragáns-Martínez, A. B., Costa-Montenegro, E., Burguillo, J. C., Rey-López, M., Mikic-Fonte, F. A. and Peleteiro, A. (2010), « A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition », Information Sciences **180**(22), 4290–4311.

Bendahmane, M., Elfalaki, B. and Benattou, M. (2016), « Services-oriented model for the regulation of learning », Int. J. Soc. Behav. Educ. Econ. Bus. Ind. Eng **10**(7).

Bétrancourt, M., Guichon, N. and Prié, Y. (2011), Assessing the use of a Trace-Based Synchronous Tool for distant language tutoring, dans « Proceedings of the 9th International Conference on Computer-Supported Collaborative Learning (CSCL 2011) », Vol. 1, Hong-Kong, Chine, 486–493.

Bigham, J. P., Lau, T. and Nichols, J. (2009), TrailBlazer : Enabling blind users to blaze trails through the Web, dans « Proceedings of the 14th international conference on Intelligent user interfaces », Vol. 14, Sanibel Island, Florida, USA, 177–186.

Bobadilla, J., Ortega, F., Hernando, A. and Gutiérrez, A. (2013), « Recommender systems survey », Knowledge-Based Systems **46**(18), 109–132.

Bobadilla, J., Serradilla, F., Hernando, A. and Others (2009), « Collaborative filtering adapted to recommender systems of e-learning », Knowledge-Based Systems **22**(4), 261–265.

Bose, R. P. J. C. and Van Der Aalst, W. M. P. (2009), Context aware trace clustering : Towards improving process mining results, dans « 2009 SIAM International Conference on Data Mining », SIAM, Sparks, NV, USA, 401–412.

Bouraga, S., Jureta, I., Faulkner, S. and Herssens, C. (2014), « Knowledge-based recommendation systems : a survey », International Journal of Intelligent Information Technologies (IJIIT) **10**(2), 1–19.

Bozkaya, M., Gabriels, J. and van der Werf, J. M. (2009), Process diagnostics : a method based on process mining, dans « International Conference on Information, Process, and Knowledge Management. eKNOW'09. », IEEE, Cancun, Mexico, 22–27.

Breese, J. S., Heckerman, D. and Kadie, C. (1998), Empirical analysis of predictive algorithms for collaborative filtering, dans « Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence », Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 43–52.

Buijs, J. C. A. M., van Dongen, B. F. and Van Der Aalst, W. M. P. (2014), « Quality dimensions in process discovery : The importance of fitness, precision, generalization and simplicity », International Journal of Cooperative Information Systems **23**(1), 1–40.

Burke, R. (2002), « Hybrid recommender systems : Survey and experiments », User modeling and user-adapted interaction **12**(4), 331–370.

Burke, R. (2007), Hybrid web recommender systems, dans « The adaptive web », Springer, 377–408.

Carmona, J., Cortadella, J. and Kishinevsky, M. (2010), « New region-based algorithms for deriving bounded Petri nets », IEEE Transactions on Computers **59**(3), 371–384.

Carrer-Neto, W., Hernández-Alcaraz, M. L., Valencia-García, R. and García-Sánchez, F. (2012), « Social knowledge-based recommender system. Application to the movies domain », Expert Systems with Applications **39**(12), 10990–11000.

Castro-Schez, J. J., Miguel, R., Vallejo, D. and López-López, L. M. (2011), « A highly adaptive recommender system based on fuzzy logic for B2C e-commerce portals », Expert Systems with Applications **38**(3), 2441–2454.

Chaachoua, H., Croset, M.-C., Bouhineau, D., Bittar, M., Nicaud, J.-F., Caroline Croset, M. and François, J. (2007), « Description et exploitations des traces du logiciel d'algèbre Aplusix », STICEF **14**(1), 1–26.

Champagnat, R., Leblay, J., Nowakowski, S. and Rabah, M. (2016), Aide à l'analyse des parcours d'apprentissage en IUT par reconnaissance de procédés et recommandations à base de traces, dans « Congès National de la Recherche des IUT », Nantes, France.

Champin, P.-A., Mille, A. and Prié, Y. (2013), « Vers des traces numériques comme objets informatiques de premier niveau », Intellectica-La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo) **1**(59), 171–204.

Champin, P.-A., Prié, Y. and Mille, A. (2003), Musette : Modeling USEs and Tasks for Tracing Experience, dans « ICCBR », Vol. 3, Trondheim, Norway, 279–286.

Clauzel, D., Sehaba, K. and Prié, Y. (2009), Modelling and visualising traces for reflexivity in synchronous collaborative systems, dans « INCOS'09. International Conference on Intelligent Networking and Collaborative Systems, 2009. », IEEE Computer Society, Barcelona, Spain, 16–23.

Cordier, A., Lefevre, M., Champin, P.-A., Georgeon, O. L. and Mille, A. (2013), Trace-Based Reasoning-Modeling Interaction Traces for Reasoning on Experiences., dans « Florida Artificial Intelligence Research Society Conference (FLAIRS) », Florida, USA, 363–368.

Costa-Montenegro, E., Barragáns-Martínez, A. B. and Rey-López, M. (2012), « Which App ? A recommender system of applications in markets : Implementation of the service for monitoring users' interaction », Expert systems with applications **39**(10), 9367–9375.

Crespo, R. G., Martínez, O. S., Lovelle, J. M. C., García-Bustelo, B. C. P., Gayo, J. E. L. and De Pablos, P. O. (2011), « Recommendation System based on user interaction data applied to intelligent electronic books », Computers in Human Behavior **27**(4), 1445–1449.

de Medeiros, A. K. A., Van Dongen, B. F., Van Der Aalst, W. M. P. and Weijters, A. (2004), Process mining : Extending the alpha-algorithm to mine short loops, Technical report, Eindhoven University of Technology, Eindhoven.

Desel, J. and Esparza, J. (2005), Free choice Petri nets, Cambridge university press.

Di Noia, T., Mirizzi, R., Ostuni, V. C. and Romito, D. (2012), Exploiting the web of data in model-based recommender systems, dans « Sixth ACM conference on Recommender systems », ACM, Dublin, Ireland, 253–256.

Diamantini, C., Genga, L. and Potena, D. (2016), « Behavioral process mining for unstructured processes », Journal of Intelligent Information Systems **47**(1), 5–32.

Dix, A. (2009), Human-computer interaction, dans « Encyclopedia of database systems », Springer, 1327–1331.

Dixit, P. (2019), Interactive process mining, thèse de Doctorat, Department of Mathematics and Computer Science. Proefschrift.

Djouad, T., Mille, A. and Benmohammed, M. (2011), SBT-IM : Système à base de traces-Indicateurs d'interactions Moodle, dans « EIAH 2011 », Mons, Belgique, 1–3.

Downey, D., Dumais, S. T. and Horvitz, E. (2007), Models of Searching and Browsing : Languages, Studies, and Application., dans « IJCAI », Hyderabad, India, 2740–2747.

Dyke, G., Lund, K. and Girardot, J. (2010), « Tatiana, un environnement d'aide à l'analyse de traces d'interactions humaines », Technique et Science **29**(10), 1179–1205.

En-Naimi, E. M. and Zouhair, A. (2016), « Intelligent dynamic case-based reasoning using multi-agents system in adaptive e-service, e-commerce and e-learning systems », International Journal of Knowledge and Learning **11**(1), 42–57.

Gras, B. and Boyer, A. (2016), Identifying Grey Sheep Users in Collaborative Filtering : a Distribution-Based Technique, dans « Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization », Halifax, Canada, 17–26.

Han, J., Cheng, H., Xin, D. and Yan, X. (2007), « Frequent pattern mining : current status and future directions », Data Mining and Knowledge Discovery **15**(1), 55–86.

He, J. and Chu, W. (2010), « A Social Network-Based Recommender System (SNRS) », Data Mining for Social Network Data **12**, 47–74.

Heineman, G. T., Pollice, G. and Selkow, S. (2016), Algorithms in a nutshell : a practical guide, " O'Reilly Media, Inc.".

Heraud, J. M. (2002), Pixed : projet d'intégration de l'expérience dans l'enseignement à distance, thèse de Doctorat, Université Claude Bernard Lyon1.

Herlocker, J. L., Konstan, J. A., Terveen, L. G. and Riedl, J. T. (2004), « Evaluating collaborative filtering recommender systems », ACM Transactions on Information Systems (TOIS) **22**(1), 5–53.

Ho, H. N., Rabah, M., Nowakowski, S. and Estraillier, P. (2018), Trace-Based Multi-Cristeria Preselection Approach for Decision Making in Interactive Applications like Video Games, dans « The Digital Turn in Higher Education », Springer, 211–234.

Ho Hoang, N. and Ho, H. N. (2015), Trace-Based Multi-Criteria Decision Making in Interactive Application for Adaptive Execution, Theses, Université de La Rochelle.

Huang, Z., Chung, W., Ong, T.-H. and Chen, H. (2002), A graph-based recommender system for digital library, dans « Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries », ACM, Portland, Oregon, USA, 65–73.

Huang, Z., Zeng, D. and Chen, H. (2007), « A comparison of collaborative-filtering recommendation algorithms for e-commerce », IEEE Intelligent Systems **22**(5), 68–78.

Iegorov, O. (2015), Data Mining Approach to Temporal Debugging of Embedded Streaming Applications, thèse de Doctorat, Université de Grenoble.

Jouck, T., Bolt, A., Depaire, B., de Leoni, M. and Van Der Aalst, W. M. P. (2018), « An Integrated Framework for Process Discovery Algorithm Evaluation », arXiv **1806**(70), 1–14.

Katarya, R. and Verma, O. P. (2018), « Efficient music recommender system using context graph and particle swarm », Multimedia Tools and Applications **77**(2), 2673–2687.

Kavalanekar, S., Worthington, B., Zhang, Q. and Sharda, V. (2008), Characterization of storage workload traces from production windows servers, Seattle, WA, USA, 119–128.

Kimball, R. and Caserta, J. (2011), The Data Warehouse ETL Toolkit : Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data, John Wiley & Sons.

Kluver, D., Ekstrand, M. D. and Konstan, J. A. (2018), Rating-based collaborative filtering : algorithms and evaluation, Springer.

Leblay, J. (2016), Aide à la navigation dans les parcours d'apprentissage par reconnaissance de procédés et recommandations à base de traces, dans « Rencontres Jeunes Chercheurs Environnements Informatiques pour l'Apprentissage Humain », Montpellier, France.

Leblay, J., Rabah, M., Champagnat, R. and Nowakowski, S. (2018), Process-based Assistance Method for Learner Academic Achievement, dans « E-Learning », Proceedings of 12th Multi Conference on Computer Science and Information Systems (MCCSIS'2018), Madrid, Spain, 89–96.

Lee, S. K., Cho, Y. H. H. and Kim, S. H. (2010), « Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations », Information Sciences **180**(11), 2142–2155.

Lee, S., Park, S., Kahng, M. and Lee, S.-G. (2013), « Pathrank : Ranking nodes on a heterogeneous graph for flexible hybrid recommender systems », Expert Systems with Applications **40**(2), 684–697.

Leemans, S. J. J., Fahland, D. and Van Der Aalst, W. M. P. (2013 a), Discovering block-structured process models from event logs-a constructive approach, dans « International conference on applications and theory of Petri nets and concurrency », Springer, Milan, Italy, 311–329.

Leemans, S. J. J., Fahland, D. and Van Der Aalst, W. M. P. (2013 b), Discovering block-structured process models from event logs containing infrequent behaviour, dans « International conference on business process management », Springer, Beijing, China, 66–78.

Leemans, S. J. J., Fahland, D. and Van Der Aalst, W. M. P. (2014), Discovering block-structured process models from incomplete event logs, dans « International Conference on Applications and Theory of Petri Nets and Concurrency », Springer, Tunis, Tunisia, 91–110.

Levandoski, J. J., Sarwat, M., Eldawy, A. and Mokbel, M. F. (2012), LARS : A Location-Aware Recommender System, dans « IEEE 28th International Conference on Data Engineering », Arlington, Virginia, USA, 450–461.

Linden, G., Smith, B. and York, J. (2003), « Amazon. com recommendations : Item-to-item collaborative filtering », Internet Computing, IEEE **7**(1), 76–80.

Liu, X., Dong, M., Ota, K., Yang, L. T. and Liu, A. (2018), « Trace malicious source to guarantee cyber security for mass monitor critical infrastructure », Journal of Computer and System Sciences **98**, 1–26.

Loghin, G.-C. (2008), Observer un Environnement Numérique de Travail pour réguler les activités qui s'y déroulent, thèse de Doctorat, Cotutelle entre l'Université de Savoie et l'Université Technique de Cluj-Napoca, France.

Marty, J.-C. and Mille, A. (2009), Traces, traces d'interactions, traces d'apprentissages : définitions, modèles informatiques, structurations, traitements et usages, Hermes Sciences Publications.

Mbatchou, G. M., Bouchet, F. and Carron, T. (2018), Multi-scenario modelling of learning, dans C. M. Kebe, A. Gueye, A. Ndiaye and A. Garba, eds, « Conférence sur le Recherche en

Informatique et ses Applications », Vol. 249 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Association Sénégalaise des Chercheurs en InformatIque (ASCII), Springer, Ziguinchor, Senegal, 199–211.

McNally, K., O'Mahony, M. P., Coyle, M., Briggs, P. and Smyth, B. (2011), « A case study of collaboration and reputation in social web search », ACM Transactions on Intelligent Systems and Technology (TIST) **3**(1), 4–10.

Melville, P., Mooney, R. J. and Nagarajan, R. (2002), Content-boosted collaborative filtering for improved recommendations, dans AAAI Press, ed., « Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence », Vol. 23, The MIT Press, Edmonton, Alberta, Canada, 187–192.

Mooney, R. J. and Roy, L. (2000), Content-based book recommending using learning for text categorization, dans « Proceedings of the fifth ACM conference on Digital libraries », ACM, 195–204.

Murata, T. (1989), « Petri nets : Properties, analysis and applications », Proceedings of the IEEE **77**(4), 541–580.

Muratet, M., Yessad, A., Carron, T. and Ramolet, A. (2018), « Un système d'aide à l'analyse des traces des apprenants dans les jeux sérieux », Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation **25**(1), 37–63.

Nanopoulos, A., Rafailidis, D., Symeonidis, P. and Manolopoulos, Y. (2010), « Musicbox : Personalized music recommendation based on cubic analysis of social tags », IEEE Transactions on Audio, Speech, and Language Processing **18**(2), 407–412.

Nicaud, J.-F. (1987), Aplusix : un système expert de résolution pédagogique d'exercices d'algèbre, thèse de Doctorat, Paris 11.

Nikolaos, A., Vassilis, K., Georgios, F., Meletis, M. and Eleni, V. (2005), Logging of fingertip actions is not enough for analysis of learning activities, dans « AIED 05. 12th International Conference on Artificial Intelligence in Education », Amsterdam, The Netherlands, 1–8.

Núñez-Valdéz, E. R., Lovelle, J. M. C., Martínez, O. S., García-Díaz, V., de Pablos, P. O. and Marín, C. E. M. (2012), « Implicit feedback techniques on recommender systems applied to electronic books », Computers in Human Behavior **28**(4), 1186–1193.

Park, D. H., Kim, H. K., Choi, I. Y. and Kim, J. K. (2012), « A literature review and classification of recommender systems research », Expert Systems with Applications **39**(11), 10059–10072.

Pazzani, M. J. (1999), « A framework for collaborative, content-based and demographic filtering », Artificial Intelligence Review **13**(5-6), 393–408.

Pazzani, M. J. and Billsus, D. (2007), « Content-Based Recommendation Systems », The Adaptive Web **4321**(Springer 2007), 325–341.

Piccoli, G. (2007), Information systems for managers : texts and cases, Wiley Publishing.

Porcel, C., Tejeda-Lorente, A., Martínez, M. A. and Herrera-Viedma, E. (2012), « A hybrid recommender system for the selective dissemination of research resources in a technology transfer office », Information Sciences **184**(1), 1–19.

Qiang, L. (2013), Modélisation et exploitation des traces d'interactions dans l'environnement de travail collaboratif, thèse de Doctorat, Université Technologique de Compiègne.

Rebuge, Á. and Ferreira, D. R. (2012), « Business process analysis in healthcare environments : A methodology based on process mining », Information systems **37**(2), 99–116.

Ricci, F., Rokach, L. and Shapira, B. (2015), Recommender systems : introduction and challenges, dans « Recommender systems handbook », Springer, 1–34.

Ricci, F., Rokach, L., Shapira, B. and Kantor, P. B. (2011), Recommender Systems Handbook, Cambridge university press.

Roth, M. A., Korth, H. F. and Silberschatz, A. (1988), « Extended algebra and calculus for nested relational databases », ACM Transactions on Database Systems (TODS) **13**(4), 389–417.

Rozinat, A. and Van Der Aalst, W. M. P. (2008), « Conformance checking of processes based on monitoring real behavior », Information Systems **33**(1), 64–95.

Russell, N., Ter Hofstede, A. H. M., Mulyar, N. and Van Der Aalst, W. M. P. (2006), « Workflow control-flow patterns : A revised view », BPM reports **6**(22), 1–134.

Schafer, J. B., Frankowski, D., Herlocker, J. and Sen, S. (2007), Collaborative filtering recommender systems, dans « The adaptive web », Springer, 291–324.

Sehaba, K. (2014), Adaptation dynamique des Environnements Informatiques pour l'Apprentissage Humain interactifs, thèse de Doctorat, Université Lumière Lyon 2.

Serrano-Guerrero, J., Herrera-Viedma, E., Olivas, J. A., Cerezo, A. and Romero, F. P. (2011), « A google wave-based fuzzy recommender system to disseminate information in University Digital Libraries 2.0 », Information Sciences **181**(9), 1503–1516.

Settouti, L. S. (2011), Systèmes à Base de Traces Modélisées : Modèles et Langages pour l'exploitation des traces d'Interactions, thèse de Doctorat, Université Claude Bernard Lyon 1.

Settouti, L. S., Prié, Y., Champin, P.-A., Marty, J.-C. and Mille, A. (2009), A Trace-Based Systems Framework : Models, Languages and Semantics, Technical report.

Settouti, L. S., Prié, Y., Mille, A. and Marty, J.-C. (2006), Systèmes à base de trace pour l'apprentissage humain, dans « Colloque international TICE », Gênes, France, 131–139.

Shani, G. and Gunawardana, A. (2011), Evaluating recommendation systems, dans « Recommender systems handbook », Springer, 257–297.

Song, J., Luo, T. and Chen, S. (2008), Behavior pattern mining : Apply process mining technology to common event logs of information systems, dans « Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on », IEEE, Sanya, China, 1800–1805.

Soualah-Alila, F. (2015), CAMLearn : Une Architecture de Systeme de Recommandation Sémantique Sensible au Contexte. Application au Domaine du M-Learning., thèse de Doctorat, Université de Bourgogne.

Sun, Z., Guo, Q., Yang, J., Fang, H., Guo, G., Zhang, J. and Burke, R. (2019), « Research commentary on recommendations with side information : A survey and research directions », Electronic Commerce Research and Applications **37**, 879–909.

Tan, S., Bu, J., Chen, C. and He, X. (2011), Using rich social media information for music recommendation via hypergraph model, dans « Social media modeling and computing », Springer, 213–237.

Tarus, J. K., Niu, Z. and Kalui, D. (2018), « A hybrid recommender system for e-learning based on context awareness and sequential pattern mining », Soft Computing **22**(8), 2449–2461.

Tarus, J. K., Niu, Z. and Mustafa, G. (2018), « Knowledge-based recommendation : a review of ontology-based recommender systems for e-learning », Artificial intelligence review **50**(1), 21–48.

Toussaint, B.-M. and Luengo, V. (2015), Mining surgery phase-related sequential rules from vertebroplasty simulations traces, dans « Conference on Artificial Intelligence in Medicine in Europe », Springer International Publishing, Pavia, Italy, 35–46.

Van Der Aalst, W. M. P. (1996), « Structural characterizations of sound workflow nets », Computing Science Reports **96**(23), 18–22.

Van Der Aalst, W. M. P. (2016), Data Science in Action, Springer.

Van Der Aalst, W. M. P., Adriansyah, A. and Van Dongen, B. (2012), « Replaying history on process models for conformance checking and performance analysis », Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery **2**(2), 182–192.

Van Der Aalst, W. M. P., Buijs, J. C. A. M. and van Dongen, B. F. (2011), Towards Improving the Representational Bias of Process Mining., dans « International Symposium on Data-Driven Process Discovery and Analysis », Springer, Berlin, Heidelberg, 39–54.

Van Der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G. and Weijters, A. J. M. M. (2003), « Workflow mining : A survey of issues and approaches », Data & knowledge engineering **47**(2), 237–267.

Van Der Aalst, W. M. P., van Hee, K. M., ter Hofstede, A. H. M., Sidorova, N., Verbeek, H. M. W., Voorhoeve, M. and Wynn, M. T. (2011), « Soundness of workflow nets : classification, decidability, and analysis », Formal Aspects of Computing **23**(3), 333–363.

Van Der Aalst, W. M. P. and Weijters, A. (2004), « Process mining : a research agenda », Computers in industry **53**(3), 231–244.

Van Der Aalst, W. M. P., Zhao, J. L. and Wang, H. J. (2015), « Business Process Intelligence : Connecting Data and Processes », ACM Transactions on Management Information Systems (TMIS) **5**(4), 1–7.

Verbeek, H. M. W., Buijs, J. C. A. M., Van Dongen, B. F. and Van Der Aalst, W. M. P. (2010), XES, XESame, and ProM 6, dans « Information Systems Evolution », Springer, 60–75.

Wang, N. (2014), Vers un système de recommandation à partir de traces sémantiques pour l'aide à la prise de décision, dans « Inforsid 2014 », Lyon, France, 29–32.

Wang, N. (2016), Towards a competency recommender system from collaborative traces, thèse de Doctorat, Université de Technologie de Compiègne.

Weijters, A. and Ribeiro, J. T. S. (2011), Flexible heuristics miner (FHM), dans « IEEE Symposium on Computational Intelligence and Data Mining (CIDM) », IEEE, Paris, France, 310–317.

Weijters, A., Van Der Aalst, W. M. P. and De Medeiros, A. K. A. (2006), « Process mining with the heuristics miner-algorithm », Technische Universiteit Eindhoven, Tech. Rep. WP **166**, 1–34.

Winoto, P. and Tang, T. Y. (2010), « The role of user mood in movie recommendations », Expert Systems with Applications **37**(8), 6086–6092.

Wirz, M., Franke, T., Roggen, D., Mitleton-Kelly, E., Lukowicz, P. and Tröster, G. (2012), Inferring crowd conditions from pedestrians' location traces for real-time crowd monitoring during city-scale mass gatherings, dans « 2012 IEEE 21st International Workshop on Enabling Technologies : Infrastructure for Collaborative Enterprises », IEEE, Toulouse, France, 367–372.

Wu, D., Zhang, G. and Lu, J. (2015), « A fuzzy preference tree-based recommender system for personalized business-to-business e-services », IEEE Transactions on Fuzzy Systems **23**(1), 29–43.

Yang, Z., Wu, B., Zheng, K., Wang, X. and Lei, L. (2016), « A survey of collaborative filtering-based recommender systems for mobile internet applications », IEEE Access **4**, 3273–3287.

Yessad, A., Muratet, M. and Carron, T. (2017), Aider à l'analyse du comportement d'un apprenant dans les jeux sérieux, dans « EIAH 2017 », Strasbourg, France.

Yu, Z., Zhou, X., Hao, Y. and Gu, J. (2006), « TV program recommendation for multiple viewers based on user profile merging », User modeling and user-adapted interaction **16**(1), 63–82.

Zaíane, O. R. (2002), Building a recommender agent for e-learning systems, dans « ICCE '02. Proceedings of International Conference on Computers in Education », IEEE, Auckland, NZ, 55–59.

Zarka, R., Champin, P.-A., Cordier, A., Egyed-Zsigmond, E., Lamontagne, L. and Mille, A. (2013), TStore : A Trace-Base Management System using Finite-State Transducer Approach for Trace Transformation, dans S. Hammoudi, L. F. Pires, J. Filipe and R. C. das Neves, eds, « International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2013) », SciTePress, Barcelona, Spain, 117–122.

Zhang, S., Yao, L., Sun, A. and Tay, Y. (2019), « Deep learning based recommender system : A survey and new perspectives », ACM Computing Surveys (CSUR) **52**(1), 5 :1–5 :38.

Zhang, Z.-K., Liu, C., Zhang, Y.-C. and Zhou, T. (2010), « Solving the Cold-Start Problem in Recommender Systems with Social Tags », EPL Journal (Europhysics Letters Journal) **92**(6), 1–6.