Contributions à la modélisation, la réutilisation et la flexibilité des systèmes d'information

Isabelle Mirbel

Habilitation à Diriger des Recherches

présentée devant

L'Université de Nice-Sophia Antipolis

par

Isabelle Mirbel

Contributions à la modélisation, la réutilisation et la flexibilité des systèmes d'information

soutenue publiquement le 25 juin 2008 devant le jury composé de :

Président :

Michel Riveill	Université de Nice-Sophia Antipolis

Rapporteurs :

Corine Cauvet	Université Aix-Marseille
Jean-Pierre Giraudin	Université de Grenoble
Barbara Pernici	Politecnico di Milano (Italie), absente

Examinateurs :

Rose Dieng-Kuntz	INRIA Sophia Antipolis, absente
Michel Léonard	Université de Genève (Suisse)
Colette Rolland	Université Paris I

Table des matières

Table des figures

Chapitre 1

Introduction

L'ingénierie des systèmes d'information fournit des méthodes, des techniques et des outils pour développer et implanter le plus efficacement possible les systèmes d'information. Ce domaine d'activité a ouvert un champ considérable d'activités de recherche et de développement d'applications avec aujourd'hui de nouvelles perspectives pour aborder la complexité croissante des systèmes d'information d'entreprise [24]. Ce mémoire a pour objectif de montrer comment mes activités de recherche se sont inscrites dans ce domaine de recherche riche et en perpétuelle évolution. J'ai choisi pour cela de donner un aperçu synthétique et précis de mes travaux de recherche selon trois angles d'approche : la séparation des dimensions durant l'activité de modélisation des systèmes d'information, la réutilisation dans l'ingénierie des systèmes d'information et la flexibilité dans l'ingénierie des systèmes d'information. Je développerai plus avant ces aspects dans la section 1.3 de ce chapitre. Avant cela, j'évoquerai dans la section 1.1 les motivations de mon travail dans le domaine de l'ingénierie des systèmes d'information. Puis, je donnerai dans la section 1.2 un aperçu des projets auxquels j'ai participé et qui ont été mon contexte de travail.

1.1 Motivations

Le terme de développement de système d'information [49] a été introduit dans les années soixante. Il désigne une discipline ayant pour but de concevoir un système d'information de la façon la plus indépendante possible des changements technologiques. Dans son discours, l'auteur distingue clairement l'étude des informations (*infological*) de l'étude des données (*datalogical*). L'étude des informations porte sur l'abstraction et la modélisation des informations nécessaires au système d'information pour fonctionner. L'étude des données porte sur la description de la structure et des opérations internes au système d'information à développer. Ces tâches correspondent respectivement aux phases d'analyse et de conception que nous connaissons aujourd'hui dans le cycle de développement d'un système d'information. Les activités d'analyse et de conception constituent donc la phase du processus de développement ayant pour but de transformer un énoncé informel (traduisant les besoins des utilisateurs) en une description non ambigu du fonctionnement futur du système, afin d'en faciliter la réalisation. Pour mener à bien ces activités, des modèles de produit permettant de spécifier le système à développer et des modèles de processus permettant d'indiquer la démarche à suivre pour construire des diagrammes selon les modèles produits ont été proposés. Les premières approches [28,75] cherchaient à permettre la spécification d'applications construites autour

d'une base de données relationnelle. Dans les années quatre-vingt dix, l'orientation objet des langages de programmation et des bases de données se répercute sur les modèles et démarches proposés [84,9,80] pour couvrir les phases d'analyse et surtout de conception. Les approches basées sur les scénarios et sur les buts s'imposent pour les activités liées à la phase d'analyse. Dans ce contexte j'ai toujours été particulièrement intéressée par les activités d'analyse et de conception qui consistent à réfléchir et à formaliser les besoins d'une organisation pour résoudre un problème spécifique au travers de la mise en place ou de l'adaptation d'un système d'information. Je me suis donc intéressée aux modèles et aux démarches permettant de supporter ce travail de formalisation et d'abstraction et d'en représenter les résultats.

Les changements économiques et technologiques de ces dernières années ont bouleversé profondément la vie des organisations, engendrant la multiplication et la dissémination d'informations hétérogènes. Dans ce contexte, abstraire, organiser, synthétiser l'information pour permettre de piloter les systèmes d'information qui deviennent de plus en plus complexes est une autre de mes motivations. L'information n'est plus rare : elle devient surabondante et très accessible. Le problème est de la retrouver, de la sélectionner, de l'interpréter. Mes travaux de recherche ont également cherché à répondre à ce besoin.

Enfin, la notion de démarche est omni-présente dans les activités d'analyse et de conception. Pour être menées à bien, elles nécessitent d'être accomplies selon des directives aidant à réaliser le travail de formalisation et d'abstraction. Une directive est un ensemble de prescriptions. Le terme de méthode qui vient du grec methodos, signifie « moyen d'investigation ». L'auteur de [42] les décrit comme « une collection de procédures, de techniques, de descriptions de produit et d'outils logiciels pour le support effectif, efficace et consistant du processus d'ingénierie d'un système d'information ». Une méthode est une démarche organisée rationnellement pour aboutir à un résultat. Les méthodes d'analyse et de conception de systèmes d'information sont des démarches dédiées à la réalisation de tels systèmes. Elles sont essentielles d'une part pour structurer la pensée des ingénieurs d'application, c'est-à-dire les personnes qui mettent la démarche en pratique dans leurs tâches quotidiennes, et d'autre part pour structurer les actions dans les projets de développement afin de réaliser les systèmes désirés. L'ingénierie des méthodes [12] a pour but d'apporter des solutions efficaces à la construction, l'amélioration et l'évolution des méthodes de développement des systèmes d'information. Contribuer à représenter les processus de développement afin d'aider les ingénieurs d'application dans les tâches conceptuelles de ces processus a été une autre de mes préoccupations.

1.2 Contextes de recherche

Je distingue trois périodes dans mes activités de recherche. La première période correspond à mon travail de thèse de doctorat. Je ne présenterai pas ce travail en détail dans ce mémoire mais en ferai une rapide synthèse dans la section 1.2.1 afin que le lecteur puisse comprendre l'origine de mes travaux de recherche. La deuxième période correspond à mon séjour post doctoral au *Politecnico di Milano*. Durant cette période j'ai été intégrée dans une équipe de recherche très dynamique de renommée internationale qui m'a permis de travailler sur la modélisation des situations dites exceptionnelles dans les *workflows* et sur la consistance temporelle de présentations multimédias. Je présenterai ces deux contextes de travail dans

les sections 1.2.2 et 1.2.3. Enfin, depuis mon recrutement à l'Université de Nice-Sophia Antipolis, je me suis consacrée à des travaux sur les démarches de développement de système d'information. Je présenterai ce dernier contexte de travail dans la section 1.2.4.

1.2.1 Modélisation et intégration de schémas dans le projet C.O.D.

Le projet C.O.D.

Mon travail de thèse s'est déroulé dans le cadre du projet C.O.D. (Class and Object Definition) qui a eu pour but de proposer une aide aux concepteurs lors de la modélisation d'une application sous forme de schémas de conception orientée objet. Ce projet proposait quatre démarches de conception éventuellement complémentaires.

L'approche par spécialisation était fondée sur le principe d'applications types et de classes d'objet générales. Elle permettait de réutiliser la connaissance du domaine et l'expertise d'un concepteur sur un type d'application donné. C'était une approche précurseur des travaux sur les patrons et les *frameworks* [24].

L'approche par superposition était fondée sur des diagrammes de flots de données et des diagrammes exprimés à l'aide du modèle entité-association. Elle avait pour but de mettre en évidence des classes d'objet en rapprochant les schémas entité-association des diagrammes de flot de données.

L'approche par composition permettait de créer des classes d'objet à partir d'un dictionnaire d'attributs et de méthodes. L'algorithme de composition proposé dans le cadre de ce projet pouvait également être utilisé pour faire du *reverse engineering*.

Enfin, l'approche par généralisation était fondée sur l'utilisation d'exemples et de contre-exemples, pour élaborer un diagramme de classe correspondant à l'application.

Ces démarches étaient toutes fondées sur l'expérience et le savoir-faire des concepteurs. L'idée était de permettre aux concepteurs de réutiliser leur expertise et leurs connaissances des démarches traditionnelles dans le cadre du développement d'applications orientées objet. Grâce à ses différentes démarches, le projet C.O.D. apportait une aide aussi bien à des concepteurs confirmés qu'à des débutants.

J'ai participé à ce projet en proposant un mécanisme d'intégration de schémas de conception orientée objet. En effet, plusieurs concepteurs peuvent être amenés à travailler sur le même schéma de conception du fait qu'ils ont des compétences complémentaires, des perceptions différentes de la réalité ou bien simplement parce que le domaine à modéliser est trop vaste et nécessite la collaboration de plusieurs concepteurs. Chacun d'eux propose donc un schéma qui correspond à une certaine vision du problème. Les différents schémas doivent ensuite être intégrés dans un même schéma résultat.

Un processus d'intégration de schémas de conception orientée objet

Dans le processus que j'ai proposé, je me suis attachée à prendre en considération la dimension sémantique des schémas conceptuels. Pour cela, j'ai proposé un thésaurus flou, inspiré de travaux réalisés dans le domaine de la linguistique, pour capturer la dimension sémantique des schémas de conception orientée objet et la prendre en considération durant le processus d'intégration. Par la suite, les travaux de l'époque sur les thésaurus ont donné lieu à des courants de recherche sur les ontologies à la fois dans le domaine des bases de données et dans celui de l'ingénierie des connaissances.

L'intégration de schémas permet de prendre en compte les points de vue des différents concepteurs impliqués dans une modélisation complexe ou de taille importante. Elle permet également au cours du temps d'intégrer à des schémas de conception les changements résultant de l'évolution des systèmes d'information. Le but recherché est de produire un schéma unique et le plus complet possible à partir de schémas multiples d'une même réalité perçue selon des points de vues différents. Le mécanisme d'intégration de schémas que nous avons proposé est dédié à l'intégration de schémas de conception orientée objet, c'est-à-dire qu'il tient compte des particularités d'un tel paradigme et tente d'en respecter les principes. Afin de rendre notre travail utilisable dans un maximum de contextes, nous ne nous sommes pas attachés à une notation orientée objet en particulier. Nous avons au contraire basé notre travail sur les concepts partagés par un grand nombre de notations, à savoir les notions d'attribut, méthode, classe, lien de référence et hiérarchie d'héritage. Le processus d'intégration que nous avons proposé est fondé sur une étude simultanée des dimensions structurelle et sémantique des schémas. Cette dernière dimension a été particulièrement privilégiée dans notre approche, lors de l'étape de comparaison des éléments des schémas, de résolution des conflits et de présentation du résultat de l'intégration. Nous sommes partis du principe qu'aucune équivalence n'était fournie *a priori* par les concepteurs pour réaliser l'intégration.

Un processus d'intégration se décompose en quatre étapes (qui sont généralement précédées par une mise sous forme canonique) :
- comparaison des schémas,
- résolution des conflits,
- superposition (intégration) et
- choix et présentation du résultat.

Durant l'étape de comparaison, nous avons distingué les couples d'éléments dont la dimension sémantique doit être privilégiée sur la dimension structurelle et ceux dont nous devons privilégier la dimension structurelle sur la dimension sémantique. En effet, si nous examinons par exemple un couple d'attributs (l'un provenant d'un des deux schémas à intégrer et l'autre provenant du second schéma), nous devons privilégier l'aspect sémantique sur l'aspect structurel car deux attributs ayant le même domaine et exprimés dans la même unité ne constituent pas nécessairement des attributs identiques. En revanche, lorsque nous examinons un couple de classes, le fait qu'elles soient constituées d'éléments semblables doit être privilégié sur le fait qu'elles portent des noms proches. L'aspect structurel doit donc primer sur l'aspect sémantique.

Comme à ce stade du processus il n'est pas toujours possible de choisir le rapprochement qui permettra la meilleure intégration, il se peut que plusieurs rapprochements soient conservés pour un élément donné et, qu'en conséquence, un résultat soit construit à partir de chacun

d'eux. Cela constitue l'arbre des solutions dont chaque feuille porte un sous-ensemble cohérent et non redondant d'appariement des éléments du premier schéma avec ceux du second schéma.

L'étape de résolution des conflits consiste à résoudre les conflits détectés durant la phase de comparaison en choisissant la représentation d'un des deux schémas. Quand cela est possible, il faut choisir la représentation la plus riche des deux.

Enfin, l'étape d'intégration consiste à superposer les hiérarchies d'héritage et les graphes de référence obtenus après résolution des conflits dans les deux schémas de départ. À l'issue de l'étape de comparaison, nous obtenons plusieurs ensembles cohérents, chacun d'eux pouvant mener à un résultat possible de l'intégration de deux schémas initiaux. À partir de chacun de ces ensembles cohérents, nous déduisons un schéma résultat en appliquant successivement les règles de résolution de conflits et de superposition. Nous obtenons donc à ce stade du mécanisme plusieurs schémas résultats, parmi lesquels un est choisi comme résultat de l'intégration.

Nos travaux sur le mécanisme d'intégration que je viens de présenter brièvement ont été publiés dans [56,62,55,57]. Une des originalités de notre approche était de prendre en compte la dimension sémantique des schémas à l'aide d'un thésaurus flou. Je vais détailler ce thésaurus dans la suite de cette section.

Un thésaurus flou pour l'intégration de schémas

Pour représenter la sémantique des mots utilisés pour désigner les attributs, les méthodes, les classes et les liens dans les schémas conceptuels, nous avons proposé un thésaurus flou. Il s'inspire de travaux réalisés dans le domaine de la linguistique. Notre but fut de proposer une structure permettant de définir la sémantique des mots, afin d'obtenir une mesure de la similarité et de l'ambiguïté entre deux termes lors de l'intégration de schémas de conception. Dans la structure que nous proposons, les mots (ou termes) sont distingués des concepts (ou sens). Chaque concept est représenté par une phrase, la plus explicite possible, expliquant le sens des mots qui lui sont attachés. La différence est faite entre mots et concepts afin de pouvoir travailler à un niveau (celui des concepts) où la connaissance peut être définie de façon précise et ainsi éviter de rencontrer des problèmes de synonymie et d'homonymie. Dans les schémas de conception, certains mots sont utilisés plus fréquemment que d'autres. Citons l'exemple des mots `identifiant` ou `nom`. Un poids sémantique est rattaché à chaque mot du thésaurus et varie de façon inversement proportionnelle à la fréquence d'apparition du mot dans les schémas de conception (tous schémas et tous domaines confondus). Il est mis à jour à chaque intégration réalisée. Les concepts sont liés entre eux par des relations conceptuelles de nature sémantique. Les mots sont reliés aux concepts par des relations d'interprétation. Nous distinguons deux sortes de relations conceptuelles : généricité et composition. La première permet d'indiquer l'appartenance d'un concept c1 à l'ensemble des éléments plus spécifiques qu'un concept c2. Le concept de *femme* par exemple appartient à l'ensemble des éléments plus spécifiques que le concept de *personne*. La relation est orientée de l'élément le plus spécifique vers l'élément le plus générique. Dans notre cas, elle est orientée de c1 vers c2. Cette appartenance est graduée à l'aide d'un degré de vraisemblance compris entre 0 et 1 qui indique le degré d'appartenance du couple de concepts à la relation de généricité. Il y a plusieurs types de relations de composition, selon que la relation est transitive ou non et selon qu'elle

autorise une décomposition en éléments hétérogènes ou pas. Des relations d'interprétation entre les mots et les concepts permettent par ailleurs d'expliciter les sens donnés aux mots. La relation d'interprétation est une relation floue dont la fonction d'appartenance associe à chaque élément un degré compris entre 0 et 1 indiquant le degré d'appartenance du couple mot-concept à la relation « a-pour-sens ».

L'ensemble des concepts et des relations conceptuelles sont communs à tous les concepteurs. Les relations d'interprétation, les mots et les coefficients associés aux relations conceptuelles sont propres à chaque concepteur. La somme des coefficients apparaissant sur les relations d'interprétation partant d'un même mot doit toujours être égale à 1: ces coefficients montrent la probabilité d'utilisation pour le concepteur de chaque sens du mot. Les expertises des différents concepteurs sont fusionnées dans le thésaurus en faisant la moyenne pondérée des coefficients qui apparaissent dans plusieurs expertises. Cette façon de procéder a l'avantage d'autoriser chaque concepteur à saisir ses connaissances sans avoir à les attacher directement à celles des autres concepteurs. D'autre part, elle permet de pondérer l'importance des expertises les unes par rapport aux autres. L'exploitation du thésaurus permet de détecter les mots voisins et les mots ambigus pendant le processus d'intégration. Le degré d'ambiguïté d'un mot, qui varie entre 0 et 1, est défini par la formule de Shanon en théorie de l'information [85]. Le degré et le type de voisinage entre deux mots est fonction des concepts auxquels ils sont liés. Nous distinguons deux cas de figure :

- Si les deux mots ont au moins un concept commun, alors leur type de voisinage est la synonymie. Dans ce cas, le degré de voisinage correspond au minimum des degrés d'interprétation des deux relations d'interprétation liées au concept commun.
- Si un mot a au moins un concept directement ou indirectement lié à un concept lui-même lié à l'autre mot, alors :
  - si la liaison est directe, le voisinage est du type de la relation conceptuelle reliant les deux concepts et le degré de voisinage correspond au minimum des coefficients associés aux relations d'interprétation et aux relations conceptuelles impliquées dans le chemin qui constitue la liaison directe entre les deux termes,
  - sinon, il est nécessaire d'étudier la transitivité potentielle entre les concepts qui constituent le chemin allant d'un des concepts liés au mot à un des concepts liés à l'autre mot. Nous avons étudié les différents cas de figure possibles et proposé un type de voisinage correspondant: pas de voisinage (pas de transitivité), voisinage générique, voisinage spécifique, voisinage composé ou voisinage composant. Nous avons également proposé une façon de calculer le coefficient de voisinage (pour le détail de ce calcul voir [55]).

Nous ne conservons comme résultat de l'examen du voisinage entre deux mots que le meilleur résultat, en terme de degré de voisinage, parmi ceux obtenus dans tous les cas de figure.

Durant le processus d'intégration, un seuil de similarité compris entre 0 et 1 et un seuil d'ambiguïté, compris lui aussi entre 0 et 1, sont définis par le concepteur en charge de l'intégration afin de lui permettre de fixer le niveau de finesse de l'intégration. Les résultats de notre travail sur la proposition d'un thésaurus flou pour l'intégration de schémas ont été publiés dans [54,53,55].

Après avoir soutenu ma thèse de doctorat, je suis partie en postdoc au *Politecnico di Milano*

où je suis restée un an et demi. J'ai participé au projet WIDE et au projet CHOROCHRONOS. Je vais maintenant présenter le contexte de ces deux projets dans les sections qui suivent.

1.2.2 Workflow et réutilisation : le projet WIDE

Nous appellons *workflow* la modélisation et la gestion informatique de l'ensemble des tâches (ou activités) à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier. Un workflow permet de décrire et de contrôler un flot d'activité, c'est-à-dire les différentes tâches à accomplir par les différents acteurs, les délais éventuels entre les tâches et les informations à fournir à chacun des acteurs pour lui permettre de réaliser les tâches qui lui sont affectées. Pour un processus de publication sur le web par exemple, il s'agit de modéliser les tâches de l'ensemble de la chaîne éditoriale. Un système de gestion de workflow (WFMS) est un dispositif logiciel permettant d'exécuter une ou plusieurs spécifications de workflow (*workflow schema*). Par spécification, nous entendons la description du flot d'activité souhaité, les documents et informations associés aux activités, la liste des acteurs nécessaires à la réalisation de ces activités et également la description des situations non souhaitées qui peuvent survenir et qu'il faut gérer comme par exemple la terminaison anormale de l'exécution d'une instance de workflow ou la suspension d'une activité.

Notre travail sur la modélisation de workflow a été initié dans le cadre du projet européen Esprit WIDE[1] qui avait pour but de proposer un WFMS construit à partir d'un système de gestion de base de données [2].

Mon travail a porté plus précisément sur la modélisation des situations dites exceptionnelles (c'est-à-dire non idéalement souhaitées mais qu'il est nécessaire de gérer) et sur les aspects méthodologiques de la spécification de workflow. Pour cela, nous avons proposé un langage générique de spécification de situations exceptionnelles, sous forme de patrons pouvant être conservés dans une bibliothèque dédiée. Nous avons proposé des patrons pour représenter :

- des situations exceptionnelles récurrentes dans un domaine d'application particulier. Par exemple, dans les applications liées à la réservation de service, nous pouvons vouloir modéliser un morceau de processus correspondant à la préparation d'un document et à sa soumission, ainsi que les situations exceptionnelles qui lui sont associé. Nous pouvons également vouloir modéliser le morceau de processus qui consiste à demander un service à l'aide d'une réservation et à obtenir la réservation de ce service.
- des situations exceptionnelles plus génériques, indépendantes du domaine d'application. Par exemple, nous pouvons vouloir modéliser ce qui doit être entrepris lorsque la durée d'exécution d'une activité dépasse une durée maximale prévue ou lorsque des informations qui sont saisies le sont en dehors des intervalles de valeur autorisés.

En plus du langage générique de spécification de situations exceptionnelles sous forme de patron que je détaillerai dans le chapitre 2 de ce mémoire, nous avons proposé WERDE (*Workflow Exceptions Reuse and Design Environment*), un prototype permettant la spécification et la réutilisation de spécification de situations exceptionnelles ainsi qu'une bibliothèque de patrons construits en analysant un ensemble de cas réels fournis par nos partenaires dans le cadre du projet européen WIDE [8]. Cette bibliothèque sera détaillée dans le chapitre 3 de ce

[1] http://cordis.europa.eu/esprit/src/20280.htm

mémoire.

1.2.3 Le projet Chorochronos

Durant mon séjour postdoctoral au Politecnico de Milan, j'ai également fait partie du projet Chorochronos[2], qui était un réseau européen (*training and mobility research*) dont l'objectif était d'étudier la conception, l'implantation et les applications des systèmes de gestion de bases de données spatio-temporelles [3].

Les applications spatio-temporelles traitent de situations dans lesquelles des informations spatiales, temporelles ou spatio-temporelles interviennent. Dans ce spectre d'applications, se trouvent par exemple les systèmes d'information géographiques [91], comme les applications cadastrales dans lesquelles sont représentées des parcelles de terrain, des régions, etc. ainsi que les contraintes qui leur sont associées. Les applications de contrôle des objets en mouvement [39] sont un autre exemple d'applications spatio-temporelles dans lesquelles les objets se déplacent dans un espace et envoient en permanence leur position exacte. Les applications multimédias [92] sont aussi spatio-temporelles. Toutes ces applications ont comme spécificité commune de devoir gérer des objets qui changent de forme, de taille, de structure ou de position. La nature complexe des notions de temps et d'espace ainsi que celle tout aussi complexe des interactions spatio-temporelles et des contraintes d'intégrité entre objets spatio-temporels rendent difficile leur modélisation. Il n'en existe pas de spécification claire et acceptée par tous. Il y a des divergences sur l'identification d'un objet spatio-temporel, dont les propriétés mais également la structure peuvent être amenées à changer et qui peut même dans certains cas migrer d'une structure à une autre. Différents modèles de temps, d'espace et d'espace-temps ont également été proposés. Les modèles temporels varient en terme de structure, qui peut être linéaire, cyclique ou en branche. Certains sont bornés alors que d'autres ne le sont pas. Nous distinguons des modèles basés sur un temps discret, d'autres sur un temps dense ou continu. Enfin, il existe des modèles basés sur la notion d'intervalle de temps [4,14] et d'autre basés sur la notion de point dans le temps. Les modèles d'espace se distinguent les uns des autres par le nombre de dimensions appréhendées et par leur continuité. Enfin, les modèles d'espace temps se distinguent par le fait qu'ils sont dédiés à la modélisation des mouvement ou des changements (comme la distorsion, l'évolution ou les changements structurels). Ces différents modèles affectent bien sûr la façon de spécifier les objets, leurs propriétés spatio-temporelles ainsi que les relations qui permettent de les lier les uns avec les autres [13].

Dans ce contexte, les objectifs principaux du projet Chorochronos étaient de :

- stimuler des recherches dans le domaine des bases de données spatiales et des bases de données temporelles,
- permettre aux chercheurs de ces communautés d'améliorer leur compréhension de leurs travaux respectifs et
- permettre à ces chercheurs de coopérer avec des chercheurs d'autres disciplines (environnement, multimédia, transport, etc.).

Pour atteindre ces objectifs, le projet Chorochronos a rassemblé des travaux trés variés allant de questions liées aux ontologies, à la structuration et à la représentation du temps et

[2] http://www.dbnet.ece.ntua.gr/~choros/

de l'espace, aux modèles et aux langages de requête pour les systèmes de gestion de bases de données spatio-temporelles, aux interfaces graphique pour les informations spatio-temporelles, aux algorithmes de traitement des requêtes, aux structures de stockage et techniques d'indexation ou encore aux architectures et techniques d'implantation dédiées aux systèmes de gestion de bases de données spatio-temporelles.

C'est sur la consistance temporelle des applications multimédias qu'a porté plus précisément mon travail. Un document multimédia interactif est constitué d'objets multimédias (images, sons, vidéos, etc.) présentés selon un scénario donné. Durant la présentation du document, ces objets peuvent être transformés spatialement et/ou temporellement pour être présentés selon les spécifications données par l'auteur de la présentation. Au moment de concevoir le document, l'auteur se concentre donc sur des aspects distincts de la présentation à différents instants et il est difficile dans ce contexte d'assurer la consistance de la présentation dans son ensemble, ce qui peut avoir pour conséquence de rendre le document partiellement ou totalement inexécutable. Aussi avons nous proposé une démarche permettant de vérifier la consistance temporelle d'un tel document au moment de sa conception. Cette approche se démarquait des approches concurrentes par le fait qu'elle permettait de vérifier la consistance temporelle qualitative et quantitative des présentations, comme nous le verrons dans le chapitre 2 où je détaillerai ce travail.

1.2.4 Les démarches de développement

Depuis que j'ai été recrutée à l'Université de Nice-Sophia Antipolis, j'ai travaillé sur l'ingénierie des méthodes [12,41,90,76] qui a pour but d'apporter des solutions efficaces à la construction, l'amélioration et l'évolution des méthodes de développement de systèmes d'information.

Parallèlement à l'évolution des modèles proposés pour la modélisation des systèmes d'information, qui se sont enrichis et sont devenus de plus en plus nombreux afin de capturer les différentes facettes d'un système d'information (structurelle, dynamique ou fonctionnelle), la notion de démarche de développement de système d'information a également été précisée. Des efforts ont été faits pour mieux formaliser les éléments qui constituent une méthode [12,41,90,76]. Une démarche de développement se définit comme un ensemble cohérent de modèles de produit et de modèles de processus. Le produit est le résultat à atteindre. L'ensemble des concepts et des contraintes nécessaires pour représenter un produit constitue un modèle de produit. Le processus est l'ensemble de activités qu'il faut enchaîner pour construire un produit. Un modèle de processus est un ensemble de prescriptions pour fabriquer un produit d'un type donné. Nous parlons de directives. L'ingénierie des méthodes s'attache à étudier et améliorer ces processus de production et d'adaptation des modèles de produits et de processus qui constituent une démarche de développement.

Deux familles d'approche sont actuellement proposées en ingénierie des méthodes [76] : les approches par métamodélisation et les approches situationnelles. Dans les approches par métamodélisation, les modèles de produit et les modèles de processus sont les instances d'un même métamodèle. Ces approches visent à permettre la comparaison de méthodes par référence à un métamodèle commun [90,76].

Les approches situationnelles prônent la construction de nouvelles méthodes par assemblage

de fragments prédéfinis, validés et accessibles dans une bibliothèque de fragments de démarche. Les méthodes sont alors construites à la volée à partir de ces fragments pour satisfaire aux spécificités d'un projet [42,11,72,5]. Cette famille d'approche trouve sa justification dans l'analyse de la pratique qui montre qu'une méthode n'est jamais suivie à la lettre mais au contraire adaptée à la situation de chaque projet : des étapes sont sautées, d'autres ajoutées, les documents et points de décision sont ajustés, etc. Toutes sortes de facteurs relatifs au projet, à la technologie, à l'expertise de l'équipe, aux caractéristiques du domaine d'application induisent une adaptation de la méthode au profil du projet. Cette famille d'approche a l'avantage de préserver les acquis et aussi d'apporter une certaine flexibilité en fournissant les moyens d'adapter une méthode aux besoins spécifiques d'un environnement (projet ou organisation).

Les travaux sur l'ingénierie des méthodes situationnelles ont cherché plus précisément à permettre la réutilisation de fragments de démarches existantes dans le cadre de la construction de nouvelles démarches mieux adaptées aux spécificités d'un projet ou d'une organisation. C'est dans ce contexte que s'est inscrit mon travail de recherche après mon recrutement à l'Université de Nice-Sophia Antipolis. Les approches actuelles en matière de méthodes situationnelles sont dédiées aux ingénieurs méthode, c'est-à-dire aux personnes en charge de la mise en place de la méthode dans l'entreprise. Ma contribution dans ce domaine a cherché à recentrer ce type d'approche sur les ingénieurs d'application (c'est-à-dire sur les personnes qui mettent la démarche en pratique dans leurs tâches quotidiennes) en leur permettant de configurer une méthode pour leurs besoins spécifiques. Pour mener à bien ce travail j'ai été amenée à collaborer d'abord avec l'entreprise Amadeus S.A.S.[3] dans le cadre du projet Jecko[4] puis avec l'équipe MATIS[5] du C.U.I. de Genève dans le cadre du projet Sesame.

Le projet Jecko est le fruit d'une collaboration entre le Laboratoire I3S et Amadeus S.A.S. durant laquelle nous avons cherché à proposer une approche contextuelle des activités conceptuelles du processus de développement en place chez Amadeus S.A.S. Notre effort a porté sur :

- la proposition de directives concrètes et précises pour mener à bien les activités conceptuelles du processus de développement,
- la proposition de moyens d'adapter les directives aux spécificités des projets de développement et
- l'amélioration de l'accès aux directives au travers d'une fragmentation du processus de développement.

Nous sommes partis des premières phases du processus de développement mis en place chez Amadeus S.A.S qui sont les suivantes : analyse des besoins, analyse des objets et du domaine métiers, étude de l'architecture du système à développer, spécification des composants et spécification interne. Nous avons découpé chacune de ces phases en quatre étapes permettant de mieux préciser l'intention des directives qui leur sont associées. Nous avons distingué : une étape préliminaire, une étape principale, une étape de raffinement et une étape complémentaire. L'étape préliminaire rassemble des directives permettant de mieux appréhender les activités de l'étape principale de la phase. L'étape principale rassemble des directives sur le cœur du travail de la phase considérée. L'étape de raffinement consiste à

[3] http://www.amadeus.com/fr/fr.html
[4] http://www.i3s.unice.fr/~mirbel/jecko/jecko.html
[5] http://matis.unige.ch/

réorganiser et raffiner les spécifications produites durant l'étape principale. Enfin, l'étape complémentaire rassemble des activités pour compléter les spécifications produites durant l'étape principale. Nous avons ensuite proposé des fragments de méthode, chacun attaché à une étape d'une phase du processus. Ces fragments renferment des directives précises et concrètes pour supporter les activités conceptuelles du processus de développement à l'aide de la notation UML. Afin de permettre une certaine adaptabilité du processus de développement et en examinant les projets de développement d'Amadeus S.A.S., nous avons isolé trois facteurs susceptibles d'influencer la façon dont les activités conceptuelles du processus de développement doivent être appréhendées : la présence d'un système existant (legacy system), le fait de devoir concevoir, développer et intégrer une base de donnée dans le système à développer et le fait de devoir concevoir, développer et intégrer une interface homme machine graphique dans le système à développer. Pour chacun de ces facteurs, nous avons proposé un profil UML dédié rassemblant des extensions du langage UML ayant pour but de faciliter leur prise en considération durant les activités conceptuelles du processus de développement. Nous avons également proposé un ensemble de directives pour mettre en œuvre ces trois profils. Le détail de ces facteurs sera présenté dans le chapitre 2 de ce mémoire. Enfin, parmi les fragments que nous avons attachés aux différentes étapes du processus de développement, nous avons séparé les fragments primaires contenant des directives utiles à tous les types de développement de ceux spécifiques à un des critères isolés et proposant des directives dédiées à la prise en compte de ce critère. L'intérêt de cette fragmentation des directives et de leur qualification est d'offrir une certaine flexibilité dans l'accès à la connaissance méthodologique, comme cela sera développé dans le chapitre 4 de ce mémoire. Ce travail s'est concrétisé, en plus des publications sur le sujet, par la proposition de profils et de directives dédiés et accessibles sur l'intranet d'Amadeus S.A.S. et la proposition d'un ensemble de fragment de démarche disponible sur le site du Laboratoire I3S [6].

Le projet Sesame (Smart Environment for SituAtional Method Engineering) est le résultat d'une collaboration avec l'équipe MATIS du C.U.I. de Genève afin de proposer un environnement pour l'ingénierie des méthodes situationnelles. Cet environnement a la particularité de combiner deux approches complémentaires : une approche de construction de méthode par assemblage de fragments et une approche de configuration de méthode au travers de la définition d'un itinéraire. Au cœur de cet environnement se trouvent une bibliothèque de fragments de démarche de développement et une structure permettant de conserver des critères pertinents de qualification des fragments de démarche et des situations de réutilisation de ces fragments. Cet environnement est dédié à la fois aux ingénieurs méthode afin de leur permettre de construire une méthode spécifique à un projet ou à une organisation en assemblant des fragments parmi ceux proposés dans la bibliothèque et aux ingénieurs d'application en leur permettant de configurer la méthode spécifique au projet ou à l'organisation en y inscrivant leur propre itinéraire.

Dans le contexte de ce projet, mon travail a plus précisément porté sur :

- la définition d'une structure permettant de mieux représenter les caractéristiques du contenu d'un fragment de méthode et les caractéristiques du besoin méthodologique d'un ingénieur d'application,
- la proposition d'une mesure permettant d'apparier des fragments de méthode à des

[6] http://www.i3s.unice.fr/ mirbel/jecko/jecko.html.

besoins méthodologiques et
– la proposition d'une démarche de configuration de méthode basée sur les itinéraires.

Je reviendrai sur la solution que nous avons proposée pour caractériser les fragments et les situations de réutilisation dans le chapitre 3. Je montrerai également dans ce chapitre comment cette structure est utile durant le processus de construction d'une démarche de développement. Enfin, dans le chapitre 4, j'expliquerai comment cette structure permet d'apporter de la flexibilité dans l'ingénierie des méthodes.

1.2.5 Synthèse

Ces différents contextes de travail m'ont donné la possibilité de couvrir des aspects très variés et très riches du développement de système d'information. Ils m'ont permis de travailler à la fois sur des projets plutôt appliqués et ayant des partenaires industriels (le projet WIDE et le projet JECKO) et sur des projets plus académiques (le projet CHOROCHRONOS et le projet SESAME). Ils m'ont également permis de participer à des projets de grande ampleur impliquant beaucoup de partenaires (le projet WIDE et le projet CHOROCHRONOS) et à des projets de plus petite taille (le projet JECKO et le projet SESAME).

Je vais maintenant revenir sur les 3 axes selon lesquels j'ai souhaité présenter mes activités de recherche dans ce mémoire.

1.3 Axes de recherche

Le domaine de l'ingénierie des systèmes d'information dans lequel s'est inscrit mon travail de recherche constitue un contexte à la fois riche de beaucoup de travaux, d'évolutions, de changements et cependant encore jeune et faisant face à beaucoup de défis à relever, notamment celui de la complexité croissante et de la globalisation des systèmes d'information. Dans ce cadre, la nécessité de bien structurer l'information, de la partager et de l'accéder de façon flexible me semblent être trois points forts sur lesquels il convient de travailler. Dans ce mémoire, je vais donc m'attacher à montrer comment j'ai effectivement répondu à ces trois exigences de structuration, de partage et de flexibilité au travers des différents travaux de recherche auxquels j'ai participé.
Même si plusieurs de mes travaux m'ont permis de contribuer à chacune de ces trois thématiques à des moments différents et au travers de projets différents, ces thèmes suivent dans l'ensemble l'évolution chronologique de mon travail. Mes premières contributions sur l'intégration de schémas de conception orientée objet et sur les documents multimédias étaient essentiellement accès sur des problématiques de structuration des informations. Mes travaux sur la modélisation des situations dites exceptionnelles dans les workflows et sur les démarches de développement de systèmes d'information ont répondu à une problématique de réutilisation. Le thème de la flexibilité a été pleinement abordé et de façon croissante dans mes travaux sur les démarches de développement. Cependant, le thème de la réutilisation était déjà présent dans mon travail de thèse où l'idée même d'intégration avait, entre autres pour but, d'intégrer des schémas existants à de nouveaux schémas dans un contexte incrémental de modélisation. Le thème de la flexibilité était également présent dans mon travail sur l'intégration dans lequel j'ai proposé un mécanisme de vue permettant un accès personnalisé au schéma résultat. De même, mes travaux sur la modélisation des situations dites exceptionnelles dans les workflows

abordaient déjà le thème de la flexibilité au travers des mécanismes d'adaptation et d'instanciation que nous avons proposés pour les patrons pour les situations dites exceptionnelles.

1.3.1 La structuration et la séparation des dimensions

Le premier angle d'approche que j'ai choisi pour présenter mon travail de recherche est celui de la structuration et de la séparation des dimensions dans la modélisation des systèmes d'information, exposé dans le chapitre 2 de ce mémoire. Au travers des différents travaux de recherche auxquels j'ai participé, j'ai travaillé sur différents types de système d'information (des workflows, des documents multimédias et des logiciels de gestion) et abordé différents aspects de leur modélisation : le paradigme objet dans les schémas de classe, la modélisation des situations dites exceptionnelles dans les workflows, la modélisation du temps dans les documents multimédias, la modélisation des aspects liés aux interfaces homme machine graphiques et au développement à partir d'applications existantes (*legacy application*) dans les logiciels de gestion. J'ai tenté de comprendre et d'exploiter ces dimensions dans la perspective de contribuer à en améliorer leur représentation et leur prise en compte.

J'ai contribué à l'amélioration de la modélisation des situations dites exceptionnelles attachées aux workflows au travers de leur spécification sous forme de règles actives distinctes du flot normal d'activité et de la proposition d'une démarche associée. J'ai également contribué à proposer une approche pour vérifier la consistance de la dimension temporelle intrinsèque aux documents multimédias et j'ai proposé une façon d'étiqueter des fragments de méthode de développement afin d'en dégager les informations susceptibles d'en améliorer la mise en pratique dans le contexte particulier d'un projet. J'ai aussi cherché, de façon de plus en plus nette au cours de mes travaux de recherche, à favoriser la modélisation des informations permettant le partage et la réutilisation des éléments manipulés dans le domaine de l'ingénierie des systèmes d'information. En effet, la complexité croissante des systèmes d'information et l'accélération du rythme de leur évolution nécessitent de tenir compte des expériences passées et de les réutiliser le plus souvent possible. Dans ce sens, j'ai contribué à proposer un formalisme de représentation générique des situations exceptionnelles dans les workflows et une façon de représenter à l'aide de fragments une démarche de développement. Produire des éléments réutilisables de systèmes d'information nécessite de penser différemment leur structure. C'est pourquoi je présenterai mes travaux sur cet aspect de la réutilisation dans la partie dédiée à la structuration et à la séparation des dimensions de ce mémoire.

1.3.2 Le partage et la réutilisation dans les systèmes d'information

Le deuxième point de vue que j'ai choisi sur mes activités de recherche est celui du partage et de la réutilisation. Comme je l'ai évoqué en ce qui concerne la structuration des informations, la complexité croissante des systèmes d'information et l'accélération du rythme de leur évolution imposent qu'ils soient conçus de manière modulaire, à base de composants. Cela pose un certain nombre de problèmes en matière de production d'éléments réutilisables. Mais l'activité de développement d'application par réutilisation de composants pose aussi un certain nombre de problèmes auxquels nous avons tenté de répondre. Dans mon travail, j'ai contribué sur ce point au travers de la définition d'une bibliothèque de patrons génériques pour la modélisation des situations exceptionnelles dans les workflows, de mécanismes d'adaptation et d'instanciation de ces patrons et de la proposition d'une démarche de construction

de méthode de développement par réutilisation comme cela sera présenté dans le chapitre 3.

1.3.3 La flexibilité dans l'ingénierie des systèmes d'information

Enfin, le dernier angle d'approche que j'ai choisi pour présenter mes travaux de recherche est celui de la flexibilité. Il sera abordé dans le chapitre 4. Dans mon travail de thèse déjà, j'avais abordé le thème de la flexibilité, de façon très simple et très succincte, au travers d'un mécanisme de vue sur le résultat de l'intégration de schémas de conception orientée objet. La complexité croissante des systèmes d'information et l'évolution des technologies liées à Internet ont rendu l'information surabondante et très accessible. Le problème devient alors de la retrouver, de la sélectionner et de l'interpréter. Il est nécessaire de penser l'accès à cette information de façon flexible et personnalisée. C'est à cette problématique que j'ai tenté de répondre en plaçant le thème de la flexibilité au centre de mes préoccupations dans mes travaux sur les démarches de développement de système d'information.

1.4 Organisation du document

Dans la suite de ce mémoire, je vais décrire de façon précise mes contributions aux trois thèmes que je viens d'évoquer. Le chapitre 2 sera dédié à mes contributions en matière de structuration et de séparation des dimensions dans les activités d'analyse et de conception de système d'information. Le chapitre 3 sera consacré à mes contributions à l'ingénierie des systèmes d'information par réutilisation. Le chapitre 4 montrera mes contributions en terme de flexibilité dans les processus de développement de systèmes d'information. Dans le dernier chapitre, je présenterai les perspectives que je vois à ces travaux

Chapitre 2

Structuration de l'information

2.1 La séparation des dimensions

Dans le développement de système d'information, la phase de modélisation est traditionnellement découpée en deux étapes : une première étape qui consiste à produire un schéma conceptuel indiquant ce que le système d'information doit faire et une seconde étape qui consiste à produire un schéma interne expliquant comment le système va effectivement permettre de faire ce qui a été défini dans le schéma conceptuel. Il faut donc distinguer une phase d'analyse suivie d'une phase de conception. Pour représenter ces schémas, des modèles sont utilisés. Ceux utilisés pour construire un schéma conceptuel sont plus proches d'une spécification abstraite de besoins, alors que ceux utilisés pour construire le schéma interne tendent à devenir des spécifications exécutables sur des plates-formes techniques déterminées [36]. Dans le contexte informatique, un modèle [36] est un système formel ou semi-formel qui permet d'établir une abstraction d'un système d'information. Dans le domaine de la modélisation des systèmes d'information, métamodèle signifie modèle de modèle ou modèle pour construire, par instanciation, des modèles. Un diagramme est une représentation d'une description qui capture une sémantique partielle d'un système. Pour éviter des interprétations divergentes un diagramme doit clairement être associé à un modèle [36]. Pour construire des diagrammes, des directives sont données. Une directive est une description plus ou moins formelle de prescriptions, d'actions et de techniques expliquant comment fabriquer le produit de la méthode. L'ensemble des directives nécessaires à la construction du produit constituent une démarche ou une méthode. Des outils sont aussi parfois proposés pour supporter le travail d'analyse et de conception. Il s'agit par exemple d'outils pour réaliser les différents diagrammes, pour aider à gérer la cohérence entre différents diagrammes ou pour gérer les documents décrivant à chaque étape les produits résultats de l'application de la méthode. Les diagrammes permettent de donner une vue abstraite du système à développer et ainsi de réfléchir au système à mettre en place, de lever les ambiguïtés, de partager l'information entre les différents acteurs du processus de développement. Le choix du modèle (et de la démarche) influence la façon dont le problème est appréhendé et la solution trouvée. Différents modèles ont été proposés afin de :

- appréhender le système selon des paradigmes différents : fonctionnel [82], systémique [75,81,40] ou objet [84,22,1] par exemple,
- couvrir plus particulièrement certaines étapes du processus de développement, comme par exemple la capture des besoins [48,95] ou l'analyse des besoins [82,75,22],
- répondre aux spécificités des domaines d'application, comme dans le cas des profils UML

Enterprise Computing ou *Real Time Computing* [1] et
- s'approcher de technologies particulières en intégrant leurs spécificités dans les modèles comme c'est le cas dans l'approche MDA pour les modèles spécifiques (*Platform Specific Model*[2]).

En même temps, les systèmes à construire sont devenus de plus en plus complexes. Ils ne peuvent plus être compris dans leur globalité et nécessitent l'utilisation de plusieurs modèles différents. Comme cela est rappelé dans [36], l'intelligibilité du compliqué se fait par simplification alors que l'intelligibilité du complexe ne peut se construire que par modélisation. Un système compliqué peut être simplifié en ne retenant que les éléments essentiels. Au contraire, la nature complexe d'un système est inhérente aux combinaisons et coordinations nombreuses entre éléments. Un système complexe doit être modélisé pour être compris. Dans ce contexte, il est important que les modèles, démarches et outils proposés permettent aux concepteurs d'appréhender chaque dimension pertinente du système à développer et cela de façon adéquate, systématique et en accord avec les autres dimensions du système. L'une des difficultés en multimodélisation est dans la coordination de modèles décrivant les aspects différents afin d'établir un modèle unifié supposé complet du système [36].

J'ai contribué à cette problématique au travers de trois des projets de recherche auxquels j'ai participé :
- Le premier, le projet WIDE, portait sur le développement de processus métiers automatisés (workflows). Pour ma part, j'ai plus précisément travaillé sur les aspects conception et modélisation de spécification de workflow. La spécificité de cette contribution a été de proposer une approche favorisant la distinction entre la modélisation des situations dites exceptionnelles et la modélisation du flot normal d'activité. Dans le projet WIDE, les situations exceptionnelles sont implantées sous forme de règles actives [26]. Cela a permis de gagner en terme de lisibilité durant la phase de modélisation. La majorité des approches de l'époque étaient basées sur une formalisation à l'aide des réseaux de Petri, ce qui les pénalisait en terme de lisibilité. Cela a aussi permis de faciliter le passage de la modélisation à l'implantation.
- Le second projet au travers duquel j'ai contribué à structurer et séparer des dimensions importantes du système à développer est le projet CHOROCHRONOS. Dans le cadre de ce projet, j'ai travaillé sur la conception de document multimédia. Au moment de concevoir un tel document, le concepteur se concentre sur des aspects distincts de la présentation du document à différents instants. Il est difficile, dans ce contexte, d'assurer la consistance de la présentation dans son ensemble, ce qui peut avoir pour conséquence de rendre le document partiellement ou totalement inexecutable. Aussi avons-nous proposé une approche permettant d'isoler la dimension temporelle et d'en vérifier la consistance.
- Enfin, le troisième projet dans lequel j'ai contribué à mettre en évidence des dimensions particulières des systèmes d'information pour en améliorer la modélisation est le projet JECKO. Ce projet, mené en collaboration avec Amadeus S.A.S., avait pour but d'isoler des caractéristiques des systèmes logiciels susceptibles d'influencer la façon dont les activités conceptuelles doivent être appréhendées dans le processus de développement d'Amadeus S.A.S. et de proposer des solutions pour en améliorer la prise en compte dès les premières phases du processus.

[1] http://www.omg.org/technology/documents/profile˙catalog.htm
[2] http://www.omg.org/mda

Je vais maintenant décrire plus précisément chacune de ces contributions.

2.1.1 L'exemple des situations exceptionnelles dans les workflows

Un système de gestion de workflow (WFMS) a pour but de permettre la planification de tâches et leur affectation à des agents (humains ou non) pour qu'elles puissent être exécutées (de façon manuelle ou automatique) selon la description du fonctionnement attendu du workflow. La figure 2.1 montre un exemple de workflow. Dans cet exemple sont décrites les différentes étapes nécessaires à la réservation d'un voyage. Lorsqu'un nouveau client souhaite mettre en place un voyage, une personne de l'agence de voyage définit avec lui les éléments du voyage (sélection et réservation des modes de transport et d'hébergement). Si le client décide d'interrompre la réservation, l'exécution du workflow se termine, sinon elle se poursuit par l'activation en parallèle de l'envoi d'un courrier de confirmation et de l'envoi d'un courrier de demande de paiement suivi de l'attente de ce paiement. Lorsque ces deux activités sont terminées, les documents nécessaires au voyage sont préparés et envoyés au client. Dans le

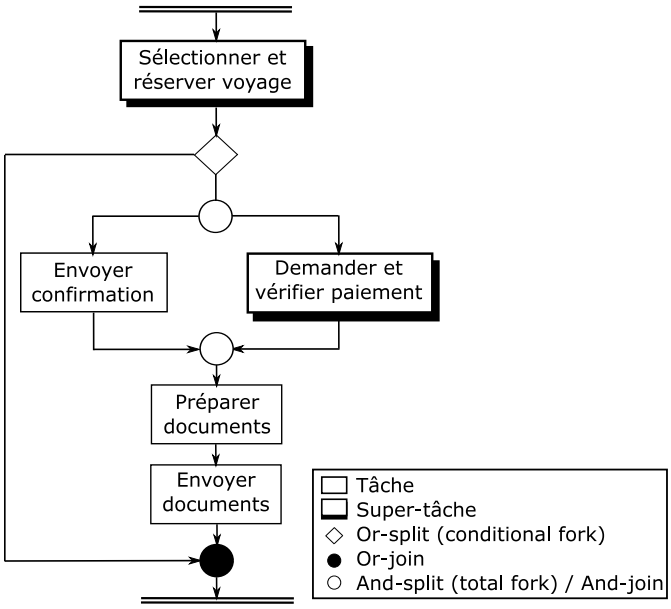

FIG. 2.1 – Un exemple de workflow

contexte du projet WIDE, mon travail a porté plus précisément sur la façon de décrire le fonctionnement attendu du workflow et plus particulièrement sur la modélisation des situations dites exceptionnelles. Dans l'exemple de workflow de la figure 2.1, nous pouvons vouloir ajouter des contraintes supplémentaires comme par exemple le fait de pouvoir annuler la réservation dans le cas où le paiement du client n'est pas reçu après un certain laps de temps. Nous pouvons également vouloir nous assurer que la personne qui va préparer et envoyer les documents de voyage est bien la même que celle qui a défini les éléments du voyage avec le client. Par situation exceptionnelle, nous entendons une réponse à un événement idéalement non souhaité durant l'exécution du workflow. Idéalement, l'annulation d'une réservation pour cause de non paiement n'est pas souhaitée dans l'exemple de la figure 2.1. Nous entendons

également toutes les règles de gestion contraignant l'exécution du workflow. Le fait que la définition des éléments du voyage et l'envoi des documents sont deux tâches qui doivent être effectuées par la même personne est un exemple de règle de gestion. Ces cas de figure sont traditionnellement représentés par des conditions et des boucles dans la spécification du workflow. Prendre en compte ces situations tend à compliquer le schéma et à perdre en lisibilité et en compréhension. Aussi, dans notre approche, ces situations dites exceptionnelles sont représentées sous forme de règles actives décrites séparément du flot normal d'activité auquel elles se superposent.

Pour aider à la modélisation de ces situations exceptionnelles (ou règles de gestion au sens large), nous avons proposé une démarche basée sur l'utilisation de patrons. Dans la suite de cette section, je vais d'abord présenter le modèle WIDE sur lequel nous nous sommes appuyés et plus particulièrement le modèle de processus et le modèle des exceptions. Ensuite, je décrirai la démarche que nous avons proposée pour concevoir des workflows. J'expliquerai plus particulièrement notre contribution pour la modélisation des situations exceptionnelles.

Le modèle WIDE

Le modèle proposé dans le cadre du projet WIDE pour spécifier le fonctionnement souhaité d'un workflow est constitué de cinq modèles : un modèle de processus, un modèle organisationnel, un modèle d'information, un modèle transactionnel et un modèle d'exception. Je ne décrirai pas ici le modèle transactionnel qui n'est pas intervenu dans notre travail sur la modélisation des situations exceptionnelles. Les autres modèles auxquels je n'ai pas personnellement contribué sont succinctement décrits ici pour permettre une meilleure compréhension de mes contributions dans le cadre du projet WIDE.

Le **modèle de processus** décrit l'aspect comportemental du workflow, c'est-à-dire la façon dont une instance de workflow (*case*) est censée évoluer entre le moment où elle est démarrée et celui où elle est définitivement arrêtée. L'élément principal dans la description comportementale d'un workflow est la *tâche*. Cet élément est atomique et représente une activité manuelle ou automatisée dont la planification et l'exécution sont assistées par le WFMS. Une tâche est caractérisée par un nom, une description du travail qui doit être réalisé pour effectuer la tâche, un ensemble de rôles permettant de déterminer quels agents du modèle organisationnel peuvent exécuter la tâche, un ensemble d'informations accessibles pour la tâche (formulaires, documents, répertoires) et un ensemble de droits indiquant les actions possibles sur la tâche (l'ouvrir, y naviguer, la suspendre, la terminer, la déléguer, la réaffecter, etc.). Quand une instance de workflow est démarrée (symbole de début de spécification du workflow), la première tâche dans la description comportementale du workflow est exécutée, c'est-à-dire planifiée et affectée à un agent. L'affectation est faite par le *dispatcher* en fonction des rôles associés à la tâche. Cette dernière peut être soit affectée à un acteur (*pushed*) soit placée dans un espace où les tâches sont partagées par plusieurs agents (*shared task desk*) afin qu'un des agents la sélectionne (*pulled*). Lorsque l'exécution de cette première tâche est terminée, le flot d'activité passe à la / aux tâche(s) suivante(s) selon la description donnée dans la spécification du workflow et ainsi de suite. Quand une tâche connectée au symbole de fin de spécification du workflow se termine, c'est l'instance du workflow toute entière qui est terminée. L'exécution d'une instance de workflow peut être distribuée sur plusieurs sites. Dans la description comportementale du workflow, les tâches sont reliées les unes aux autres soit directement (il s'agit

alors d'une exécution séquentielle des tâches) soit via des connecteurs:
- *fork* permet, dans le flot de contrôle, d'indiquer l'exécution de plusieurs activités en parallèle (*Total fork* ou *AND-split*) ou bien le fait de choisir conditionnellement une activité à exécuter parmi plusieurs (*Conditional fork* ou *OR-split*),
- *join* permet, lorsque plusieurs activités ont été lancées en parallèle, d'indiquer d'attendre la fin de leurs exécutions à toutes pour continuer l'exécution du workflow (*Total join* ou *AND-join*) ou bien d'attendre la fin de l'exécution de seulement k d'entre elles (*Partial join* ou *k-AND-join*) et
- *cycle* permet de représenter des boucles dans le flot de contrôle.

Des exemples de ces connecteurs sont donnés dans la figure 2.1.

Il est également possible d'inclure des *tâches d'attente* (*wait task*), des *super-tâches* et des *multi-tâches*. Les *tâches d'attente* ne réalisent aucune action, n'ont pas besoin d'être affectées à un acteur et réagissent à l'arrivée d'un document ou d'une donnée, en fonction du temps ou de l'occurrence d'un événement. Les notions de *sous processus*, *super-tâche* et *transaction métier* sont trois façons différentes de rendre modulaire un workflow. Un sous-processus dispose de paramètres en entrée et en sortie et permet la réutilisation de spécifications de workflow. Une super-tâche permet d'agréger plusieurs tâches, n'a pas de possibilité de passage de paramètre et ne dispose que des informations de la tâche parent. Elle ne peut donc pas constituer un élément de modélisation réutilisable. Enfin, les tâches rassemblées dans une transaction métier constituent une unité d'exécution au sens transactionnel du terme. Dans l'exemple de la figure 2.1, `Sélectionner et réserver voyage` et `Demander et vérifier paiement` sont des super-tâches, ce qui signifie qu'elles sont composées de tâches plus spécifiques qui doivent également être décrites dans le document de spécification. La notion de *multi-tâche* permet de spécifier la nécessité de démarrer l'exécution de plusieurs instances d'une même tâche de façon concurrente. Dans le workflow présenté dans la figure 2.1, la super-tâche `Demander et vérifier paiement` peut par exemple être composée d'une tâche `Préparer facture` suivie d'une multi-tâche `Vérifier facture` suivie des tâches `Envoyer facture` et `Attendre paiement`. La tâche `Vérifier facture` peut être modélisée sous forme d'une multi-tâche pour indiquer que plusieurs personnes sont amenées de façon concurrente à vérifier le contenu de la facture.

Le **modèle organisationnel** permet de décrire les ressources de l'organisation dans laquelle le workflow est déployé et la façon dont ces ressources sont organisées. Les entités qui constituent ce modèle sont:
- Les *acteurs* qui représentent des personnes ou des machines capables de réaliser une tâche du workflow,
- Les *groupes* qui représentent des classes d'acteurs partageant les mêmes caractéristiques (du point de vue organisationnel),
- Les *fonctions*, c'est-à-dire des classes de groupe ou d'acteur ayant des caractéristiques communes au vu de ce qu'elles peuvent et sont autorisées à faire,
- Les *équipes*, qui sont des listes de fonctions.

Les relations qui existent entre les concepts sont de deux types:
- les relations qui permettent de lier les éléments du modèle organisationnel à ceux du modèle de processus et
- les relations entre entités du modèle organisationnel.

Elles permettent d'organiser les entités. Parmi les agents, certains sont distingués du fait du rôle qu'ils jouent dans le fonctionnement d'une instance de workflow :

- L'exécuteur d'une instance de workflow (*Case executor*) qui est autorisé à lancer l'exécution d'un workflow,
- le responsable d'une instance de workflow (*Case responsible*) qui veille au bon déroulement de l'instance et en est responsable,
- l'exécuteur d'une tâche (*Task executor*) qui est responsable de l'exécution d'une tâche particulière dans une instance de workflow,
- le responsable d'un groupe (*Group supervisor*) qui gère un groupe et ses participants,
- le concepteur d'un workflow (*Workflow designer*),
- l'administrateur d'un domaine de workflow (*Workflow administrator*), qui est responsable du WFMS et également de définir la structure de l'organisation et les acteurs travaillant dans cette organisation.

Le **modèle d'information** décrit les éléments de documentation nécessaires à la spécification et au fonctionnement du workflow. Il est constitué de variables (accessibles par les différentes tâches d'une même instance de workflow) et de documents (c'est-à-dire d'ensembles d'informations explicitement utilisées, créées ou modifiées par un agent pour réaliser une tâche). Ces documents peuvent être des formulaires, des documents contrôlés par d'autres applications, des répertoires permettant de combiner des documents, etc.

Le **modèle d'exception :** La notion de situation exceptionnelle (ou exception dans la suite de cette section) sur laquelle nous avons travaillé est à prendre au sens large. Elle regroupe l'ensemble des contraintes définies sur le workflow qui peuvent être exprimées sous la forme *événements, conditions, actions*. Elle inclut donc aussi le traitement de la terminaison anormale d'une instance de workflow, de la suspension d'une tâche, du démarrage (qui peut également être modélisé comme une réponse à une événement), de l'arrêt et de la répétition de l'exécution d'une spécification de workflow ou de l'occurrence périodique d'une tâche dans une spécification de workflow. Une exception peut être associée dans la spécification du workflow, à une tâche, une super-tâche, un sous-processus ou une instance de workflow dans son ensemble. Définir une exception associée à la spécification d'un workflow signifie définir un comportement commun à toutes les tâches décrites dans la spécification du workflow. Le fait de vouloir suspendre toutes les tâches d'une instance de workflow à un instant précis est un exemple de ce type d'exception.

Dans le projet WIDE, les exceptions sont spécifiées dans le langage Chimera-Exc, langage dérivé du langage de requête orienté objet Chimera [26] et développé dans le cadre du projet européen IDEA [25]. Chimera permet de manipuler des descriptions orientées objet de workflow. Ces descriptions sont stockées dans un système de gestion de base de données orienté objet. Trois catégories de classe d'objet sont distinguées pour modéliser un workflow :

- les classes pour décrire l'organisation et les informations nécessaires au fonctionnement du workflow (modèle organisationnel et modèle d'information),
- les classes dédiées à la description du comportement du workflow (modèle de processus) et
- les classes dédiées à la description des événements (modèle de processus).

À chaque instance d'un workflow en cours d'exécution correspond une instance de classe dédiée

à la description du workflow. Cette instance est créée au moment du démarrage de l'instance du workflow et conserve les valeurs des attributs associés à cette instance. Dans Chimera-Exc, les exceptions peuvent faire référence aux informations contenues dans les instances des différentes classes décrites ci-dessus. L'intérêt de spécifier les exceptions à l'aide de règles actives est double :

- d'une part cela permet une bonne spécification des situations exceptionnelles (c'est-à-dire des règles de gestion au sens large) au niveau conceptuel et
- d'autre part cela permet une traduction facilitée et séparée de cet aspect de la description du workflow lors d'une implantation du workflow dans un WFMS sur un système de gestion de base de données (comme c'est le cas de celui proposé dans le cadre du projet WIDE).

Une règle active est constituée d'une partie *événements*, d'une partie *conditions* et d'une partie *actions*. La partie *événements* de la règle permet de spécifier le ou les événements susceptibles de déclencher une exception. Plusieurs événements peuvent être indiqués dans la partie *événements*. Dans ce cas, la règle est exécutée dès qu'un des événements est détecté. La partie *conditions* permet de spécifier précisément comment déterminer si l'événement détecté correspond bien à la situation exceptionnelle à traiter. Enfin, la partie *actions* permet de spécifier les actions à réaliser pour répondre à la situation exceptionnelle identifiée. Les règles actives sont ordonnées les unes par rapport aux autres grâce à des niveaux de priorité définis dans l'absolu et associés à chacune d'elles. L'ordre de déclenchement des règles de même niveau de priorité n'est donc pas déterminé.

La partie *événements* d'une règle active peut porter sur :

- Des événements liés aux données : ils sont instanciés lors de la création, de la modification ou de la suppression de données dans la base de données. Une instance de l'événement `modify(task)`, par exemple, est créée au moment où un attribut d'une instance de la classe `task` est modifiée.
- Des événements liés au workflow : ils sont instanciés lors du démarrage ou de l'arrêt de l'exécution d'une instance de workflow ou d'une tâche du workflow. L'événement `taskStart(myTask)`, par exemple, est instancié quand une instance de la tâche `myTask` est démarrée.
- Des événements temporels : ils sont créés lorsque des dates limites sont atteintes, des périodes de temps sont écoulées. L'événement `elapsed 1 day since taskStart(myTask)` en est un exemple. Il est instancié lorsqu'un jour complet s'est écoulé depuis le démarrage d'une instance de la tâche `myTask`.
- Des événements externes : ils sont instanciés par des applications externes. Un exemple d'événement externe est `raise(cancelReservation)`.

La partie *conditions* d'une règle active permet d'examiner le contenu de la base de données (sans changer ce contenu) sous forme d'une conjonction de prédicats dont les termes peuvent inclure des variables permettant de référencer les objets gérés par le WFMS. Dans une exception qui serait par exemple déclenchée par l'événement `caseStart` (indiquant le démarrage d'une instance de workflow), la correspondance avec l'instance de workflow démarrée est obtenue via la condition : `case(C), occurred(caseStart C)`. Les instances affectées par les événements déclarés dans la partie événements d'une règle sont référencées dans la partie conditions à l'aide du prédicat `occurred`. Si le résultat de l'évaluation de la condition in-

dique qu'aucune correspondance ne peut être établie avec des instances démarrées, alors la condition est considérée comme non satisfaite et la partie actions de la règle active ne sera pas exécutée. Dans le cas contraire, les correspondances établies peuvent être exploitées dans la partie actions de la règle afin que les actions qui y sont spécifiées soient exécutées sur les instances identifiées dans la partie conditions.

La partie *actions* d'une règle active permet de spécifier les réactions souhaitées en réponse à la détection d'une situation exceptionnelle. Les réactions possibles peuvent consister en des actions de manipulation des objets (création, modification, suppression) ou des opérations à exécuter par le WFMS comme la planification, l'affectation ou la délégation des tâches[3]. `startCase(mySchema, startingParameter)` est un exemple d'action spécifiant qu'en réponse à la détection d'un événement et à la satisfaction des conditions requises dans la spécification de l'exception, une autre instance de workflow, *mySchema*, doit être démarrée avec les paramètres *startingParameter*. Dans l'exemple de la figure 2.1, l'annulation des réservations dans le cas où le paiement du client n'est pas reçu après un certain laps de temps pourrait par exemple lancer l'exécution d'une instance de workflow dédiée à l'annulation d'une réservation. `notify(T.executor, "deadline is approaching")` est un autre exemple d'action qui consiste à envoyer un message à un agent.

La figure 2.2 donne un exemple complet d'une règle qui est activée si l'exécution de la supertâche `Sélectionner et réserver voyage` de l'exemple de la figure 2.1 est trop longue. Lorsque la durée maximale autorisée pour la tâche `Sélectionner et réserver voyage` est atteinte, un événement est généré. Il est capturé dans la partie événements de la règle : `elapsed(maxTaskDuration) since taskStart`. Dans cette règle, la partie conditions permet de typer les éléments (`temporalEvent(E)`, `task(T)`), de déterminer l'instance de tâche sur laquelle la règle doit être appliquée (`task(T), occurred(ev1,E), E.dependsOnTask=T`) et de vérifier que l'instance est bien toujours active (`T.state="running"`). Si la condition est vérifiée, la partie actions est alors exécutée et consiste à envoyer un message à l'agent responsable de l'exécution de la tâche : `notify(T.executor,"...")`.
Pour une description plus détaillée du modèle de spécification des exceptions, voir [21,37]. Pour

```
define     trigger saleSpeed
events     ev1:elapsed(maxTaskDuration) since
           taskStart("Sélectionner et réserver voyage")
condition  temporalEvent(E), task(T), T.state="running",
           occurred(ev1,E), E.dependsOnTask=T
actions    notify(T.executor,"Exécution trop lente: Demander
           au client de se décider")
end
```

FIG. 2.2 – Exemple de règle active pour modéliser une exception

une description détaillée des modèles de référence WIDE, voir [37].

[3]Ces réactions peuvent entraîner le déclenchement d'autres règles actives. Il est donc nécessaire de pouvoir vérifier qu'aucune interaction non souhaitée ne survient. Ce point sera abordé dans la section 3.1.3 du chapitre 3.

La démarche de conception WIRES

Le modèle WIDE qui vient d'être présenté permet de distinguer clairement cinq dimensions fondamentales dans la description de workflow : le processus, l'organisation, les informations, les transactions et les exceptions. Je vais maintenant détailler la démarche que nous avons proposée pour modéliser des workflows à l'aide de ce modèle. Cette démarche propose une approche systématique pour d'une part décider si la mise en place d'un workflow est souhaitable dans une organisation et d'autre part guider les concepteurs du workflow de la phase d'analyse des besoins jusqu'à la phase de choix du produit commercial pour implanter le workflow. À l'époque où ce travail fut réalisé, l'aide méthodologique pour la modélisation de workflow proposée dans la littérature portait sur les phases de conception et d'implémentation. Dans ce contexte, nous avons proposé la démarche WIRES qui permet de :

- prendre en considération les processus métier et les systèmes d'information existant dans le processus de conception d'un workflow afin que ce dernier soit intégré à son contexte,
- modéliser de façon distincte des situations exceptionnelles du flot normal d'activité afin d'améliorer la lisibilité des spécifications de workflow,
- tirer profit des modélisations antérieures, notamment en ce qui concerne les exceptions, en proposant un langage de spécification de patron d'exceptions, une bibliothèque de ces patrons et une démarche par réutilisation associée et
- réduire l'écart entre modélisation et implémentation en proposant un paradigme de représentation uniforme pour les exceptions.

La démarche WIRES se décompose en trois phases : une phase d'analyse, une phase de conception et une phase de transformation vers un système commercial de gestion de workflow. Je vais détailler chacune de ces phases dans la suite de cette section.

La **phase d'analyse** consiste à identifier les processus métiers candidats et à identifier les pré et post-conditions ainsi que les buts associés à chacun d'eux. Pour identifier les processus métiers candidats, nous avons défini une liste de critères de « workflo-abilité » [19]. D'après ces critères, la mise en place d'un workflow est souhaitable lorsque :

- l'exécution des tâches dont il est constitué, la planification de ces tâches et l'affectation de ressources à ces tâches peuvent être automatisées,
- une coordination entre différentes unités organisationnelles est requise et implique plusieurs utilisateurs,
- le processus métier analysé nécessite des périodes d'attente, un contrôle automatisé de son exécution ou l'exécution de certaines activités avant des dates limites,
- des personnes différentes dans des unités organisationnelles différentes et physiquement éloignées sont impliquées dans un processus métier, afin d'en assurer la coordination et de garantir la synchronisation des échanges de documents nécessaires au bon déroulement du processus métier.

Au contraire, un tel support ne sera pas souhaitable si les activités qui constituent le processus métier demandent l'utilisation de logiciels spécifiques avec peu ou pas d'interaction, si le processus métier est simple (constitué d'une ou deux activités seulement) ou s'il n'est pas facilement accessible aux acteurs interagissant avec lui (par exemple dans le cas de travail hors site). Dans le cas d'un processus métier peu souvent effectué, il faut également se poser

la question de l'intérêt d'un support à base de workflow. Il ne faut pas que la conception et la mise en place du workflow demandent trop de travail comparé à ce que nécessite l'exécution du processus métier tel qu'il est au moment de l'étude.

L'identification des processus métiers candidats s'accompagne de l'étude du nombre et de la nature des points de connexion entre les sous-processus et de l'étude du nombre d'unités organisationnelles impliquées dans leur exécution. La mise en place d'un workflow est conseillée pour chaque groupe de sous-processus faiblement couplés avec les autres, ayant un grand nombre de points de connexion à l'intérieur du groupe et exécutés dans différentes unités organisationnelles. L'identification des pré et post-conditions ainsi que des buts associés aux processus métiers candidats se fait à l'aide d'une matrice processus/but permettant d'indiquer pour chaque processus métier les buts qu'il satisfait. Les systèmes d'information et les applications externes qui vont interagir avec le workflow sont également étudiés durant cette phase d'analyse.

Le but de **la phase de conception** est de spécifier à l'aide du modèle WIDE le fonctionnement de chaque workflow identifié durant la phase d'analyse. Pour cela, il faut identifier les différents fragments de processus qui ont des caractéristiques homogènes en terme d'accès dans l'organisation et évaluer la localisation idéale pour chaque fragment du processus examiné. Cette étape du processus de conception dont le résultat est un ou plusieurs modèles de processus est complétée par une étape de modélisation des situations exceptionnelles. Afin d'aider les concepteurs à recenser toutes les situations exceptionnelles nécessaires à une modélisation complète d'un workflow et cela de façon systématique, nous avons proposé une classification de ces situations selon trois catégories : les exceptions indépendantes de la spécification d'un workflow particulier, les exceptions spécifiques à la spécification d'un workflow et les exceptions dédiées au démarrage ou à la terminaison d'une instance de workflow.

Les exceptions indépendantes de la spécification d'un workflow affectent toutes les instances de workflow en cours d'exécution dans un WFMS. Dans cette catégorie d'exception, nous distinguons des exceptions universelles (*predefined system-level exceptions*) et des exceptions dédiées à un domaine métier (*business rules*). Les contraintes exprimant des politiques d'accès aux informations ou aux traitements sont généralement fonction du domaine métier. Elles sont donc traduites sous forme d'exceptions métiers. En revanche, le fait d'indiquer que les tâches affectées à un acteur qui quitte l'organisation doivent être réaffectées à d'autres acteurs de l'organisation est une règle indépendante du domaine métier et que l'on souhaite voir appliquée à l'ensemble des spécifications de workflows. Nous parlons alors d'exception universelle.

Les exceptions spécifiques à la spécification d'un workflow correspondent généralement à des contrôles sur des dates limites et des durées maximales d'exécution des tâches. Elles couvrent également la spécification des actions à réaliser en réponse à l'occurrence d'un événement externe déterminé et la spécification des procédures automatiques de relance.

Les exceptions dédiées au démarrage ou à la terminaison d'un workflow permettent de contrôler le démarrage et l'arrêt des instances de workflow.

Les flots d'activité exceptionnels associés à un workflow sont des portions de workflow qui n'appartiennent pas au flot normal d'activité. Ces portions représentent généralement des

réponses à des événements idéalement non prévus (comme le non paiement du client dans l'exemple de la figure 2.1) ou des règles de gestion qui contraignent la façon dont le workflow doit être exécuté (comme le fait de s'assurer que la personne qui va préparer et envoyer les documents de voyage est bien la même, toujours dans l'exemple de la figure 2.1). Ces portions de workflow sont généralement caractérisées par le fait que :

- elles sont asynchrones,
- elles modifient le flot normal d'activité,
- elles sont substitutives et non additives et
- elles font partie de la sémantique de l'application (au même titre que les contraintes d'intégrité).

Le fait que ces situations exceptionnelles fassent partie de la sémantique de l'application justifie le fait qu'elles soient prises en compte dès la phase de conception, au même titre que la description du flot normal d'activités. En effet, il s'agit de règles de gestion au sens large, c'est-à-dire d'exceptions indépendantes des futures plateformes de déploiement et d'implémentation. Elles font partie de la description du métier et doivent être prises en considération en tant que telles durant la phase de conception. Les caractéristiques que nous avons proposées pour repérer les portions de workflow correspondant à des flots d'activité exceptionnels sont plus des indices forts que de véritables critères. L'expérience du concepteur est donc essentielle au moment du choix. Pour l'aider dans sa démarche, nous avons proposé une classification des différentes exceptions rencontrées dans les workflows afin d'aider à leur identification et à leur modélisation. Au plus haut niveau, nous distinguons :

- les exceptions déclenchées par des événements temporels,
- les exceptions déclenchées par des événements liés aux changements sur les données et
- les exceptions déclenchées par des événements liés à un fonctionnement, d'un autre workflow par exemple, (comme les événements externes par exemple).

Notre approche de la modélisation des workflows a la particularité d'être basée sur l'utilisation de patrons afin d'aider les concepteurs à identifier et à modéliser les différents types d'exception susceptibles d'être nécessaires à la spécification du workflow. Pour cela nous avons proposé un modèle de représentation de patron d'exception (qui sera présenté dans la section 2.2.1), un mécanisme d'instanciation de ces patrons (qui sera détaillé dans la section 3.1.2) et une bibliothèque de patrons que nous avons établie en étudiant les besoins de nos partenaires en matière de représentation des exceptions dans les workflows (qui sera présentée dans la section 3.1.1). L'activité d'identification des exceptions est rendue plus facile grâce à la bibliothèque que nous proposons et dans laquelle les concepteurs peuvent sélectionner les patrons adéquats, puis les adapter au workflow en cours de conception. Nous distinguons deux dimensions dans cette adaptation :

- la dimension contextuelle : le concepteur peut réduire le contexte de l'exception sélectionnée en ajoutant de nouveaux prédicats dans la partie conditions de l'exception. De nouvelles actions peuvent également être ajoutées dans la partie actions.
- la dimension paramétrable : le concepteur spécialise les paramètres génériques de l'exception définie dans le patron et réduit ainsi la portée de l'exception pour l'adapter au workflow.

La phase de conception est complétée par l'étude des interactions entre le workflow et les systèmes d'information et les applications externes en précisant les modes d'interaction à l'aide de diagrammes d'interaction.

La phase de transformation qui est la dernière phase de notre méthodologie consiste à traduire la spécification du workflow obtenue à l'issue de la phase de conception dans le langage d'un système commercial de gestion de workflow choisi comme système cible. La plupart des outils commerciaux ont des caractéristiques semblables. Ils permettent de représenter les workflows comme composés de sous-processus et de tâches élémentaires dont le flot d'activité est modélisé sous forme de graphe. Ces outils permettent également la spécification des rôles auxquels les tâches sont affectées ainsi que les informations nécessaires à l'exécution de ces tâches. Un des aspects les plus critiques dans cette phase de transformation est la traduction des exceptions car il n'est généralement pas couvert ou l'est de façon limitée. Nous avons donc proposé des techniques de transformation afin d'intégrer les exceptions au flot normal d'activité. Cela a généralement pour conséquence de modifier la sémantique du flot normal d'activité et de le complexifier. Le traitement de certaines exceptions asynchrones, plus délicat, peut nécessiter dans certains cas le développement de programmes ad-hoc.

Le détail de cette démarche a été publié dans [7,19,8].

2.1.2 L'exemple du temps dans les documents multimédias

Un document multimédia interactif est constitué d'objets multimédias (images, sons, vidéos, etc) présentés selon un scénario préétabli. Durant la présentation du document (l'exécution du scénario), ces objets peuvent être transformés spatialement et/ou temporellement pour être présentés selon les spécifications données par l'auteur du document. Le document est dit interactif car il est possible dans sa description de spécifier des comportements qui sont des réponses à des événements déclenchés par l'utilisateur. Le fait que l'utilisateur puisse choisir d'intervenir durant la présentation donne lieu à plusieurs exécutions possibles d'un même scénario. En effet, selon que l'utilisateur intervient ou pas durant la présentation et selon le moment de ses interventions, le scénario s'exécute différemment. Nous parlons alors des différents plans d'exécution associés à un scénario donné.

Les objets d'un exemple de document multimédia sont schématisés dans la figure 2.3. L'exécution du scénario commence (événement `StartApp`) avec la présentation du texte `Title` immédiatement suivie par la présentation du clip audio `Intro`. Après 3 secondes, l'image `IMG1` est présentée. Ce scénario peut être interrompu par l'événement `KeyEsc` ou par l'événement généré quand la fin du clip audio est atteinte (`IntroStop`). L'événement `IntroStop` entraîne le démarrage d'un vidéoclip `Video1` et le texte `Texte1` est affiché. Après 6 secondes, le bouton `ExitBtn` est affiché. Cette partie de la présentation est interrompue par l'évènement `ExitEvent` généré lorsque l'utilisateur clique sur le bouton `ExitBtn`. Le clip audio `Audio1` est alors lancé, immédiatement suivi du texte `Texte1` qui est remplacé par le texte `Texte2` après 2 secondes. Après 6 secondes, le bouton `ExitBtn` est à nouveau affiché. L'application se termine quand l'événement `ExitBtn` est généré ou bien lorsque l'événement généré par l'horloge de l'application indique que la durée maximale de présentation a été atteinte. Les objets `Audio1` et `Texte2`, s'ils sont encore actifs, sont arrêtés. Nous pouvons déjà constater sur cet exemple très simple qu'il est difficile, pendant la conception du document, de se rendre compte si l'exécution sera conforme à ce qui vient d'être spécifié et si tous les objets seront bien visibles (i.e., exécutés de façon à être vus ou entendus par l'utilisateur). Nous pouvons également comprendre que plusieurs plans d'exécution sont possibles suivant que l'utilisateur décide d'intervenir ou pas (il peut par exemple appuyer sur la touche `KeyEsc` pendant que

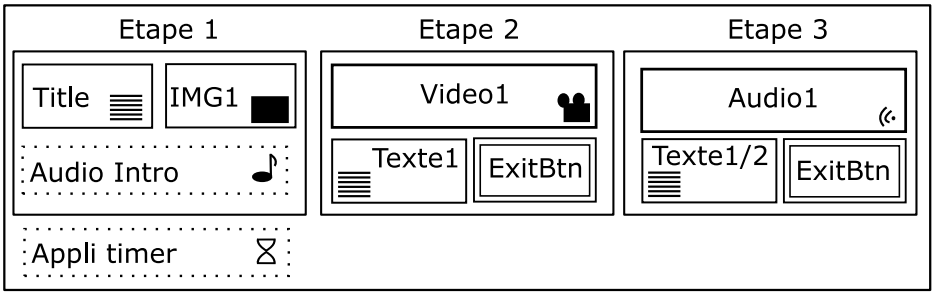

FIG. 2.3 – Un exemple d'objets d'un document multimédia interactif

l'image `IMG1` est affichée ou bien ne pas le faire) et suivant le moment où il décide d'intervenir. La possibilité de permettre plusieurs plans d'exécution pour un même scénario fait qu'il est encore plus difficile de se rendre compte si l'exécution sera conforme aux spécifications établies par l'auteur de la présentation. Dans le cadre du projet CHOROCHRONOS, notre but a donc été de proposer une technique d'aide à la vérification de la cohérence temporelle d'un tel document multimédia interactif.

Pour ce travail, nous sommes partis d'une description d'un scénario sous forme d'un ensemble d'unités fonctionnelles indépendantes que l'on appelle des *scenario tuples* [92]. Un *scenario tuple* (abrégé tuple par la suite) est caractérisé par les événements qui le déclenchent, les événements qui éventuellement l'interrompent, les actions de présentation associées et éventuellement des événements de synchronisation avec d'autres tuples du scénario. Chaque étape de la présentation est représentée par un tuple qui contient une liste d'actions à exécuter sur les objets impliqués dans l'étape de la présentation. Le tuple correspondant par exemple à la première étape de l'exemple de scénario illustré dans la figure 2.3 est présenté dans la figure 2.4.

```
TUPLE STAGE 1
Start Event          =   startApp
Stop Event           =   ANY(1 ;KeyEsc ;IntroStop)
Action List          =   Title> 0 Intro> 3 IMG1>
Start Synch Event    =   None
Stop Synch Event     =   None
```

FIG. 2.4 – Un exemple de tuple

Au moment de concevoir le document, l'auteur se concentre donc sur des aspects distincts de la présentation à différents instants et il est difficile dans ce contexte d'assurer la consistance de la présentation dans son ensemble, ce qui peut avoir pour conséquence de rendre une partie du document inaccessible ou invisible, voire de rendre la présentation inexécutable. Les documents sur lesquels nous avons travaillé sont dit interactifs car la séquence et les instants auxquels les interactions surviennent ne sont pas prédéfinis, ce qui signifie que plusieurs plans d'exécution sont possibles pour un même scénario. Pendant la conception du document, il

peut être difficile de prévoir tous ces plans d'exécution et aussi d'en assurer la consistance. Aussi avons nous proposé une technique [64,63] permettant d'isoler et de vérifier la consistance temporelle d'un document multimédia interactif au moment de sa conception.

Deux types de consistance temporelle sont distingués dans les scénarios multimédias : la consistance qualitative et la consistance quantitative. La consistance qualitative permet d'indiquer que les combinaisons des relations temporelles entre les objets multimédias sont correctes indépendamment de leur durée (un objet est bien arrêté après avoir été démarré par exemple). La consistance quantitative permet en plus d'indiquer que les durées de présentation des objets multimédias interactifs ne causent pas d'inconsistance (un objet n'est pas mis en pause alors que son exécution, ayant une durée limitée, est terminée).
Les événements auxquels nous nous sommes intéressés pour vérifier la cohérence des scénarios multimédias sont de plusieurs sortes : nous distinguons les événements générés par l'utilisateur (qui peut démarrer une vidéo, la mettre en pause ou l'arrêter par exemple), les événements générés par les objets multimédias (une vidéo est arrivée à la fin de son exécution) et ceux générés par le système. Ils peuvent être simples ou complexes et sont caractérisés par une signature spatio-temporelle.
Dans notre approche, la consistance temporelle d'un scénario est vérifiée en plusieurs niveaux : d'abord au niveau de chaque instance d'objet multimédia puis au niveau de chaque tuple et enfin au niveau du scénario (et de chaque plan d'exécution possible).

Vérification de la consistance temporelle au niveau d'un objet multimédia : Pour vérifier la consistance au niveau de l'objet multimédia, nous avons proposé d'utiliser un diagramme état-transition représentant les différents états possibles de l'objet et les différentes transitions possibles entre ces états. Ce diagramme permet de représenter toutes les combinaisons d'opérations sur les objets ainsi que toutes les contraintes de durée qui leur sont associées. Il permet de vérifier une consistance qualitative au niveau de l'objet. La figure 2.5 montre le diagramme état-transition pour un objet dépendant du temps (comme les objets `Intro`, `Audio1` et `Video1` dans l'exemple de la figure 2.3). Les intervalles de temps entre les actions de contrôle des objets sont également examinés.

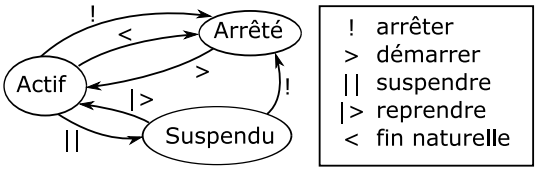

FIG. 2.5 – Diagramme état-transition pour les objets dépendant du temps

Vérification de la consistance temporelle au niveau d'un tuple : Pour vérifier la consistance au niveau d'un tuple, nous extrayons sa dimension temporelle. Elle est contenue dans les événements de début, les événements de fin et les événements générés par les actions associées à l'exécution du tuple ainsi que dans les durées des objets présents dans le tuple. Nous représentons cette dimension temporelle sous forme d'un réseau de contrainte (*conjunctions of bounds of differences*) sur lequel il est ensuite possible de raisonner. Les nœuds représentent les événements, les arcs représentent les successions possibles entre les événements et les inter-

valles attachés aux arcs représentent les distances temporelles minimales et maximales entre les événements. En raisonnant sur les contraintes spécifiées dans ces réseaux à l'aide de l'environnement STP [30], nous détectons les éventuelles inconsistances. Un exemple de réseau de contrainte temporelle correspondant au tuple décrit dans la figure 2.4 est donné dans la figure 2.6. Dans cet exemple, les contraintes sur les objets qui n'ont pas de durée, comme une image, apparaissent comme non bornées (c'est le cas de `IMG1` et de `Title`). Les durées des objets qui en possèdent une apparaissent comme des bornes maximales (c'est le cas de `Intro` dont la durée est de 47 secondes). L'enchaînement des actions décrites dans le tuple (`Title> 0 Intro> 3 IMG1>`) apparaît également sous forme de contraintes entre les événements de début (`Title>`, `Intro>` et `IMG1>`). Comme l'événement généré quand la fin du clip audio est atteinte (`IntroStop`) interrompt l'exécution du tuple (`Stop Event = ANY(1 ;KeyEsc ;IntroStop)`), des contraintes sont également ajoutées entre l'événement `Intro<` et les événements `Title<` et `IMG1<`.

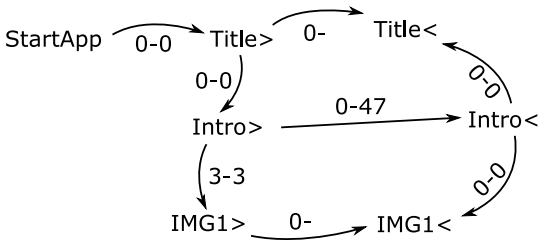

FIG. 2.6 – Réseau de contraintes temporelles du tuple `Stage 1`

Vérification de la consistance temporelle au niveau d'un scénario : Pour vérifier la consistance d'un scénario nous avons proposé dans un premier temps de construire un automate état-transition dans lequel :

- chaque état correspond à un ensemble de tuples en cours d'exécution à un instant t et
- chaque transition porte un ou plusieurs événements qui modifient l'ensemble des tuples en cours d'exécution à un instant t.

Cet automate contient un état *Idle* symbolisant le début et la fin de l'exécution normale du scénario et un état indéfini éventuellement atteint lorsque plus aucun tuple n'est en cours d'exécution et que l'événement de fin de la présentation n'a pas été détecté. Si le document est bien conçu du point de vue temporel, cet état ne devrait jamais être atteint. De cet automate sont ensuite déduits les différents plans d'exécution possibles. Pour chaque plan d'exécution possible, les réseaux de contrainte correspondant aux tuples impliqués dans le plan d'exécution sont alors composés les uns avec les autres. La composition se fait sur les noms des nœuds qui sont fusionnés s'ils sont identiques. Des informations supplémentaires n'ayant pas pu être indiquées dans les réseaux de contrainte lors de l'étape de vérification de la consistance temporelle au niveau des tuples, comme par exemple la durée d'exécution d'un objet qui peut être démarré dans un tuple et arrêté dans un autre, sont ajoutées à ce réseau de contrainte ainsi que les contraintes modélisant la séquence d'actions correspondant précisément au plan d'exécution (dans l'ordre où l'utilisateur génère ces actions).

À ce stade, nous prenons également en compte les éléments suivants.

- Les exécutions partielles éventuelles afin de prévoir les situations où certains évènements ne sont pas générés par l'utilisateur.
- Le fait que le moment auquel un utilisateur génère un événement peut conduire à des plans d'exécution différents : le fait que l'utilisateur demande l'arrêt d'une vidéo affichée à l'écran après ou avant la fin de l'exécution de cette vidéo par exemple donne lieu à des plans d'exécution différents qu'il faut tous analyser.
- La possibilité de faire intervenir plusieurs instances d'un même objet dans un même scénario: il faut alors donner un nom distinct à chaque instance pour que les contraintes temporelles soient correctement interprétées et vérifiées.
- Les événements complexes [92] comme une suite d'événements consécutifs, le fait qu'un événement ne survienne pas pendant un intervalle de temps donné ou au contraire le fait qu'il y survienne. Nous avons proposé des solutions pour traduire correctement ces événements complexes dans les réseaux de contraintes temporelles.

À ce stade à nouveau, en raisonnant sur les contraintes spécifiées dans les réseaux à l'aide de l'environnement STP [30], les éventuelles inconsistances sont détectées.

Cette technique de vérification que nous avons proposée et que nous venons de présenter permet de dériver pour chaque réseau de contrainte un réseau minimal représentant tous les chemins possibles à l'intérieur d'un plan d'exécution d'un scénario donné. Cela permet d'indiquer au concepteur de la présentation multimédia tous les plans d'exécution possibles pour le document qu'il est en train de concevoir. Cela peut être difficile à appréhender sans cette technique automatique que nous avons proposée. Cette dernière permet également de dériver des contraintes par exemple sur la durée maximale de présentation d'un objet si celle-ci était inconnue, afin d'éviter justement des situations inconsistantes. Cela permet au concepteur de la présentation multimédia de compléter la description du document pour la rendre plus robuste, encore une fois à l'aide d'informations difficiles à déduire sans la technique automatique que nous avons proposée. Enfin, notre technique permet au concepteur de la présentation multimédia de soumettre des requêtes pour vérifier des propriétés du document, en améliorer sa compréhension et faciliter sa construction. Un prototype permettant de mettre en œuvre cette technique de vérification a été développé.

2.1.3 L'approche Jecko

Dans la cadre de l'approche Jecko [52] développée en collaboration avec Amadeus S.A.S, notre contribution a porté sur l'adaptation des activités conceptuelles du processus de développement pour mieux prendre en compte des aspects critiques du développement de systèmes qu'il est nécessaire d'anticiper dès les premières phases du processus de développement. Dans le contexte du processus de développement et des projets de développement d'Amadeus S.A.S., nous avons isolé les critères suivants : la présence d'un système existant (*legacy system*), le fait de concevoir, développer et intégrer une base de données dans le système à développer, le fait de devoir concevoir, développer et intégrer une interface homme-machine graphique (abrégé IHM dans la suite) dans le système à développer et le fait de devoir concevoir et développer un système distribué.

Ces critères constituent une façon de spécifier un type de projet, selon la définition donnée dans [15]. Dans cet article, les auteurs, qui ancrent leur travail dans le domaine de l'ingénierie

des méthodes situationnelles, définissent la notion de situation d'un développement de système comme constituée d'un type de projet et d'un contexte de développement. Le type du projet est pour eux représenté par l'état du système à développer avant que le processus de développement ne commence et par l'état du système lorsque le développement est terminé. Ils expliquent comment ces caractéristiques influencent le choix de la méthode de développement à utiliser, c'est-à-dire, entre autres, la façon d'appréhender les activités conceptuelles du processus de développement. C'est donc à ce besoin d'identifier les caractéristiques du système à développer et d'adapter les activités conceptuelles du processus de développement que nous avons tenté de répondre. Nous nous sommes concentrés sur les critères de présence d'un système existant (*legacy system*) et de devoir concevoir, développer et intégrer une IHM dans le système à développer. Pour ces deux critères, nous avons d'une part proposé des extensions de la notation UML, sous forme de profils dédiés aux critères que nous avons isolés. D'autre part, nous avons proposé un ensemble de directives pour aider les concepteurs à mener à bien les activités conceptuelles du processus de développement en tenant compte de ces spécificités. Les profils proposés ont été définis en dehors de tout processus de développement particulier afin d'être utilisables dans n'importe quel environnement. Les directives ont été par la suite découpées en fragments pour être intégrées au processus de développement d'Amadeus S.A.S. comme nous le verrons dans la section 2.2.2. Dans la suite de cette section, je décris d'abord notre travail dans le cadre d'un développement à partir d'un existant. Puis je décrirai notre contribution pour le développement de système incluant une IHM.

Développer à partir d'un existant

Développer à partir d'un existant nécessite d'appréhender les activités conceptuelles du processus de développement de façon différente d'un développement entièrement nouveau (*from scratch*). Les spécificités de ce type de développement sont rarement prises en considération dans les phases d'analyse et de conception des démarches proposées dans la littérature. Pour pallier ce manque, nous avons d'abord étudié les particularités de ce type de développement puis nous avons défini un profil UML permettant de mieux les appréhender dans les modèles d'analyse et de conception et enfin nous avons proposé un ensemble de directives pour mettre en œuvre ce profil. Nous détaillons ces trois aspects de notre travail dans la suite.

Définition du développement à partir d'un existant

Nous avons considéré plusieurs façons d'appréhender un développement à partir d'un existant:

- continuer à utiliser le système existant, au moins en partie, et interfacer la nouvelle partie du système (les nouvelles fonctionnalités) avec elle. Nous parlons alors de *réutilisation de code*. La réutilisation de bibliothèque entre dans cette catégorie.
- développer un nouveau système en s'inspirant du fonctionnement de celui existant, de ses fonctionnalités, des données qu'il manipule et de l'IHM définie pour interagir avec lui. Ainsi, lorsque le développement d'un système est initié à partir d'un prototype ayant été réalisé au préalable, nous sommes dans ce cas de figure. Nous parlons alors de *réutilisation du domaine fonctionnel*.
- enfin, développer un nouveau système à la place de celui existant, pour le remplacer de façon transparente et donc en conservant rigoureusement toutes les interfaces du système existant avec d'autres systèmes informatiques existants. Nous parlons alors de *réutilisation d'interface*.

Pour chacune de ces façons de concevoir le développement d'un système à partir d'un existant, nous avons distingué plusieurs niveaux de réutilisation : *faible*, *moyenne* et *forte*. Une réutilisation de *code forte*, par exemple, correspond à l'utilisation d'un composant dont l'organisation n'est pas propriétaire et sur lequel aucune modification ne peut être réalisée. Cela peut aussi correspondre à l'utilisation d'une bibliothèque existante qui ne doit pas être modifiée. A l'opposé, le fait de réutiliser du code existant en faisant du « copier coller » correspond à une réutilisation de *code faible* puisque le code est largement modifié (seuls certains morceaux sont conservés). Une réutilisation du *domaine fonctionnel moyenne* signifie que le prototype qui a été réalisé sert de guide pour développer le nouveau système et qu'une grande partie de ses caractéristiques (fonctionnalités, données et IHM) est conservée mais que des adaptations sont également possibles. Une réutilisation d'*interface faible* signifie que les interfaces du système existant sont utilisées comme point de départ mais qu'elles peuvent être changées au cours de la modélisation et du développement du nouveau système. La table présentée dans la figure 2.7 résume tous les cas de figure possibles. En ce qui concerne la *réutilisation de code*, nous parlons de composant propriétaire pour indiquer que l'organisation a la possibilité de modifier le code du composant en question. Nous parlons au contraire de composant non propriétaire pour indiquer que l'organisation ne peut pas modifier le code du composant. En ce qui concerne la *réutilisation d'interface*, nous parlons de contraintes pour indiquer que d'autres systèmes sont susceptibles de s'interfacer avec le système en cours de développement. Plusieurs cas de figure peuvent se présenter suivant que l'on est, ou pas, en mesure de modifier les systèmes qui s'interfacent avec le système en cours de développement. S'il est possible de les modifier largement, alors ces systèmes contraignent peu le système en cours de développement. Au contraire, s'il est absolument impossible de les modifier, ces systèmes contraignent fortement le système en cours de développement.

Réutilisation	**de code**	**du domaine fonctionnel**	**d'interface**
Faible	Composant propriétaire, évolutions conséquentes souhaitées dans le code du composant	Prototype existant, nouveaux besoins nombreux	Application existante, peu de contraintes sur les interfaces existantes
Moyen	Composant propriétaire, quelques améliorations souhaitées	Prototype existant, quelques besoins nouveaux	Application existante, quelques contraintes sur les interfaces existantes
Fort	Composant non propriétaire, modification impossible ou non souhaitée	Prototype existant, pas de nouveau besoin	Application existante, fortes contraintes sur les interfaces existantes

FIG. 2.7 – Développement d'un système à partir d'un existant

Un profil UML dédié au développement à partir d'un existant Afin de bien appréhender les

spécificités du développement d'un système à partir d'un existant dans le cadre du processus de développement et de la notation UML utilisés chez Amadeus S.A.S., nous avons proposé un profil UML dédié. Le but de ce profil est de permettre une séparation claire dans les documents d'analyse et de conception entre ce qui est réutilisé tel quel, ce qui, dans le système existant, est modifié et ce qui est ajouté. Pour ce faire, nous avons proposé de raffiner la définition de certains méta-éléments de la notation UML afin de distinguer les parties nouvelles, les parties existantes et vouées à être modifiées et les parties réutilisées telles quelles. Nous les avons respectivement étiquetés *new*, *to-be-modified* et *reuse*. Nous avons appliqué cette classification aux éléments constituant les diagrammes de cas d'utilisation, les diagrammes de classes et les diagrammes de composants, qui couvrent les aspects structurels et comportementaux d'une modélisation et correspondent aux trois types de réutilisation que nous avons distingués (code, domaine fonctionnel et interface).

Durant les activités conceptuelles du processus de développement, il peut être nécessaire de documenter des points qui ne font pas partie du système à développer mais qui sont utiles pour aider à comprendre la façon dont ce nouveau système est conçu et doit fonctionner. Pour cela nous avons proposé des stéréotypes étiquetés *out-of-scope* dans les diagrammes de cas d'utilisation, de classes et pour les paquetages.

Enfin, en ce qui concerne les acteurs, nous avons proposé un ensemble de stéréotypes ayant pour but de bien distinguer les acteurs humains (*human*) des acteurs correspondant à des systèmes informatiques (*system*). Cela est important pour appréhender correctement les aspects réutilisation d'interface et réutilisation de code. Parmi les acteurs correspondant à des systèmes informatiques, nous avons distingué les nouveaux acteurs (*new*) de ceux dont le système à développer est dépendant (*dependent*). Parmi ces derniers, nous avons distingué ceux qui imposent une façon de communiquer au système à développer (*constraining*) de ceux avec qui l'interface de communication peut être redéfinie (*collaborative*). Le détail du profil est consultable en ligne sur les pages du Laboratoire I3S [4].

Des directives dédiées au développement à partir d'un existant

Nous avons proposé des directives pour mettre en œuvre le profil présenté ci-dessus durant les phases d'analyse et de conception du processus de développement. Sur la base des expériences de projet d'Amadeus S.A.S., nous avons rassemblé des façons de faire dans le but :

- d'identifier précisément ce qui doit être conservé du système existant durant la phase d'analyse,
- d'expliciter comment mettre en œuvre cette conservation durant la phase de conception.

Ces directives sont également consultables en ligne sur les pages du Laboratoire I3S [5]. Le profil et les directives ont donné lieu à publication dans [69].

Je vais maintenant présenter le second critère que nous avons étudié : le développement d'un système incluant une IHM.

[4] http://www.i3s.unice.fr/ mirbel/jecko/jecko-9.html.
[5] http://www.i3s.unice.fr/ mirbel/jecko/jecko-9.html.

Développer un système incluant une interface graphique

Il existe maintenant de nombreux outils et techniques pour faciliter le développement d'IHM. Notre contribution a porté plus précisément sur l'adaptation des activités clés des phases d'analyse et de conception afin d'améliorer la distinction entre les spécifications liées à l'IHM et les spécifications modélisant le fonctionnement des processus et le comportement des objets du domaine métier. Nous avons pensé qu'il était important de bien appréhender cet aspect du développement et de le faire de façon distincte de la modélisation des processus et des objets du domaine métier et cela dès les phases d'analyse et de conception. En effet, les processus et les objets du domaine métier évoluent relativement indépendamment de la façon dont ils sont montrés aux utilisateurs finaux du système à développer, au travers des IHM et réciproquement. De plus, les IHM évoluent généralement plus vite que les processus et les objets du domaine métier du fait de l'évolution rapide des technologies sur lesquelles elles sont basées. Enfin, plusieurs IHM peuvent être développées pour des processus et des objets du domaine métier identiques et inversement. Notre but a été de proposer une approche intégrée permettant de concilier la modélisation des IHM et des processus et des objets du domaine métier alors que les démarches existantes qui prennent en considération la modélisation des IHM supposent toujours que les processus et les objets du domaine métier ont déjà été spécifiés. Pour cela, en plus des directives que nous avons établies sur la base des expériences de projet chez Amadeus S.A.S., nous avons proposé un profil UML constitué de deux vues : la première est dédiée aux spécifications liées à l'IHM. La seconde est dédiée aux spécifications liées aux processus et aux objets du domaine métier. Ce profil permet d'aider les acteurs du processus de développement à mieux prendre en compte les spécificités de chacune de ces dimensions tout en mettant également en évidence les relations et les contraintes qui existent entre elles.

Un profil UML dédié au développement incluant une IHM

Nous distinguons la vue métier et la vue de l'IHM qui constituent à elles deux le modèle de l'application. Durant la phase d'analyse, les processus et les objets du domaine métier modélisés permettent une compréhension du cadre dans lequel le système doit s'intégrer. Ils ont généralement une portée supérieure à celle du système à développer proprement dite. Aussi, il est nécessaire d'extraire du modèle métier le modèle de l'application afin de se concentrer précisément sur le système qui devra être développé. Bien appréhender l'IHM à développer nécessite de bien modéliser le processus métier concerné, et plus particulièrement les séquences d'activités supportées par le système et réalisées en interaction avec les utilisateurs finaux du système. La description de ces processus constitue la vue métier du système à développer. Dans cette vue, nous nous sommes concentrés sur les aspects structurels et comportementaux des objets et des processus du domaine métier utiles à la modélisation des IHM. Pour cela nous avons proposé un profil permettant d'identifier clairement les cas d'utilisation spécifiant le contrôle du flot d'activité (*workflow*), de distinguer les acteurs humains (*Human*) des autres acteurs (*System*), de spécifier en détail les relations entre acteurs et cas d'utilisation afin de mettre en évidence les sections des processus métiers où une IHM est nécessaire. Nous avons également proposé des stéréotypes pour les diagrammes d'activité afin de systématiser la modélisation des points de décision de manière à ce que soit explicité le fait que la décision soit prise de façon automatique ou via une interaction avec l'utilisateur grâce à une IHM (*User-source*). L'utilisation d'un tel profil a pour but d'aider à anticiper la modélisation de l'IHM pendant la spécification des processus et des objets du domaine métier d'une part

et de favoriser la séparation entre la modélisation de l'IHM et la modélisation des processus et des objets du domaine métier dans les diagrammes de cas d'utilisation d'autre part.

Dans la vue de l'IHM, nous nous sommes attachés à proposer des façons d'expliciter les liens avec la vue métier notamment au moyen de stéréotypes de relations de dépendance entre cas d'utilisation permettant d'expliciter la dérivation des IHM à partir des processus et des objets du domaine métier (*For*). Nous avons également proposé des stéréotypes pour les diagrammes état-transition afin de permettre une représentation adéquate des différents éléments qui constituent l'IHM au moyen d'états. Dans cette vue, des stéréotypes sont également proposés pour clairement distinguer les événements générés par l'utilisateur de ceux générés par le système d'une part et les actions associées aux transitions dans les diagrammes état-transition qui sont déclenchées par l'utilisateur via l'IHM de celles dont le déclenchement est contrôlé par le système d'autre part.
La figure 2.8 montre un exemple de modélisation à l'aide du profil que nous avons proposé. Dans cet exemple très simple, un morceau de processus de commande via Internet est montré.

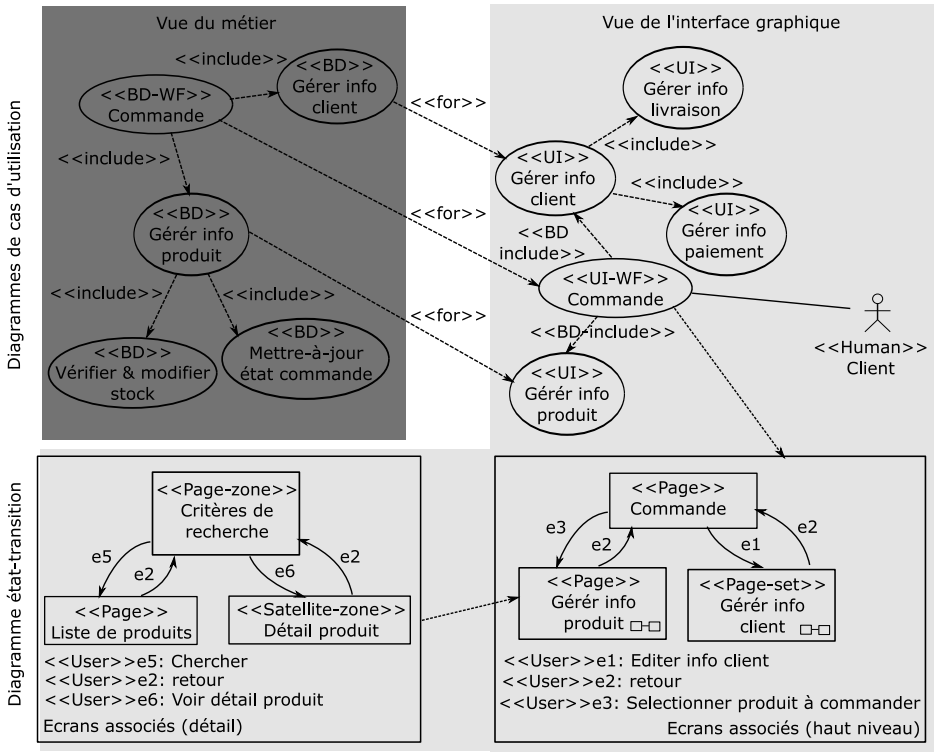

FIG. 2.8 – Exemple de spécification d'une application incluant une interface graphique

Nous distinguons une vue du métier et une vue de l'IHM. Dans la vue métier, le cas d'utilisation responsable du contrôle des activités (`Commande`) est mis en évidence à l'aide du stéréotype ≪ `BD-WF` ≫. Tous les cas d'utilisation de la vue sont étiquetés ≪ `BD` ≫ (pour *business domain*). Les relations de dépendance entre la vue du métier et la vue de l'IHM sont mises en évidence à l'aide du stéréotype ≪ `For` ≫. Dans la vue de l'IHM sont décrits les différents traitements supportés par l'IHM (`Commande`, `Gérer info client`, `Gérer info produit`, etc.). Remarquez que la façon de modéliser que nous proposons permet de dis-

tinguer des découpages entre les cas d'utilisation différents dans la vue métier et dans la vue de l'IHM. Dans cet exemple, le fonctionnement de l'IHM associée au cas d'utilisation qui contrôle le flot d'activité dans la vue de l'IHM est également décrit à l'aide de diagrammes d'état-transition qui permettent d'expliciter les différents écrans associés aux cas d'utilisation (`Critères de recherche`, `Liste des produits`, `Détail produit` d'une part, `Commande`, `Gérer info produit`, `Gérer info client` d'autre part) et la façon dont ils s'enchaînent. Les événements présents sur les transitions sont stéréotypés `User` pour indiquer que c'est l'utilisateur qui contrôle le passage d'un écran à l'autre. Différents stéréotypes sont utilisés pour distinguer les différents types de présentation souhaités dans l'IHM (≪ `Page` ≫, ≪ `Page-set` ≫, ≪ `Page-zone` ≫ et ≪ `Satellite-zone`).

En ce qui concerne la relation de dépendance ≪ `For` ≫, remarquez que dans l'exemple de la figure 2.8, elle permet de lier *un* cas d'utilisation de la vue métier avec *un* cas d'utilisation de la vue de l'IHM. Il faut éviter de lier *un* cas d'utilisation de la vue métier avec *plusieurs* cas d'utilisation de la vue de l'IHM afin de ne pas manquer de représenter les dépendances entre les cas d'utilisation à l'intérieur de la vue de l'IHM. En effet, ces dépendances sont le point de départ pour ensuite décrire l'enchaînement des écrans. Dans ce sens, remarquez que dans l'exemple de la figure 2.8, le cas d'utilisation de la vue métier ≪ `BD` ≫ `Gérer info client` est lié à un seul cas d'utilisation de la vue de l'IHM, ≪ `UI` ≫ `Gérer info client`. Et c'est ce dernier cas d'utilisation qui est lié aux cas d'utilisation ≪ `UI` ≫ `Gérer info paiement` et ≪ `UI` ≫ `Gérer info livraison` qui décrivent chacun une partie de l'IHM du cas d'utilisation métier ≪ `BD` ≫ `Gérer info client`. Il peut exister des cas de figure où *un* cas d'utilisation de la vue de l'IHM est lié à *plusieurs* cas d'utilisation de la vue métier. Cela signifie que l'IHM est suffisamment générique pour être utilisée dans plusieurs processus métiers.

Directives dédiées au développement incluant une interface graphique

Les connaissances relatives aux pratiques que nous avons rassemblées portent sur les deux vues que nous avons proposées (la vue métier et la vue de l'IHM). En ce qui concerne la vue métier, nos directives visent à distinguer les éléments du workflow ayant une influence sur la façon de modéliser l'IHM. En ce qui concerne la vue de l'IHM, les directives que nous avons proposées aident à expliciter les besoins en matière d'IHM, à décrire les séquences d'écran constituant une IHM et à expliciter le support aux règles de gestion des processus et des objets du domaine métier. Notre travail s'est concrétisé par la proposition d'un profil et de directives dédiées à la modélisation d'application incluant une IHM qui font partie de l'*UML usage recommandation and guidelines* de l'équipe en charge du processus de développement chez Amadeus S.A.S. Le profil est également consultable en ligne sur les pages du Laboratoire I3S [6] et a donné lieu à publication [68].

Ces deux approches (développer à partir d'un existant et développer un système incluant une IHM) peuvent bien sûr être couplées pour un même développement. C'est d'ailleurs également le sujet de l'approche Jecko dans laquelle nous avons proposé un environnement rassemblant des fragments dits primaires qui contiennent des directives valables quel que soit le système à développer et des fragments dits spécifiques qui sont fonction des caractéristiques du système à développer. Le fait de développer à partir d'un existant et le fait de développer un système incluant une IHM sont deux des caractéristiques prises en compte dans cet environnement qui

[6] http://www.i3s.unice.fr/ mirbel/jecko/jecko-10.html.

sera présenté plus en détail dans la section 2.2.2 de ce mémoire. Dans cet environnement, les fragments spécifiques font référence aux profils dédiés qui ont été présentés dans cette section.

2.1.4 Synthèse

Dans cette section, j'ai montré mes contributions à une meilleure modélisation des systèmes d'information en terme de modèle, de démarche et d'outil.

Nous avons d'une part proposé des **modèles** adéquats permettant de mieux représenter la dimension examinée tout en maintenant les liens et la cohérence avec les autres dimensions. Sur ce point, les contributions auxquelles j'ai participé se sont concrétisées par la proposition de modèles :

- un modèle de représentation de la dimension temporelle des documents multimédias dans le cadre du projet CHOROCHRONOS et
- des profils UML dédiés dans le cadre du projet JECKO.

Notre contribution à une meilleure modélisation des systèmes d'information s'est d'autre part concrétisée par la proposition de **démarches** pour prendre en compte pleinement les dimensions auxquelles nous nous sommes intéressés :

- une démarche pour la modélisation de workflows a été proposée dans le cadre du projet WIDE,
- une technique automatique de vérification de la consistance temporelle des documents multimédias en cours de création a été proposée dans le cadre du projet CHOROCHRONOS et
- des directives dédiées à la prise en compte de la présence d'une application existante et de la nécessité de concevoir et de développer une IHM ont été proposées dans le cadre du projet JECKO.

Enfin, des **outils** ont été également proposés pour aider les concepteurs à mieux appréhender ces dimensions particulières :

- un prototype a été développé dans le cadre de l'approche WIDE,
- un autre prototype a été développé dans le cadre du projet CHOROCHRONOS et
- les profils et les directives proposés dans le cadre du projet JECKO ont été placés sur l'intranet d'Amadeus S.A.S. et sont toujours utilisés.

> En matière de **modélisation des situations exceptionnelles dans les workflows**, nous avons proposé une approche innovante pour aider à la modélisation des situations exceptionnelles (ou règles de gestion au sens large) dans les workflows. La particularité de notre approche a résidé dans le fait de séparer la modélisation des situations exceptionnelles de celle du flot d'activité normal dans un workflow et de représenter ces situations sous forme de règles actives. Cette représentation facilite la traduction de cet aspect de la description du workflow lors d'une implantation du workflow dans un systme de gestion de workflow supporté par un système de gestion de base de données. Notre approche s'est également distinguée

par le fait qu'elle était la seule à proposer une démarche d'analyse et de conception de workflow, donnant des critères et des directives pour décider quand et comment un support automatisé pour un workflow est souhaitable.

En ce qui concerne la **modélisation du temps dans les documents multimédias**, notre approche a été une des premières à prendre en considération à la fois les aspects qualitatifs et quantitatifs lors de la vérification de la consistance d'une présentation multimédia et cela sur des objets et des interactions complexes, au niveau de chaque objet et au niveau global. La possibilité de dériver d'une part un réseau minimal de contrainte temporelle et d'autre part la durée maximale de présentation des objets ayant une durée d'exécution inconnue constituent également deux originalités de l'approche que nous avons proposée.

Enfin, pour permettre l'**adaptation des activités conceptuelles du processus de développement**, nous avons proposé deux profils originaux pour la notation UML ainsi que des directives idoines afin de rendre la notation UML mieux exploitable dans le contexte du processus de développement d'Amadeus S.A.S. Ces profils ont porté sur des aspects nouveaux et critiques à prendre en considération, dès les premières phases du processus de développement, durant les activités conceptuelles du processus : la présence d'une *legacy application* et la spécification d'une interface graphique de façon distincte et concurrente à la spécification du métier.

L'étude et la proposition de modèle, de démarche et d'outil pour la modélisation des systèmes d'information m'ont tout naturellement amenée à m'interroger sur la complexité grandissante des systèmes d'information et la nécessité de rendre les modélisations réutilisables d'un système d'information à l'autre. Cela n'a pas été appréhendé dans le cas de la présentation de documents multimédias car ces derniers constituent des modélisations très spécifiques et difficilement réutilisables d'un domaine à un autre. En revanche, les travaux sur la modélisation des situations exceptionnelles dans les workflows et sur les directives pour les activités conceptuelles du processus de développement ont été pensés en terme d'éléments réutilisables. C'est ce que je décris dans la section qui suit.

2.2 La modélisation pour le partage et la réutilisation

L'idée de réutilisation a été largement discutée dans le domaine de l'ingénierie logicielle depuis presque quarante ans. La majeure partie de l'activité de réutilisation logicielle a porté sur la réutilisation de code. Les patrons d'ingénierie (ou composants conceptuels), les composants architecturaux (ou composants logiciels) et métiers [46] permettent de capturer un savoir-faire de façon formalisée afin de permettre sa réutilisation. C'est vers la fin des années quatre-vingt dix que la réutilisation fait son apparition dans le domaine des systèmes d'infor-

mation. Comme dans le domaine de l'ingénierie logicielle, le besoin de capitaliser un savoir-faire et de le réutiliser dans des situations analogues se fait sentir. La complexité croissante des systèmes d'information et l'accélération du rythme de leur évolution imposent qu'ils soient conçus de manière modulaire, à l'aide de composants ou de patrons. La mise en œuvre du paradigme de réutilisation pose un certain nombre de problèmes différents selon qu'il s'agisse de l'activité de production des éléments réutilisables ou l'activité de développement d'application par réutilisation d'éléments. L'activité de production d'éléments réutilisables soulève les problèmes d'identification, de représentation, d'organisation et de qualification. L'activité de développement d'application par réutilisation concerne les problèmes de recherche, d'adaptation et d'intégration. Dans cette section, je vais présenter ma contribution en matière de production d'éléments réutilisables. Dans le chapitre 3, je présenterai plus précisément des aspects de mon travail liés à la recherche des éléments réutilisables, leur adaptation et leur intégration.

Un composant réutilisable [97] est défini comme étant n'importe quel élément de conception actuellement utilisé ou développé pour être utilisé dans plus d'un contexte. Tous les types d'élément générés pendant le processus de développement logiciel, comme les documents de capture des besoins, les façons de concevoir un système (modèles de processus) ou les connaissances sur un domaine métier par exemple, constituent des éléments réutilisables [97]. Une grande variété de composants, allant des patrons [35,32] aux composants métiers en passant par les COTS ont été proposés. Ces composants diffèrent par leur granularité, leur niveau d'abstraction et le type de connaissance qu'ils véhiculent. D'après [24], quatre familles de modèles de composant sont distinguées : les abstractions du domaine, les cas, les patrons et les frameworks.

- Les abstractions de domaine sont des structures génériques qui peuvent être réutilisées dans le processus d'ingénierie des besoins. Elles aident à caractériser le domaine du système d'information en cours de développement.
- Les cas sont le résultat de l'application du raisonnement à partir de cas au domaine de l'ingénierie des systèmes d'information. Ils permettent de résoudre de nouveaux problèmes en adaptant des solutions utilisées pour résoudre d'anciens problèmes. Les cas peuvent être vus comme des solutions réutilisables.
- Les patrons, (Gamma et al. [35]), permettent d'associer un problème et une solution. Ils sont dédiés à la réutilisation de micro-architecture (en terme de classe, de tâche, etc.) par opposition aux composants métiers ou architecturaux qui sont dédiés à des architectures plus conséquentes. Dans [29], des critères de classification ont été proposés pour les patrons d'ingénierie :
  - le type de connaissance : certaines approches, par exemple, portent sur la capitalisation de spécifications ou d'implantations, d'autres sur la capitalisation de produit ou de processus.
  - la couverture : les patrons généraux sont distingués des patrons de domaine.
  - la portée (ou niveau d'abstraction): certains patrons sont, par exemple, utilisés durant la phase d'analyse, d'autres durant la phase de conception, d'autres encore durant la phase d'implantation.

  Les informations contenues dans les patrons sont souvent exprimées en langage naturel excepté la partie solution qui est généralement formalisée. Ils sont organisés au travers de typologies de liens comme la spécialisation, l'utilisation ou l'alternative. Ces typologies

varient d'une approche à l'autre.
– Un *framework* est défini comme une conception réutilisable définissant un ensemble de classes collaborant pour réaliser une tâche particulière. Les frameworks verticaux définissent des architectures génériques pour toutes les applications d'un même domaine (banque, comptabilité, télécommunications, etc.). Les frameworks horizontaux proposent des fonctionnalités réutilisables indépendantes de tout domaine d'application.

Les auteurs de [24] concluent leur étude en indiquant notamment que les approches présentées ne prennent que très peu en compte la dimension dynamique de la connaissance permettant de capturer le contexte dans lequel le composant peut être réutilisé. Ils préconisent de proposer à l'avenir des composants capturant à la fois la connaissance effectivement réutilisable et la connaissance pour guider la réutilisation.

La figure 2.9 synthétise les différentes formes de réutilisation mises en évidence par les auteurs de [97]. Ils distinguent trois types de réutilisation : la réutilisation fonctionnelle, la

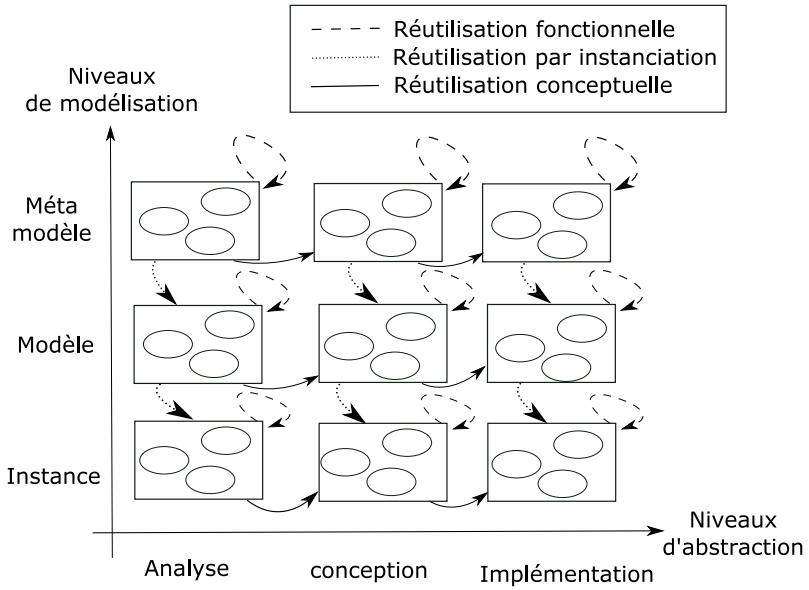

FIG. 2.9 – Les différentes formes de réutilisation [97]

réutilisation conceptuelle et l'instanciation. La réutilisation fonctionnelle porte sur des informations de même niveau de définition (métamodèle, modèle ou instance) et se situe dans des niveaux d'abstraction identiques (analyse, conception ou implémentation). Ces mêmes auteurs distinguent encore la réutilisation fonctionnelle par référence et la réutilisation fonctionnelle par copie. La première consiste à réutiliser une modélisation d'un domaine métier à un autre. Une entité *personne* modélisée sous forme de classe que l'on aurait spécialisée en deux sous-classes *étudiant* et *enseignant* est un exemple de réutilisation fonctionnelle qui s'opère donc indépendamment du domaine métier, entre conception de même niveau de modélisation et de même niveau d'abstraction. Une fonction de tri est un exemple de réutilisation fonctionnelle par copie. La réutilisation fonctionnelle est le type le plus courant de réutilisation décrite

dans la littérature selon [97]. Son exploitation efficace requiert une standardisation et une formalisation des métamodèles. La part la plus délicate du travail consiste à capturer dans ces métamodèles la sémantique des similarités et des différences entre la situation de réutilisation et les composants disponibles dans une bibliothèque de composants. La réutilisation d'instanciation, deuxième type de réutilisation isolé par les auteurs de [97], consiste, à partir d'un niveau de définition (métamodèle ou modèle) et de règles de transformation décrivant comment transformer un élément A en un élément B (de même niveau de définition), à en déduire les transformations permettant de passer d'une instance ou d'un modèle respectivement conçu à partir du modèle ou du métamodèle A à une instance ou un modèle respectivement conçu à partir du modèle ou du métamodèle B. Ce type de réutilisation, illustré dans la figure 2.10, nécessite un niveau méta clairement défini. Enfin, le dernier type de réutilisation, appelé

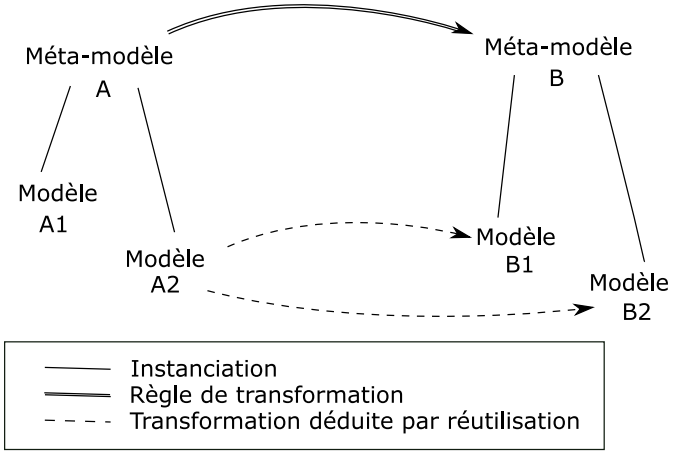

FIG. 2.10 – Réutilisation d'instanciation [97]

réutilisation conceptuelle, consiste à réutiliser les éléments d'un niveau d'abstraction dans le niveau d'abstraction inférieur. Cela peut être assimilé à de la traçabilité si les niveaux d'abstraction correspondent par exemple aux différentes phases du processus de développement. Un élément défini durant la phase d'analyse peut, par exemple, être conceptuellement réutilisé dans la phase de conception pour y être décrit plus en détail et selon des dimensions différentes.

Dans ce contexte, j'ai contribué à la problématique de la modélisation des informations pour le partage et la réutilisation au travers de deux projets de recherche auxquels j'ai participé :

- dans le contexte du projet WIDE, j'ai plus précisément travaillé sur les aspects modélisation des exceptions en participant à la proposition d'une spécification distincte du flot normal d'activité, comme je l'ai expliqué dans la première partie de ce chapitre. Dans cette proposition, nous avons également présenté une façon de spécifier les exceptions à l'aide de patron afin de tirer profit des expériences de modélisation antérieures et ainsi réutiliser l'expérience des concepteurs ayant déjà spécifié des processus métier.
- dans le contexte du projet JECKO, afin d'exploiter le mieux possible le résultat de notre travail sur l'isolation de caractéristiques des systèmes à développer susceptibles d'influencer la façon dont les activités conceptuelles sont appréhendées dans le processus de développement que nous avons présenté dans la première partie de ce chapitre, nous avons proposé de fractionner les directives couvrant un processus de développement,

en l'occurrence celui d'Amadeus S.A.S, en fragments permettant un partage plus facile des connaissances relatives aux pratiques et une meilleure qualification des directives au regard des caractéristiques que nous avons isolées.

Dans nos travaux sur les patrons d'exception et sur les fragments de méthode que je vais détailler dans la suite de cette section, nous avons particulièrement soigné l'aspect guidage de la réutilisation, comme le préconisent les auteurs de [24]. En effet, dans notre approche à base de patron, pour concevoir les exceptions attachées à une définition de workflow, nous avons proposé plusieurs catégorisations des exceptions (voir section 2.1.1). De plus, comme nous allons le voir dans la suite de cette section, nous avons inclus, dans notre définition d'un patron, une partie *exemples d'utilisation* afin d'aider le concepteur à comprendre comment exploiter les patrons proposés. Enfin, comme nous le verrons dans le chapitre 3 plus spécifiquement dédié au thème du partage et de la réutilisation, nous avons également proposé une bibliothèque d'exceptions génériques ainsi qu'une classification permettant de les organiser et aussi de guider leur réutilisation.

Dans nos travaux sur les fragments de méthode dans le cadre du projet Jecko, nous avons isolé et travaillé sur des critères déterminants dans la façon dont les activités conceptuelles du processus de développement sont appréhendées (comme je l'ai expliqué dans la section 2.1.3). Ces critères aident à comprendre comment et pourquoi réutiliser les fragments de méthode. Comme nous le verrons dans la suite de ce mémoire, notre travail sur des critères pertinents de qualification des fragments de méthode a été approfondi dans le cadre de l'approche Sesame pour laquelle nous avons proposé une structure dédiée à la représentation des critères pertinents de qualification des fragments de méthode afin de permettre aux ingénieurs méthode de diffuser cette information pour aider les ingénieurs d'application à mieux appréhender les activités conceptuelles du processus de développement. Cette structure sera présentée en détail dans le chapitre 3. Dans le chapitre 4, j'expliquerai comment cette structure est également utile aux ingénieurs d'application pour qualifier leurs besoins et aussi pour les aider à réutiliser la connaissance méthodologique dans leur contexte de travail.

Par rapport aux critères de classification proposés dans [24], le travail que nous avons effectué dans le cadre du projet Wide a porté sur des patrons généraux. Ces derniers permettent de capturer des spécifications de type produit (les exceptions dans le modèle de processus Wide). Ils ont une portée qui couvre les phases d'analyse et de conception. Le travail effectué dans le cadre du projet Jecko a porté sur des fragments de spécification de processus puisqu'ils ont pour objet de capturer un savoir-faire en matière de mise en œuvre des techniques d'analyse et de conception du processus de développement. Ils ne sont pas dédiés à un domaine d'activité particulier et traitent d'aspects généraux, même s'ils ont été développés pour une organisation particulière. Ils appartiennent donc également à la catégorie des patrons généraux. Leur portée couvre les phases d'analyse et de conception.

D'après la classification proposée dans [97], nos travaux sur les patrons d'exception et sur les fragments de méthode sont à classer dans la catégorie des réutilisations de type fonctionnelle. En ce qui concerne le niveau de modélisation, nos patrons d'exception se situent parmi les patrons de modèle (nous proposons un langage générique de spécification des règles actives) et nos fragments de méthode se situent parmi les patrons d'instance (nous proposons des savoir-faires réutilisables tels quels durant le processus de développement). Enfin, en ce qui

concerne le niveau d'abstraction, les patrons d'exception que nous avons proposés sont des patrons de conception. Nous avons proposé des fragments de méthode pour l'analyse et pour la conception.

Enfin, de façon plus générale, notre approche à base de patron pour la modélisation des exceptions dans les workflows se situe pleinement dans le domaine de l'ingénierie des systèmes par réutilisation. Notre travail sur les fragments de méthode, lui, est plus axé sur le partage que sur la réutilisation proprement dite de la connaissance relative aux pratiques. Notre but n'a pas été de proposer une approche pour construire et organiser des composants de méthode, c'est-à-dire des composants ayant pour vocation d'être intégrés au produit en cours de développement, en l'occurrence une méthode de développement de système d'information comme c'est le cas par exemple dans [72,38]. Le but de notre démarche a été de proposer des fragments d'expertise sur comment mettre en œuvre les méthodes de développement de système d'information de façon à ce que cette expertise soit partagée par plusieurs personnes et plusieurs projets au sein de l'organisation. Dans ce travail, les utilisateurs de la méthode réutilisent des connaissances relatives aux pratiques capitalisées lors de travaux antérieurs similaires pour mener à bien leurs activités de développement.

Nos deux contributions se rejoignent dans le fait qu'elles exploitent le principe d'abstraction, fondamental en matière de réutilisation. Ce principe consiste à distinguer explicitement dans un composant la partie réalisation, c'est-à-dire la connaissance réutilisable, de la partie spécification, c'est-à-dire la connaissance utile à sa réutilisation. En revanche, ces travaux se distinguent par le fait que la notion de généricité n'est présente que dans le travail sur les patrons d'exception. Il n'y a pas de généricité dans la démarche de partage de savoir-faire pour la mise en pratique des activités conceptuelles du processus de développement. En effet, dans notre travail sur les patrons d'exception, nous avons effectivement proposé un langage de définition d'exceptions génériques. En revanche, les fragments de méthode que nous avons proposés sont à utiliser tels quels. La qualification des contenus des fragments ou des patrons ainsi que les techniques nécessaires à la recherche de ces éléments partageables ou réutilisables sont des points traités dans les deux approches.

Dans ce chapitre, dédié à la structuration de l'information dans les systèmes d'information, je n'aborde que les aspects de représentation des éléments partageables ou réutilisables. Nous verrons dans le chapitre 3, dédié plus précisément au partage et à la réutilisation, comment nos travaux mettent en œuvre des éléments partageables ou réutilisables.

2.2.1 Un exemple de réutilisation : les patrons d'exception

Une originalité de la démarche de conception de workflow que nous avons proposée, WIRES (voir section 2.1.1 de ce chapitre), a été d'être basée sur des patrons pour modéliser les situations exceptionnelles.

Dans notre approche, un patron est une façon pour la personne en charge de la description de la définition d'un workflow de spécifier une situation exceptionnelle en réutilisant des expériences de conception antérieures. Un patron est constitué d'une ou plusieurs règles actives qui permettent de modéliser des exceptions au sens où nous l'entendons (cf section 2.1.1).

Nous avons défini un modèle de patron permettant la description générique de situations exceptionnelles [17,20]. Dans ce modèle, un patron se décompose en une partie *spécification* et une partie *exemples d'utilisation*. La partie spécification du patron regroupe le *nom* du patron, son *intention*, sa *classification* dans une base dédiée à ces patrons, sa *structure* (*template*), des *mots-clés* pour le qualifier, les noms d'*autres patrons* qui lui sont éventuellement liés (*related to*) et des *directives d'utilisation* (*guidelines*). La partie structure du patron peut avoir la forme soit d'exceptions génériques représentées à l'aide de règles actives génériques, soit de morceaux génériques de schéma de workflow (exprimés à l'aide du modèle de processus WIDE, voir section 2.1.1). La figure 2.11 montre le modèle d'un patron d'exception dans le projet WIDE. La généricité des règles permet de les réutiliser dans différentes appli-

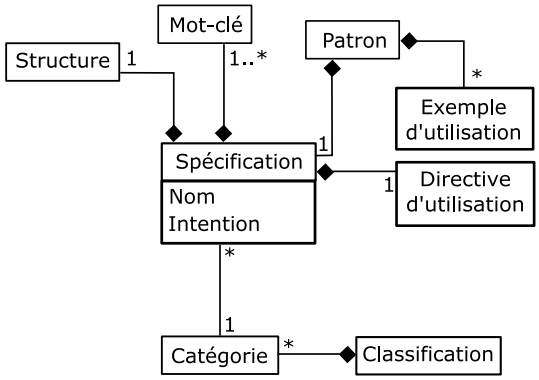

FIG. 2.11 – Le métamodèle de patron dans WIDE

cations et dans différents contextes. Nous avons défini un langage générique de définition de règles actives qui est une version étendue de Chimera-Exc. Ce langage permet de définir des instructions génériques, c'est-à-dire des instructions qui contiennent au moins un paramètre générique [17]. Les éléments d'une règle active qui peuvent être généralisés sont appelés des paramètres génériques. `<event>` est un exemple de paramètre générique qui couvre tous les types d'événements (externes, temporels, liés aux données). Dans une règle générique, la partie actions est seulement une suggestion afin de rester le plus possible indépendant du contexte. En effet, un patron d'exception a pour but d'aider le concepteur à identifier une situation exceptionnelle mais cela ne signifie pas forcément lui indiquer comment y répondre, les réactions à la détection d'une situation exceptionnelle étant souvent liées au contexte et n'étant pas alors généralisables. Un message d'alerte peut être une réaction satisfaisante en réponse à la levée d'une exception dans une application donnée quand une annulation de l'exécution de l'instance de workflow en cours peut être plus adéquate dans une autre application où ce même patron d'exception est utilisé. Une règle générique comprend des éléments prédéfinis, des éléments paramétrables et des éléments optionnels. Un exemple de patron contenant la définition d'une exception générique est donné dans la figure 2.12. Ce patron traite d'une situation dans laquelle une instance de workflow est démarrée suite à la terminaison d'une instance d'un autre workflow. Ce patron est classé dans la catégorie des patrons de continuation et plus particulièrement de terminaison (la classification dans sa globalité sera présentée dans la section 3.1.1). La règle générique proposée dans ce patron est activée lorsque l'événement de fin d'exécution d'une instance d'un workflow est détecté (`caseEnd`) et que certaines variables de l'instance de workflow en question contiennent les valeurs re-

Pattern Specification	
Name:	Termination
Intent:	This pattern allows the definition of the initiation of a case upon termination of another case, considering its status (in terms of WF variables).
Classification:	Continuation pattern | Termination
Template:	define trigger termination events caseEnd conditions case(C), occurred(caseEnd,C), <wfName>(W), W.caseId=C, W.<variable>=<value> actions startCase(<wfName>,<parameters>) end
Keywords:	case termination
Related to:	
Guideline:	The condition may contain clauses related to values of one or more variables in the terminated case.

FIG. 2.12 – Un exemple de patron

quises (`W.<variable>=<value>`), par exemple pour contrôler l'état dans lequel l'instance de workflow s'est terminée. L'action suggérée consiste à démarrer une instance de workflow (`startCase(<wfName>,<parameters>)`). Ce type de patron peut être utile par exemple pour lancer un traitement de compensation spécifique si une instance de workflow ne s'est pas terminée dans un état normal.

2.2.2 Un exemple de partage : les fragments de méthode

Je distingue différents courants de travaux en matière de spécification de démarche de développement de système d'information sous forme de fragments. Ils se différencient par les objectifs qu'ils poursuivent. Un premier courant d'approche [87] a eu pour but de documenter les démarches de développement de système d'information. Ces approches ne proposent ni support à la réutilisation de fragments de méthode d'une démarche à l'autre ni support à la personnalisation de ces démarches (ce thème sera discuté plus en détail dans le chapitre 4 de ce mémoire). La force de ces approches réside dans les efforts de formalisation des éléments qui constituent la démarche : tâches, activités, ressources, etc. Un deuxième courant de travaux [72,11] a regroupé des approches qui ont pour but d'aider à la construction de nouvelles démarches à partir de morceaux de démarches existantes. Ces travaux ont contribué à la naissance de l'ingénierie des méthodes situationnelles [42,72,38] qui vise à proposer des techniques et des outils pour construire des démarches flexibles adaptables aux spécificités de chaque projet, par opposition aux démarches universelles proposées jusque là dans la littérature. Ces approches cherchent, en plus de la formalisation du contenu des fragments, à proposer des opérateurs pour assembler les fragments les uns avec les autres et aussi des façons de comparer les fragments les uns aux autres. Elles sont dédiées aux ingénieurs méthode. Enfin, un troisième courant [66], plus récent, rassemble des approches dédiées aux ingénieurs d'application. Ces travaux cherchent à proposer des solutions pour permettre la réutilisation de fragment de méthode afin de faire bénéficier les ingénieurs méthode et les ingénieurs d'application des expériences acquises durant la résolution de problèmes antérieurs. C'est dans cette dernière famille d'approches que s'inscrivent nos travaux.

Une autre classification des fragments de méthode, illustrée dans la figure 2.13, a été proposée dans [44]. Elle est fonction du type de projet et du type de domaine d'application que les fragments couvrent. Les auteurs distinguent :

- les fragments à la volée (*on-the-fly*) qui correspondent à des solutions locales à des problèmes spécifiques,
- les fragments pratiques (*practices*) qui capturent des façons de faire qui peuvent être assemblées les unes avec les autres pour satisfaire les besoins d'un projet,
- les fragments de très haut niveau (*large disciplined domain*) qui proposent des directives qui peuvent potentiellement être utilisées quel que soit le projet et quel que soit le domaine d'application. Ces fragments sont mis en pratique par le biais d'un mécanisme de configuration afin de pouvoir être utilisés dans un contexte particulier de projet et de domaine d'application (e.g. RUP),
- les fragments spécialisés qui proposent des solutions dédiées à des types de projet bien particuliers, indépendamment du domaine d'application (e.g. XP).

Les fragments que nous avons proposés dans le cadre du projet Jecko correspondent à des fragments à la volée qui ont été développés pour répondre aux besoins spécifiques des projets développés au sein d'Amadeus S.A.S. : ils correspondent à des projets ayant un processus de développement relativement bien formalisé (au travers de phases définies dans l'organisation et de documents de spécification à rédiger à l'issue de chacune de ces phases) et sont dédiés à la spécification de systèmes d'information à l'aide de la notation UML.

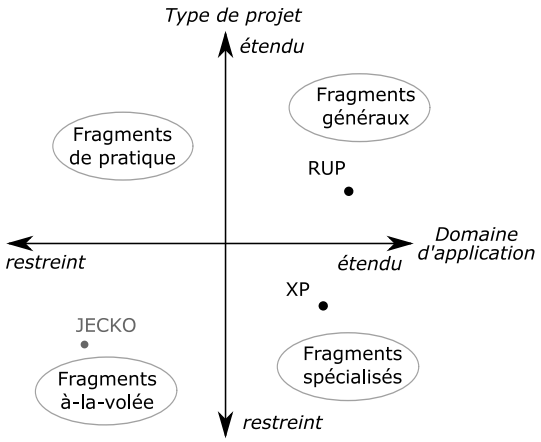

Fig. 2.13 – Les composants de méthode selon la classification de [44].

L'approche Jecko, qui a été développée en collaboration avec Amadeus S.A.S, a pour but d'aider les acteurs du processus de développement (analystes, concepteurs, architectes, chefs de projet, etc.) dans leur mise en œuvre des activités conceptuelles du processus de développement. La plupart du temps, les démarches proposées sont informelles et imprécises. Elles sont plutôt dédiées aux chefs de projet afin de les aider à contrôler le bon déroulement du processus de

développement mais elles sont généralement trop générales et trop floues pour aider les personnes qui doivent effectivement réaliser les différentes tâches qui y sont décrites. De même les environnements comme CMM (*Capability Maturity Model*)[43] décrivent des activités de très haut niveau qui nécessitent beaucoup d'efforts pour être implantables dans une organisation. Ces environnements sont plutôt dédiés encore une fois aux acteurs qui gèrent le processus de développement cette fois-ci afin de pouvoir contrôler sa certification. De plus, il y a un décalage entre la méthode vue au travers de ses concepts (*method-in-concept*) c'est-à-dire au travers de la façon dont elle est formalisée dans les manuels et la méthode en action (*method-in-action*) c'est-à-dire vue au travers de l'interprétation faite par les ingénieurs méthode [51]. L'expérience montre que cela a des effets négatifs et discrédite les méthodes de développement. Nous nous sommes donc attachés à proposer des directives concrètes et idoines afin d'aider les ingénieurs d'application à mieux prendre en considération les critères déterminants dans la façon d'appréhender les activités conceptuelles du processus de développement (voir section 2.1.3).

Une méthode peut être vue comme un ensemble de fragments faiblement couplés à différents niveaux de granularité. Un fragment de méthode est un morceau cohérent et autonome d'une méthode. Il permet la réalisation d'activités spécifiques au développement de systèmes d'information. Les fragments permettent de voir les méthodes de façon modulaire, ce qui favorise leur réutilisation et leur adaptation. Le découpage en fragment que nous avons proposé a été dicté par les critères déterminants dans la façon d'appréhender les activités conceptuelles du processus de développement.

Partant du principe qu'une méthode est constituée de deux aspects liés l'un à l'autre : le produit et le processus, plusieurs auteurs ont proposés deux types de fragments de méthode [42,11] : les fragments de processus et les fragments de produit. D'autres auteurs considèrent uniquement l'aspect processus [33], ce qui est notre cas dans l'approche Jecko. Enfin d'autres encore intègrent les deux aspects dans un même module appelé morceau de méthode (*method chunk*) ou bloc de méthode (*method block*) [79,73,70]. Un autre courant de travail a cherché à représenter les morceaux qui constituent une méthode sous forme de patron. Ces patrons conceptuels capturent les lois génériques qui gouvernent la construction de méthodes différentes mais ayant des ressemblances. Enfin, un courant récent porte sur la modélisation de fragments de méthode sous forme de services [77,38,78,31]. Ces fragments sont stockés dans des bibliothèques dédiées et représentent les briques de base pour construire des méthodes à la volée [65]. Dans notre approche, les fragments contiennent des directives qui sont exprimées en langage naturel et décrivent des façons de faire pour modéliser un système d'information à l'aide de la notation UML. Un fragment est spécifié au travers des éléments suivants :

- Un *nom* pour identifier le fragment,
- Une *situation* qui caractérise le type du projet (toujours selon la définition donnée dans [15]) auquel le fragment est dédié. Cette situation permet de positionner le fragment par rapport aux différents critères que nous avons présentés (la présence d'un système existant (*legacy system*), le fait de devoir concevoir, développer et intégrer une base de donnée dans le système à développer et le fait de devoir concevoir, développer et intégrer une IHM dans le système à développer),
- Une *intention* pour expliciter le but du fragment,
- Des *directives* pour expliquer comment procéder pour répondre à l'intention énoncée,
- Des *fragments associés* pour indiquer des fragments qu'il peut être éventuellement

intéressant d'examiner. Nous distinguons deux types de relations avec des fragments associés : la *complémentarité* et l'*alternative*. Un degré, dont la valeur varie entre 0 et 1, permet de quantifier l'association décrite entre les fragments. Dans le cas d'une complémentarité, cette valeur indique un *degré de nécessité*. Dans le cas d'une alternative, cette valeur indique un *degré de proximité* entre les fragments,

– Des *fragments incompatibles* pour indiquer qu'il est recommandé de ne pas utiliser certains fragments au sein de la même démarche que le fragment en cours d'examen. Là encore, un *degré d'incompatibilité* permet de quantifier la relation d'incompatibilité. Cette valeur varie entre 0 et 1, la valeur 0 étant exclue. En effet, une valeur nulle indique qu'il n'y a pas d'incompatibilité. Une valeur 1 indique que les fragments ne doivent jamais être utilisés ensemble. Une valeur de 0.5, par exemple, indique que le concepteur ayant proposé le fragment en cours de spécification pense que les deux fragments sont incompatibles dans la moitié des cas.

La figure 2.14 montre le modèle d'un fragment de méthode dans le projet Jecko. Le processus

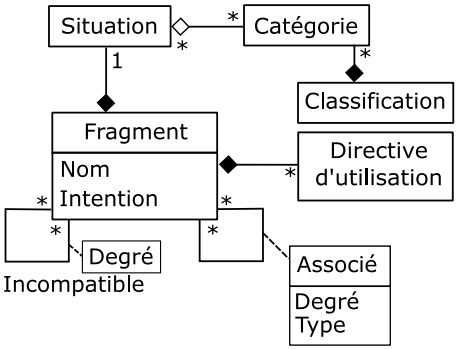

Fig. 2.14 – Le métamodèle de fragment dans Jecko

de développement d'Amadeus S.A.S. est découpé selon les phases suivantes : l'analyse des besoins, l'analyse des objets et du domaine métiers, l'architecture du système à développer, la spécification des fragments et la spécification interne. Afin d'inciter les personnes qui rédigent des fragments de méthode (ingénieurs méthode ou ingénieurs d'application) à donner des directives les plus concrètes possibles, nous avons découpé les phases du processus de développement d'Amadeus S.A.S. en étapes auxquelles des directives sont rattachées. Chaque phase du processus a été décomposée selon les mêmes étapes afin de proposer une démarche homogène et systématique. Nous distinguons une étape préliminaire, une étape principale, une étape de raffinement et une étape complémentaire.

– L'étape préliminaire rassemble des directives permettant de mieux appréhender les activités de l'étape principale de la phase. Cela consiste par exemple à synthétiser les documents disponibles sur le système existant dans la phase d'analyse des objets et du domaine métiers dans le cas d'un développement à partir d'un existant par exemple.
– L'étape principale rassemble des directives sur le cœur du travail de la phase considérée. Durant la phase d'analyse des objets et du domaine métiers par exemple, elle rassemble les activités nécessaires à la description des objets et du domaine métiers à l'aide des diagrammes de cas d'utilisation, de diagrammes de classe, etc.
– L'étape de raffinement consiste à réorganiser et raffiner les spécifications produites du-

rant l'étape principale. Dans le cas d'un développement incluant une IHM, cette étape, toujours dans la phase d'analyse des objets et du domaine métiers, propose par exemple des directives pour associer des dessins d'écrans aux traitements spécifiés durant l'étape principale.

– L'étape complémentaire rassemble des activités pour compléter les spécifications produites durant l'étape principale. Toujours dans le cas d'un développement incluant une IHM, durant la phase d'analyse des objets et du domaine métiers, cette étape propose par exemple des directives pour modéliser le traitement des erreurs de façon globale pour toute l'IHM afin que cela soit appréhendé de façon homogène.

Chaque fragment de méthode est attaché à une étape d'une phase. Quelles que soient les caractéristiques du système à développer, certaines activités conceptuelles sont incontournables. Citons par exemple la spécification des limites du système, la description des processus et des objets du domaine métiers, etc. Cependant, certaines de ces activités peuvent être menées de façon légèrement différentes suivant les caractéristiques du système à développer, c'est-à-dire du type de projet. De plus, des activités supplémentaires peuvent également être recommandées en fonction du type de projet. Dans notre approche nous distinguons donc des fragments dits primaires (*prime fragments*) qui contiennent des directives valables quel que soit le type du système à développer et des fragments dits spécifiques (*dedicated fragments*) qui sont fonction du type du projet, c'est-à-dire chacun dédié à une des trois caractéristiques du système à développer que nous avons isolées. Un exemple de fragment est donné dans la figure 2.15. Dans cet exemple, un fragment associé à l'étape principale (`core`) de la phase

Name	RequirementAnalysis::Core::UI-View
Situation	When dealing with GUI
Intention	Define User Interface (UI) specifications from Business Domain (BD) ones
Associated Fragments	RequirementAnalysis::Core::BD-View
Guideline	UML diagram: use-case diagrams * Deduct UI use-cases from <<BD>> and <<BD-WF>> use-cases. UI use-cases are derived from BD use-cases, at least as a starting point. A UI use-case is created for each BD use-case related to an actor through a <<UI>> association. Use-cases which are included or extend BD use-cases related to actor(s) through <<UI>> association may also lead to the creation of <<UI>> use-cases, except if the contrary is specified. * If dependency relationships exist among BD use-cases used to deduct <<UI>> use-cases, they also have to be taken into account through the UI view. Use <<BD-extend>> or <<BD-include>> dependency stereotypes to show the dependencies deducted from BD dependencies. * If a BD use-case has been stereotyped <<BD-WF>>, then the deducted UI use-case is stereotyped <<UI-WF>>. * Group the models related to the UI View in <<UI>> package(s). * When UI use-cases have been deducted from BD use-cases, show it explicitly via <<For>> dependencies.

FIG. 2.15 – Un fragment de méthode

d'analyse des besoins (`RequirementAnalysis`) dans le cadre du développement d'un système nécessitant de concevoir, développer et intégrer une IHM (`Situation : When dealing with GUI`) est décrit. Ce fragment a pour but de donner des indications sur comment spécifier

la vue de l'IHM à partir de la vue du métier. Dans la partie `Guideline` de la description de ce fragment, des indications sont données pour mener à bien cette tâche. Un fragment complémentaire lui est associé : le fragment `RequirementAnalysis::Core::BD-View`. L'utilisation conjointe de ces deux fragments (l'un dédié à l'analyse de l'IHM, l'autre dédié à l'analyse des objets et processus du domaine métier) permet de mettre en œuvre les directives que nous avons établies pour améliorer la modélisation d'un système incluant la conception, le développement et l'intégration d'une IHM (voir section 2.1.3).

L'ensemble du processus et des directives qui a été développé en collaboration avec Amadeus S.A.S. est décrit dans [67]. Notre contribution concernant la phase d'analyse des besoins a été publiée dans [52] ; celle concernant la phase d'analyse des objets et du domaine métiers a été publiée dans [66].

2.2.3 Synthèse

Dans cette section du document, j'ai présenté les contributions auxquelles j'ai participé en matière de réutilisation de descriptions génériques d'exception pour les workflows et en matière de partage de directives pour la mise en œuvre des activités conceptuelles du processus de développement. Ces travaux complètent mon travail sur la structuration et la séparation de certaines dimensions durant la modélisation des systèmes d'information afin d'en améliorer la qualité.

Le point fort de notre travail sur la modélisation des exceptions est d'avoir proposé une approche dans laquelle les exceptions sont représentées sous forme de règle active. Cela nous a permis d'améliorer la lisibilité des spécifications (comparé aux approches basées sur les réseaux de Pétri). Cela nous a également permis de proposer une implantation plus facile de cet aspect des workflows puisque le paradigme de règle active est commun au travail de conception et d'implantation. Enfin, cette façon de représenter les situations exceptionnelles nous a aussi permis de proposer un langage générique de description de cet aspect de la modélisation des workflows. Cela permet une conception à partir de patron pour que les concepteurs réutilisent la connaissance qui a été capitalisée sur cette dimension précisément durant des modélisations antérieures.

En ce qui concerne notre approche sur les fragments de méthode, les points forts de cette contribution ont été de proposer une approche pragmatique basée sur les expériences de projet d'Amadeus S.A.S ainsi que des critères de caractérisation des systèmes à développer afin de permettre une adaptation adéquate de la démarche de développement à la fois en terme de produit et de processus. En terme de produits, nous avons proposé des profils UML dédiés aux critères de caractérisation des systèmes à développer. En terme de processus, nous avons proposé un moyen de capitaliser des connaissances concrètes relatives aux pratiques sur les activités conceptuelles du processus de développement sous forme de fragments de méthode.

En matière de **patrons d'exception**, nous avons proposé un langage générique de définition de règles actives pour représenter les situations exceptionnelles dans les workflows. Ce mécanisme d'abstraction permet de représenter ces situations de façon indépendante des applications dans lesquelles elles sont utilisées. Notre contribution s'est distinguée des autres travaux dans le domaine des workflows qui, en terme de patrons, proposaient des bibliothèques de morceaux de processus. Chacun de ces fragments était défini pour un contexte d'application bien particulier. Au contraire, le modèle de patron que nous avons proposé permet la réutilisation des situations exceptionnelles décrites de façon générique dans des contextes applicatifs différents.

En terme de **fragment de méthode**, l'originalité de notre approche réside dans le fait que nous avons proposé des fragments dédiés aux ingénieurs d'application, alors que les fragments proposés dans la littérature étaient essentiellement dédiés aux ingénieurs méthode. Nous nous sommes pour cela efforcés de proposer des directives concrètes et idoines alors que les approches de la littérature mettaient l'accent sur la proposition de modèles de spécification de fragment. Enfin, notre modèle se démarque des autres modèles de fragment de la littérature du fait qu'il permet de nouveaux moyens de navigation entre les fragments de méthode prédéfinis et accessibles dans une bibliothèque dédiée. Ces fragments sont généralement liés les uns aux autres par les relations structurelles qui les unissent dans leur méthode d'appartenance. Nous avons proposé de spécifier, pour chaque fragment, des fragments complémentaires, des alternatives, des fragments associés et des fragments incompatibles supportant ainsi d'autres connexions entre éléments de méthode réutilisables.

Comme je viens de le détailler dans ce chapitre, mes travaux au sein des projets Chorochronos, Wide et Jecko m'ont permis de contribuer à améliorer la prise en compte de dimensions particulières durant l'activité de modélisation des systèmes d'information : le temps dans les documents interactifs multimédias (projet Chorochronos), les situations dites exceptionnelles dans la modélisation des workflows (projet Wide) et le type de projet dans le développement logiciel (projet Jecko). Dans les deux derniers projets, j'ai également contribué à améliorer la représentation de la connaissance réutilisable ainsi que de la connaissance utile à la réutilisation de cette dernière pour les éléments manipulés dans les dimensions que nous avons étudiées.

Je vais maintenant passer au deuxième volet de mon travail qui porte sur le partage de l'information. Je vais montrer comment, au delà de ma contribution à structurer les différentes dimensions nécessaires à la modélisation des systèmes d'information, j'ai également contribué à améliorer le *processus* de partage des informations.

Chapitre 3

Partage et réutilisation

Dans le chapitre 2, nous avons détaillé nos contributions en matière de production d'éléments réutilisables (ingénierie *pour* la réutilisation). Dans ce chapitre, nous montrons nos contributions en matière d'ingénierie *par* réutilisation.

Cinq phases constituent le processus de réutilisation [97,24], comme cela est montré dans la figure 3.1. La phase d'abstraction a pour but de standardiser les composants en les associant

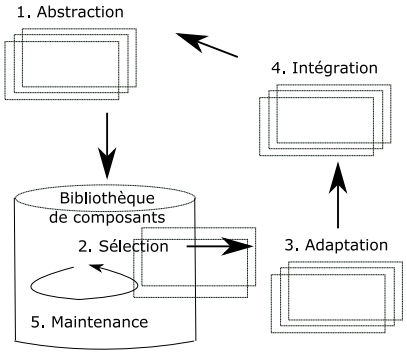

FIG. 3.1 – Le processus de réutilisation [97,24]

à des profils décrivant leur interface et leurs caractéristiques. La phase de sélection consiste à retrouver des éléments réutilisables parmi tous ceux stockés dans la bibliothèque d'éléments réutilisables. Généralement, un composant sélectionné comme candidat à la réutilisation ne satisfait pas toutes les exigences du contexte dans lequel il est réutilisé. Une phase d'adaptation est donc nécessaire pour adapter le composant à son nouveau contexte. La phase d'intégration consiste à intégrer un composant révisé pour les besoins de son contexte dans la nouvelle application. À ces activités principales s'ajoute l'activité de maintenance. En effet, au fur et à mesure que des composants sont stockés dans la bibliothèque, il est nécessaire de proposer de quoi maintenir cette bibliothèque de composants : certains composants qui ne sont plus utilisés doivent pouvoir être supprimés. Des moyens de réorganiser et de vérifier les composants conservés dans la bibliothèque de composants doivent aussi être proposés.

Développer par réutilisation signifie développer des systèmes d'information en réutilisant des composants appropriés. Les recherches dans ce domaine ont conduit à repenser le processus

de développement. Des outils pour réutiliser des composants de façon systématique durant le processus de développement ont été proposés. Dans [24], les auteurs distinguent trois types d'approches : les bibliothèques de composants, les systèmes de réutilisation et les environnements de développement par réutilisation. Ils décrivent les bibliothèques de composant comme des collections, parfois organisées, de composants réutilisables. Des techniques de recherche basées sur l'organisation des composants dans ces bibliothèques sont également proposées. Cependant, ces bibliothèques ne proposent généralement pas d'aide concernant les activités de sélection, d'adaptation et d'intégration. Toujours selon [24], les systèmes de réutilisation se concentrent sur les fonctionnalités de gestion des composants en plus d'une aide à la composition, l'adaptation et l'intégration de composant. Ce sont des outils qui sont souvent découplés du processus de développement. Enfin, les auteurs de [24] décrivent les environnements de développement par réutilisation comme des environnements qui se concentrent sur le processus de réutilisation en guidant la sélection des composants et leur adaptation. Dans ces systèmes, le processus de développement de système d'information est vu comme une activité de résolution de problème. Des outils ont été proposés pour supporter cette activité ainsi que des mécanismes pour assurer l'adéquation de la solution proposée.

Dans ce contexte, j'ai contribué à la problématique du développement par réutilisation au travers de deux des projets de recherche auxquels j'ai participé :

- Dans le contexte du projet WIDE, en plus de la démarche de modélisation et du langage de spécification de patron d'exception que nous avons présentés dans le chapitre 2, nous avons proposé une bibliothèque de patrons, des mécanismes de spécialisation et d'instanciation ainsi qu'un outil permettant de gérer la bibliothèque et d'en extraire des exceptions pour les intégrer à une spécification de workflow. Je détaillerai ces points dans la suite de ce chapitre.
- Le second projet, le projet SESAME, portait sur un environnement pour l'ingénierie des méthodes situationnelles combinant deux approches complémentaires : une approche de construction de méthode par assemblage de fragment et une approche de configuration de méthode au travers de la définition d'itinéraire. Dans le cadre de ce projet, nous avons proposé une structure évolutive et polymorphe permettant de décrire les critères pertinents de caractérisation des fragments de démarches et des situations de réutilisation ainsi qu'une mesure de similarité, permettant ainsi la réutilisation de fragment de démarche pour la construction de démarche de développement de systèmes d'information. C'est cet aspect que je développerai dans la suite de ce chapitre.

Vu sous l'angle de la classification proposée dans [24], le travail sur les exceptions dans les workflows que nous avons réalisé dans le cadre du projet WIDE s'apparente à une bibliothèque de composants. En effet, nous avons proposé une collection de patrons d'exception ainsi qu'une classification permettant de les organiser, comme nous le verrons dans la section 3.1.1. Cependant, nous avons également proposé d'une part des mécanismes d'adaptation et d'intégration de ces patrons pour qu'ils soient utilisables durant l'activité de conception d'un workflow, comme cela sera expliqué plus en détail dans la section 3.1.2. D'autre part, nous avons proposé une démarche, qui couvre les phases d'analyse, de conception et d'implantation d'un workflow, dans laquelle nous montrons comment tirer profit de la réutilisation tout au long du processus de développement d'un workflow, comme nous l'avons vu dans la section 2.1.1. Notre contribution en matière de fragment de méthode dans le projet SESAME s'apparente également à une bibliothèque de composants (proposée par J. Ralyté [72]). Dans ce projet,

notre effort a porté sur l'amélioration de la qualification des fragments de méthode à l'aide d'une structure dédiée. Nous avons également proposé des mesures afin d'aider à la sélection de fragment. Ces points seront détaillés dans les sections 3.2.1, 3.2.2 et 3.2.3. Dans la section 3.2.4, nous montrerons comment tirer profit d'un tel environnement pour la construction d'une méthode de développement de systèmes d'information par réutilisation.

3.1 Des patrons pour les exceptions dans les workflows

Dans le chapitre 2, j'ai détaillé notre proposition de patron pour modéliser les exceptions dans les workflows (voir section 2.2.1). J'ai présenté le contexte dans lequel ce travail avait été réalisé, le projet WIDE, et le modèle conceptuel duquel nous étions partis, le modèle WIDE. J'ai également présenté la démarche méthodologique globale que nous avons proposée et plus précisément la démarche de modélisation des exceptions. Cette démarche est basée sur l'utilisation de patrons définissant des situations génériques et récurrentes dans les spécifications de workflow. Nous avons également proposé une bibliothèque de patrons pour outiller le processus de réutilisation. Nous avons établit cette bibliothèque en étudiant les besoins de nos partenaires en matière d'exception dans les workflows. C'est cette bibliothèque que je présente dans la section qui suit. Les mécanismes de spécialisation et d'instanciation ainsi que le prototype qui a été développé seront présentés dans les sections qui suivent.

3.1.1 Une bibliothèque d'exceptions génériques

Le projet européen WIDE a rassemblé cinq partenaires pendant trois ans pour développer un système de gestion de workflow (WFMS) dont la spécificité était d'être supporté par un système de gestion de base de données. Parmi les cinq partenaires du projet, deux ont fourni des études de cas pour mener à bien le projet. L'hôpital de Manresa (Espagne) a fourni les spécifications d'une application de gestion des séjours de courte durée dans un service de chirurgie. Le département *Information Technology Center* du groupe financier ING (Pays-bas) a fourni les spécifications pour développer une nouvelle application de gestion des contrats d'assurance maladie. Nous nous sommes basés sur ces études de cas pour remplir la bibliothèque de patrons d'exception que nous avons proposée et concevoir une classification des différentes situations exceptionnelles pouvant être rencontrées. Cette classification n'est ni exhaustive ni figée. Elle est simplement le reflet de notre expérience en matière de modélisation d'exception. Dans la section 3.1.3 de ce chapitre, je présenterai l'outil qui a été développé afin de gérer la classification et le contenu d'une telle bibliothèque. Dans ce travail, nous avons essayé de proposer une classification et des exceptions suffisamment génériques pour être réutilisables dans d'autres domaines d'application.

Notre bibliothèque de patron d'exception permet la modélisation par l'exemple de certains éléments d'une spécification de workflow et cela à différents niveaux de granularité : au niveau de la spécification du workflow dans son ensemble ou au niveau d'une ou plusieurs des tâches qui constituent le workflow. Dans la bibliothèque, nous avons distingué trois catégories de patrons :

- les patrons élémentaires (*built-in patterns*),
- les patrons indépendants du domaine d'application (*generic tasks*) et
- les patrons propres à un domaine d'application (*generic application-oriented structure*).

Les **patrons élémentaires** servent à modéliser des exceptions simples et indépendantes d'un domaine d'application en particulier. Parmi les patrons élémentaires, nous distinguons :

- Les patrons dit *basiques* (*Basic patterns*) qui rassemblent essentiellement des alarmes. Ce type de patron permet par exemple de modéliser les actions à entreprendre lorsqu'une instance de workflow a une durée d'exécution qui dépasse sa durée maximale prévue. L'exception `SaleSpeed` donnée dans la figure 2.2 du chapitre 2 est un exemple d'instanciation d'un patron dit basique.
- Les patrons d'*autorisation*, d'*intégrité* et de *comptage* (*Authorization, integrity and count patterns*) permettent de modéliser des contraintes d'intégrité et de sécurité sur les données liées au workflow. Un exemple de ce type de patron est le patron `SeparationOfDuties` [21] qui permet de vérifier que les agents qui exécutent les tâches `t1` et `t2` qui se suivent de façon séquentielle dans la spécification d'un workflow ne sont pas les mêmes agents, afin de renforcer une politique de séparation des tâches préconisée par l'entreprise. Un autre exemple, le patron `BindingOfDuties` [21], consiste au contraire à vérifier que les tâches `t1` et `t2` sont exécutées par un même agent dans le cas où c'est une politique de confidentialité qui doit être renforcée.
- Les patrons liés au *démarrage*, aux *interactions* et à la *continuation* des instances de workflow (*Starting, interaction and continuation patterns*) permettent de modéliser les exceptions qui contrôlent l'exécution des instances de workflow. Les patrons de continuation permettent par exemple de modéliser les actions à entreprendre lorsqu'une tâche initiée pendant l'exécution d'une multitâche est encore active lorsque la multitâche est terminée. L'exemple de patron donné dans la figure 2.12 du chapitre 2 appartient à cette catégorie de patron.
- Les patrons concernant l'*organisation* et l'*évolution* dans les spécifications de workflow (*Organization and schema evolution patterns*) permettent de modéliser les exceptions contrôlant les changements dans l'organisation de l'entreprise et dans le workflow. Concernant les exceptions liées à l'organisation, nous avons, par exemple, proposé des patrons pour modéliser le départ d'un agent `agentGone` [17] ou pour détecter une supervision multiple d'une tâche `crossSupervision`[17]. Concernant l'évolution des spécifications de workflow nous avons proposé un patron dédié au changement de priorité d'une tâche `priorityChange`[17] .

Les **patrons indépendants d'un domaine d'application** rassemblent des activités récurrentes dans les spécifications de workflow et qui sont indépendantes des documents associés au workflow (et donc du domaine d'application). Dans cette catégorie, nous avons distingué :

- Les patrons d'*utilisation d'information* (*Information Use*) qui contrôlent l'utilisation correcte des informations durant l'exécution des activités. Dans cette catégorie, nous avons par exemple proposé le patron `Get&Check` [21] qui permet de récupérer des informations à partir d'un document ou d'un attribut du workflow et d'utiliser des exceptions appropriées pour vérifier qu'il n'y a pas d'éléments manquants parmi les informations récupérées.
- Les patrons de *rappel* (*Reminder*) qui permettent de lancer des opérations de rappel. Ce type de patron permet par exemple de modéliser les actions à entreprendre après qu'un nombre donné de relances aient été effectuées.

Les **patrons propres à un domaine d'application** permettent de modéliser des combinai-

sons d'activités (généralement combinées pour atteindre un but organisationnel) récurrentes et spécifiques à un ou plusieurs domaines d'application. Dans cette catégorie, nous avons distingué :

- Les patrons pour la *préparation de documents* (*Document preparation*) décrivent de façon générique le processus de préparation et de soummission d'un document pour approbation. Des patrons de vérification sur la complétude des documents sont également proposés dans cette catégorie.
- Les patrons pour la *réservation de services* (*Reservation service*) décrivent de façon générique le processus de réservation et d'obtention d'un service.
- Les patrons pour la *révision de document* (*Document revision*) qui décrivent de façon générique le processus de relecture d'un document par plusieurs personnes en parallèle, processus à l'issue duquel une décision est prise.

Le contenu de chaque catégorie est résumé dans les figures 3.2, 3.3 et 3.4. Notre classification a été publiée dans [18,20]. L'ensemble de ces patrons a été décrit dans [21].

BUILT-IN PATTERNS		
Category	**Subcategory**	**Patterns**
Basic exception	Temporal delay	temporalDelay
		taskTemporalDelay
		caseTemporaldelay
		durationExceeded
		delayAfterExternalEvent
	External event	externalEvent
		stopCase
Authorization	Agent authorization	separationOfDuties
		bindingOfDuties
		roleExamination
	Operation authorization	unhautorizedOperation
Integrity	Variable range	modificationOutOfRange
		creationOutOfRange
	Bad assigned variable	badAssignedValueAtStartTask
		badAssignedValueAtEndTask
		badAssignedValueAtStartCase
		badAssignedValueAtEndCase
	Invalid value	invalidValueAtStartTask
		invalidValueAtEndTask
		invalidValueAtStartCase
		invalidValueAtEndCase

FIG. 3.2 – Les patrons élémentaires (première partie)

3.1.2 Les mécanismes d'adaptation et d'instanciation de patrons

Afin de supporter les phases d'adaptation et de maintenance du processus de réutilisation, nous avons proposé trois moyens pour spécialiser une règle active : le raffinement, l'enrichis-

BUILT-IN PATTERNS		
Category	**Subcategory**	**Patterns**
Count		taskCount
Starting	Single entry point	singleEntryPoint
	Simulation of multiple entry points	multipleEntryPoint
Continuation	Wait	waitCondition
	Multitask	multitaskContinuation
	Join	joinContinuation
	Termination	termination
Interaction	Synchronous	synchronousDocumentExchange
	Asynchronous	asynchronousDocumentExchange
Organizational	Management	newAgent
		agentGone
		agentGone&Reassign
		changeResponsible
	Error detection	crossSupervision
Schema evolution		priorityChange

FIG. 3.3 – Les patrons élémentaires (suite)

GENERIC TASKS		
Category	**Subcategory**	**Patterns**
Information Use		get&Check
Reminder		reminder

FIG. 3.4 – Les patrons indépendants d'un domaine d'application

sement et l'instanciation.

- Le raffinement consiste en la réécriture de la règle en utilisant des termes génériques plus spécifiques. Une règle contenant par exemple dans sa partie événement un terme générique `<referenceEvent>` pourra être spécialisée en une règle dédiée plus particulièrement aux événements externes. Dans cette nouvelle règle, toujours générique, `<referenceEvent>` sera remplacé par `<externalEvent>`.
- L'enrichissement consiste à ajouter des conditions et/ou des actions dans une règle.
- Enfin, l'instanciation, consiste à remplacer certains éléments génériques de la règle par des termes qui ne le sont pas. Le terme `<class>` dans une règle donnée peut par exemple être remplacé par le terme non-générique `task` dans un patron plus spécifique et ne portant que sur les tâches.

La figure 3.6 illustre ces trois façons de spécialiser une règle active sur un exemple. Dans la partie événement du patron `temporalDelay`, l'événement générique `<referenceEvent>` est spécialisé *par raffinement* en un événement toujours générique mais plus spécifique `task-Start(<taskName>)`. Dans la partie condition, la classe de l'objet considéré dans la règle active, `<class>(C)`, est spécialisée *par instanciation* en la classe *tâche* (`task(T)`). Enfin, la partie action du patron est spécialisée *par enrichissement* : aucune action particulière n'était

GENERIC APPLICATION-ORIENTED STRUCTURES		
Category	**Subcategory**	**Patterns**
Document preparation		prepareRequest-Submit
Reservation service		reservation-Service
Document revision		multipleRevision

FIG. 3.5 – Les patrons propres à un domaine d'application

proposée dans le patron `temporalDelay`. Trois actions : `cancelTask`, `notify` et `startCase` sont suggérées dans le patron `taskTemporalDelay`. Si la réutilisation est faite dans le cadre

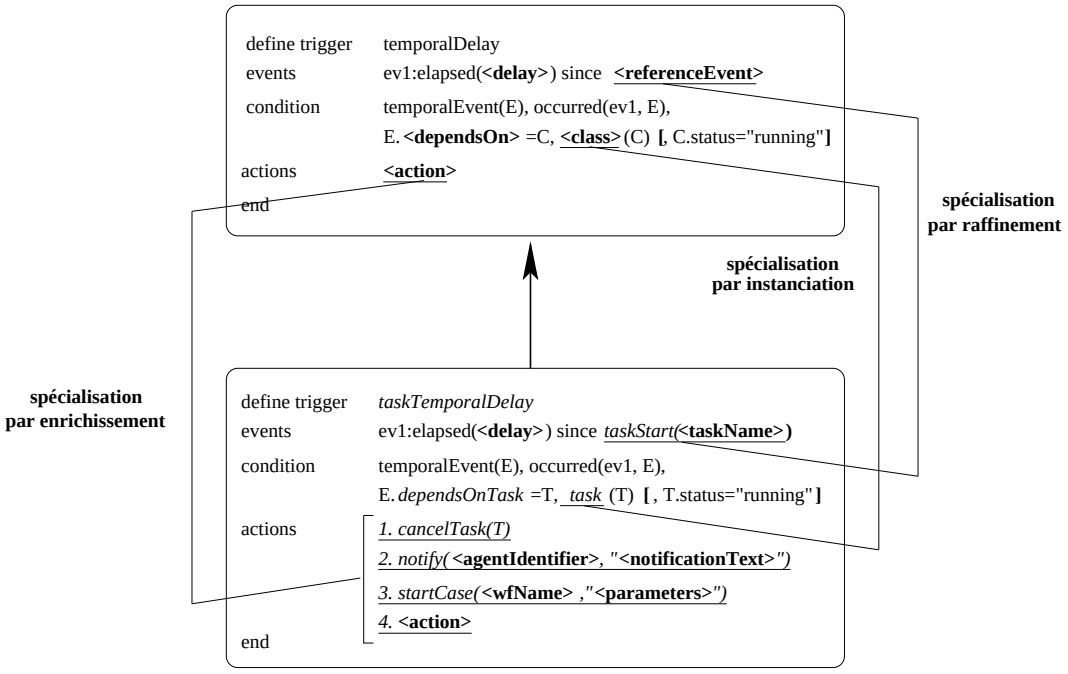

FIG. 3.6 – Spécialisation du patron `temporalDelay` en un patron `taskTemporalDelay`

de la phase d'adaptation du processus de réutilisation, c'est-à-dire durant la conception d'une spécification de workflow, alors après avoir été éventuellement spécialisé pour les besoins du workflow en cours de spécification, le patron doit être complètement instancié pour être effectivement intégré dans la spécification du workflow. Cette instanciation consiste à remplacer tous les termes génériques du patron par des valeurs. La figure 3.7 montre un exemple d'instanciation du patron `taskTemporalDelay`. Comme cela a été expliqué dans la section 2.2.1 du chapitre 2, dans une règle générique, la partie actions est seulement une suggestion. Dans l'exemple de la figure 3.7, seule l'action de notification a été conservée. Bien sûr, il est obligatoire de conserver au moins une action dans la partie actions de la règle : dans le cas contraire, la règle n'aurait aucun effet. La règle résultat de l'instanciation peut aussi contenir d'autres actions que celles suggérées dans le patron. Afin de préserver l'intégrité et la consistance des patrons lors de leur spécialisation, nous avons défini formellement notre mécanisme de spécialisation et les contraintes associées. Pour cela nous avons proposé une classification

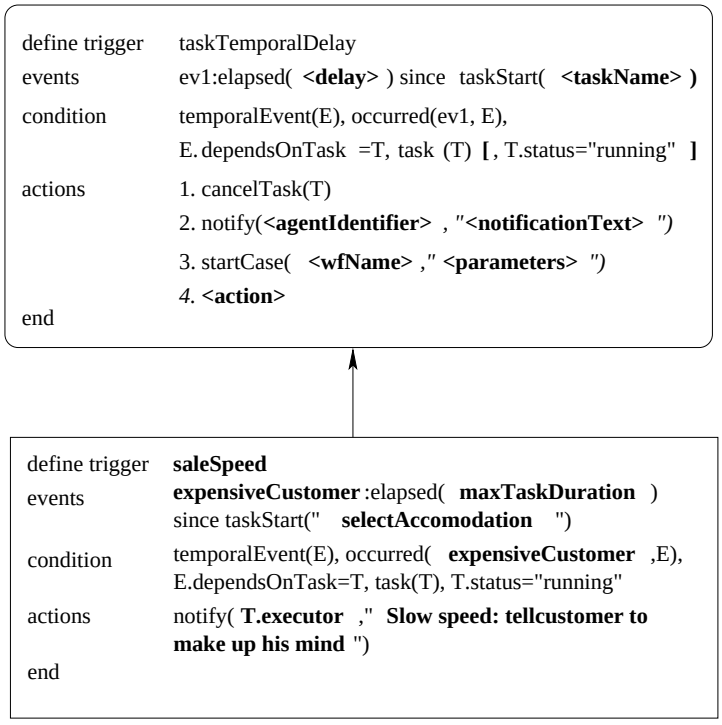

FIG. 3.7 – Instanciation du patron `taskTemporalDelay`

des instructions génériques dont les instructions les plus spécifiques correspondent à des instructions du langage Chimera-Exc (voir section 2.1.1 du chapitre 2). Nous avons représenté cette classification sous forme d'un graphe acyclique direct dont les nœuds représentent les instructions (génériques ou pas) et les arcs représentent les relations qui unissent les instructions (instanciation ou spécialisation). À l'aide de cette représentation, nous avons formalisé les règles de réécriture et les contraintes associées aux différents mode de spécialisation des patrons. Le détail de cette formalisation a été publié dans [17].

3.1.3 L'outil WERDE

Un outil a été développé pour permettre de gérer une bibliothèque de patrons telle que celle que nous venons de présenter. Cet outil permet d'insérer, de modifier, de supprimer des patrons de la bibliothèque. Il permet également d'ajouter ou de supprimer des catégories et des sous-catégories. Lors d'une suppression, l'outil contrôle qu'aucun patron ne fait référence à la catégorie ou à la sous-catégorie qui est supprimée. Il permet également de sélectionner un patron dans la bibliothèque, de le spécialiser et de l'instancier pour l'intégrer ensuite dans la spécification d'un workflow. Enfin, il permet de contrôler la consistance entre les différentes règles actives spécifiées dans une même spécification de workflow. En effet, une exception peut être considérée comme correcte quand elle est examinée de façon individuelle, mais les interactions mutuelles entre plusieurs exceptions peuvent mener à des comportements imprévus et non souhaités. Nous distinguons trois types d'interaction entre deux règles actives :

- les interactions de type action/événement,
- les interactions de type action/condition,

– les interactions de type action/action.

Le premier type d'interaction (action/événement) survient quand l'exécution de la partie action d'une règle déclenche une autre règle. Une action correspondant au démarrage d'une instance de workflow (`startCase`), par exemple, entraîne le déclenchement de toutes les règles ayant `startCase` comme événement de déclenchement. Le deuxième type d'interaction (action/condition) survient quand l'exécution de la partie action d'une règle affecte le résultat de l'évaluation d'une condition dans une autre règle (les liaisons établies). Une règle active peut par exemple inclure une action qui consiste à supprimer un agent, ce qui peut affecter le résultat de l'exécution des règles actives qui font référence à la classe *agent* dans leur partie condition. Enfin le dernier type d'interaction (action/action) survient quand deux règles actives modifient les mêmes objets dans leurs parties action. Si deux exceptions ayant des interactions sont exécutées de façon concurrente, le résultat de leur exécution dépend de l'ordre de leur exécution qui est non déterminé. L'outil que nous avons proposé permet de vérifier de façon statique qu'aucune interaction non souhaitée ne survient. Cela consiste à vérifier sur l'ensemble des exceptions associées à la spécification d'un workflow deux propriétés : la terminaison et la confluence. La terminaison est vérifiée lorsque les exceptions ne se déclenchent pas les unes les autres indéfiniment. Cette analyse est basée sur la construction de graphes décrivant les déclenchements et activations des exceptions afin d'y découvrir les cycles. La confluence nécessite que l'effet final après exécution des règles soit indépendant de l'ordre (non déterminé) dans lequel les règles de même priorité sont exécutées. Ces propriétés sont indécidables dans la majorité des cas. Nous nous sommes donc contentés dans cet outil de vérifier que les exceptions portent sur des classes différentes, ce qui assure qu'il n'y a pas d'interaction possible entre les exceptions [16].

3.2 Des fragments pour la construction de démarche

Dans le domaine de l'ingénierie des méthodes, il peut également être intéressant de réutiliser des expériences acquises pendant la résolution de problèmes antérieurs. Des démarches par réutilisation [72] ont été proposées pour adapter et intégrer des fragments de méthode dans le but de construire de nouvelles méthodes. Dans [45] est proposée une approche pour adapter les méthodes de développement en configurant un modèle standard de processus de développement. Lorsque les caractéristiques d'un projet correspondent à celles d'une situation de projet déjà rencontrée, alors la configuration de processus prédéfinie associée à cette situation est réutilisée. Cette approche a cherché à créer, à partir d'expériences de projet, des configurations de processus réutilisables. Dans le domaine de l'ingénierie des systèmes logiciels, une approche [43] basée sur le raisonnement à partir de règles pour raffiner au fur et à mesure le contexte d'un projet et proposer un ensemble de tâches et de sous-tâches adaptées aux spécificités du projet a été proposée. Dans ces approches, la réutilisation est appréhendée au travers du contexte du projet. Reconnaître un contexte déjà connu permet de réutiliser l'ensemble des activités (et les adaptations associées aux activités) qui correspondent à ce contexte.

Dans ce cadre, la façon dont sont exprimées la classification et la spécification des fragments peut affecter de façon significative le processus de réutilisation et par conséquent la méthode situationnelle résultat. Or peu de travaux ont porté sur des moyens idoines de catégoriser et

de rechercher des composants de méthode.

Dans le domaine du développement logiciel, domaine dans lequel le développement par réutilisation a été le plus étudié, l'auteur de [46] distingue trois catégories de systèmes de recherche de composants :

- les systèmes de recherche idoines dans lesquels l'utilisateur définit des requêtes auxquelles le système est censé répondre en renvoyant un ensemble de composants,
- les systèmes à base de routage dont le but est d'organiser les composants par rapport à des classes et
- les systèmes à base de filtrage qui trient un flux de composants et présente à l'utilisateur ceux qui correspondent à sa requête.

Notre travail s'inscrit dans les approches idoines. Parmi elles, les auteurs de [88,71,46] distinguent six familles de technique de recherche de composant, en plus des techniques classiques de recherche d'information : la classification externe, la classification structurée ou statique, la recherche comportementale ou dynamique, la recherche par navigation, les approches sémantiques et les approches mixtes.
La famille des techniques de classification externe [88,46] rassemble les approches à base de mot-clé, les approches par langage naturel et les approches par facettes. Les premières ont montré leur limite quand le nombre de mots-clés devient important. En effet, il est alors difficile de gérer la consistance de la classification. Les secondes sont basées sur l'analyse de descriptions textuelles en langage naturel des composants. Selon les auteurs de [46], la pertinence des informations extraites dépend donc de la précision avec laquelle les composants sont décrits. Elles sont délicates à mettre en œuvre et limitées à des bibliothèques spécifiques dans des domaines restreints. Enfin les dernières sont basées sur la description de chaque composant à l'aide de facettes (nom, date de création, taille, type de composant, domaine, producteur, etc) permettant d'effectuer ensuite une recherche. Selon les auteurs de [46], cette technique nécessite de bien choisir les facettes, de bien indexer les termes et de bien définir l'espace des termes.
La famille des techniques de classification structurées (ou statiques) [88,46] rassemble les techniques d'appariement de signatures ou de spécifications. Ces techniques sont plutôt vouées à des composants d'implantation. En effet, ces types de composant ont des signatures précises qui donnent suffisamment d'information pour supporter un mécanisme de recherche, même si la spécification transmise dans une signature est plus syntaxique que sémantique. Dans le cas des fragments de méthode, il est difficile, dans la signature du composant, de spécifier toutes les informations qui pourront être utiles à la recherche de composant et de baser le processus de recherche sur l'appariement de ces signatures. La recherche de fragments de méthode nécessite l'exploitation de spécifications sémantiques.

Les approches rassemblées dans la famille des techniques de recherche comportementale (ou dynamique) [46] exploitent l'aspect dynamique des composants. Elles sont consacrées à des composants exécutables et semblent là encore difficiles à appliquer sur des composants dédiés à la représentation d'un savoir faire comme c'est notre cas.

Les approches par navigation exploitent les relations implicites ou explicites qui peuvent exister entre les composants. La faiblesse des approches par navigation réside dans le fait que c'est l'utilisateur qui pilote le processus de recherche. Elles sont plutôt préconisées en complément d'autres approches pour raffiner le résultat d'une recherche [46].

Les approches sémantiques [88,71] utilisent des représentations supplémentaires pour améliorer la formulation de la requête et ainsi obtenir un meilleur résultat. Parmi ces représentations, citons celles centrées sur le domaine d'application et reposant généralement sur des ontologies et celles centrées sur les modèles utilisateur qui représentent de façon explicite des connaissances possédées sur un utilisateur (compétences, buts, contexte de travail). Certaines de ces approches supportent la reformulation de requêtes afin de prendre en considération les cas de synonymie, d'hyponomie et d'hypernomie[1]. C'est l'augmentation de requête [88], c'est-à-dire la possibilité de modifier la requête initiale avec des informations du domaine appropriées, afin d'améliorer les performances du système de recherche.

Enfin, les approches mixtes combinent plusieurs de ces techniques.

Dans ce contexte, notre approche s'inscrit parmi les approches sémantiques qui nous semblent les plus adaptées (avec les techniques classiques de recherche d'information et les techniques de classification externes) à l'exploitation de composants de méthode. La plupart des approches proposées dans le domaine de l'ingénierie des méthodes s'apparentent à des classifications par navigation. On trouve également une approche sémantique présentée dans [38]. Cette approche est basée sur l'utilisation de quatres ontologies respectivement dédiées aux buts, aux acteurs, aux processus et aux produits. Elles permettent la composition de fragments de méthode considérés comme des services. Notre travail s'inscrit dans la famille des approches sémantiques.

Dans le domaine de l'ingénierie des méthodes, le développement par réutilisation consiste à construire des processus de développement à partir de fragments de démarche existante qu'il s'agit de combiner pour mieux répondre aux spécificités du projet ou de l'organisation [42,41,11,72]. Cela permet également de faire évoluer les démarches. Afin d'aider les ingénieurs méthode à bien sélectionner des fragments pour construire ou faire évoluer une démarche de développement, des travaux ont cherché à enrichir la description des fragments [38]. Nous avons proposé une structure permettant de rassembler des critères pertinents : le *Reuse Frame*. Comme l'un des buts de notre travail était d'aider les ingénieurs méthode dans leur travail d'adaptation des démarches de développement aux spécificités d'un projet, c'est un moyen de représenter ces spécificités que nous avons cherché à proposer au travers de cette structure. Comme un autre de nos buts était d'aider les ingénieurs méthode à accéder au savoir-faire méthodologique capitalisé dans les fragments de méthode, ce sont aussi des éléments de qualification de ces fragments qui sont rassemblés dans la structure que nous avons proposée. Cette structure polymorphe permet donc de représenter ces critères de façon plus ou moins spécifique et prend ainsi en compte les problèmes d'hyperonymie et d'hyponymie que l'on peut rencontrer lorsque plusieurs acteurs sont amenés à définir ou représenter une même information. La structure que nous avons proposée est évolutive. En effet, même si les démarches traditionnelles de développement de système d'information se disent universelles, il n'est pas possible *a priori* de prévoir toutes les situations de développement. Aussi, avons-nous proposé une structure qui permet la définition de critère pour caractériser les situations de réutilisation et les fragments de démarche mais sans les définir tous *a priori*. Cette struc-

[1]Les hyperonymes et les hyponymes indiquent des niveaux d'une hiérarchie désignés par le concept plus général ou le concept moins général. `Personne` est un hyperonyme de `employé`.

ture, extérieure et indépendante de la bibliothèque de composants, nous permet également de proposer un moyen original d'organiser des fragments de méthode qui vient s'ajouter à l'organisation classique basée sur les relations explicites entre composants. Dans la littérature sur les fragments de démarche, certaines approches permettent d'organiser des fragments selon les phases du processus auxquelles ils appartiennent (voir l'approche JECKO, section 2.2.2) et d'autres [72] au travers des relations qui les unissent structurellement les uns aux autres telles que la spécialisation ou la composition. La recherche de fragment se fait alors par mot-clés sur l'intention ou le but du fragment. Une approche basée sur l'exploitation d'ontologies a été proposée dans [38]. Cette approche propose d'exploiter quatre ontologies pour spécifier les buts, les acteurs, les processus et les produits associés à un composant de méthode considéré comme un service. Dans ce travail la notion de contexte exploitée est très spécifique et fait référence uniquement aux « quatre P » du processus unifié UP. Nous avons travaillé sur un cadre fédérateur aux différents facteurs de contextualisation que nous avons trouvés dans la littérature afin de proposer des moyens de compléter les approches d'assemblage de composants de méthode existantes. Nous proposons une recherche sur le contexte de réutilisation du fragment, en plus de son intention (ou but), de sa place dans le processus de développement, des éléments de produit et de processus qu'il exploite. Enfin, nous avons également travaillé sur la notion de similarité (entre composants de méthode, entre un composant et une requête) et proposé une mesure originale, exploitant la structure du Reuse Frame, afin de proposer des composants qui se rapprochent les uns des autres ou qui se rapprochent de la requête. Notre travail se distingue des travaux de la littérature sur cet aspect [72,38] par le fait qu'il exploite les relations du Reuse Frame, notamment celles de généricité et qu'il permet de quantifier la proximité entre les éléments comparés.

Dans la suite de cette section, je vais dans un premier temps détailler la structure polymorphe et évolutive que nous avons proposée pour qualifier les fragments de démarche et les contextes de réutilisation (voir section 3.2.1) ainsi que les mesures que nous avons proposées pour les apparier. Puis je décrirai l'approche de construction de méthode dans laquelle s'est inscrit notre travail sur le Reuse Frame dans le cadre du projet SESAME.

3.2.1 Le Reuse Frame

Nous avons cherché à proposer une solution pour permettre le partage et la réutilisation de fragments de méthode. Cela nécessite de caractériser l'information encapsulée dans les fragments afin de permettre de la conserver puis de la retrouver.

Plusieurs approches de la notion de contexte ont été proposées dans la littérature : dans [86] des facteurs de contingence qui influencent le déroulement d'un projet de développement de système d'information ont été proposés. Nous avons proposé dans nos travaux des facteurs techniques ayant également une influence sur le déroulement des activités conceptuelles du processus de développement (voir section 2.1.3).

Pour permettre une représentation homogène de tous ces critères, nous avons proposé une structure, le Reuse Frame, qui permet de représenter les connaissances critiques dans le domaine de l'ingénierie des systèmes d'information. L'idée est de permettre une meilleure définition des besoins méthodologiques lors de la recherche de fragments de méthode et donc également lors de la spécification du contexte de réutilisation des fragments. Cela repose sur

la caractérisation des systèmes d'information selon trois dimensions [23] :

- la dimension organisationnelle, qui caractérise le contexte organisationnel dans lequel le développement a lieu,
- la dimension technique, qui caractérise le produit qui va être développé et
- la dimension humaine qui caractérise les personnes qui vont mener à bien le développement.

La structure du Reuse Frame permet une définition plus ou moins précise des éléments de classification appelés *critères*. Les trois dimensions (organisationnelle, technique et humaine) sont en fait modélisées sous forme d'un arbre de critères successivement raffinés. Chaque organisation souhaitant utiliser un Reuse Frame pour organiser ses fragments de méthode peut le remplir avec ses propres critères. En plus de la structure du Reuse Frame, nous avons proposé un contenu. Ce contenu a été élaboré en intégrant les différents travaux qui ont été réalisés sur des facteurs de contingence [86], humains et techniques (voir section 2.1.3). Ce contenu est suffisamment large pour être approprié à n'importe quelle organisation. Mais il peut aussi être remplacé, complété ou raffiné par un contenu plus approprié à l'organisation dans laquelle le Reuse Frame est déployé. Nous avons aussi défini une notion de contexte de réutilisation et de situation de réutilisation afin de qualifier respectivement les fragments de méthode et les situations de réutilisation des fragments et cela à l'aide des critères définis dans le Reuse Frame. Nous avons également proposé une mesure de similarité afin de pouvoir comparer des contextes de réutilisation entre eux et des situations de réutilisation avec des contextes de réutilisation. Notre travail sur le Reuse Frame a été publié dans [58,59,61]. Nous en donnons ici une synthèse.

Dans un premier temps je présenterai la structure du Reuse Frame, puis le contenu que nous avons proposé. Ensuite des notions de contexte de réutilisation et de situation de réutilisation seront décrites. Enfin, je détaillerai la mesure de similarité que nous avons utilisée.

La structure du Reuse Frame

L'objet du Reuse Frame est de capturer de la connaissance en matière d'ingénierie des systèmes d'information afin de permettre d'une part de qualifier les fragments de méthode dédiés au développement de systèmes d'information conservés dans une bibliothèque de fragment adéquate et, d'autre part, de qualifier les situations dans lesquelles ils sont susceptibles d'être réutilisés. Dans [15], les auteurs définissent la notion de situation de développement comme constituée d'un type de projet et d'un contexte de développement. Par type de projet, ils entendent un couple constitué de l'état du système à développer avant que le processus de développement ne commence et par l'état du système lorsque le développement est terminé. Par contexte de développement, ils entendent des critères organisationnels susceptibles d'influencer le processus de développement, comme par exemple les facteurs de contingence proposés dans [86]. Dans notre approche, la structure que nous proposons permet de rassembler les critères de définition du type de projet et du contexte de développement. Comme nous le verrons dans la section qui suit sur le contenu que nous avons proposé pour le Reuse Frame, nous avons intégré les critères de qualification des projets de développement proposés dans le cadre du projet Jecko (voir section 2.1.3) et les facteurs de contingence proposés dans [86]. Dans la suite de ce document, nous parlons de contexte de réutilisation pour qualifier les fragments de démarche et de situation de réutilisation pour qualifier le contexte de travail des acteurs du processus de développement qui réutilisent des fragments de démarche, mais

ces deux notions sont des situations selon la terminologie définie dans [15].

Le Reuse Frame est un arbre dont les nœuds portent des labels qui correspondent à des *critères* qui peuvent être ainsi spécifiés plus ou moins précisément. Dans le Reuse Frame, les nœuds de l'arbre qui sont proches de la racine portent des labels qui correspondent à des critères plutôt généraux alors que les nœuds feuilles ou proches des feuilles portent des labels qui correspondent à des critères plus raffinés donc plus précis. Un critère est en fait défini par le chemin de la racine jusqu'au nœud qui porte son label. Un critère a pour but d'expliciter soit le contexte dans lequel un fragment est susceptible d'être réutilisable soit une situation de réutilisation. `Code Reuse` est un exemple de critère qui permet de qualifier des fragments de méthode dédiés à des projets ou des situations de réutilisation qualifiant des projets dans lesquels il faut tenir compte d'une *legacy application*.

Si un fragment est qualifié par le critère `User Involvement` par exemple, cela indique que les directives qui y sont présentées sont dédiées à un projet de développement dans lequel les utilisateurs finaux de l'application sont impliqués. Cela peut convenir à un fragment décrivant des techniques de capture des besoins ou de tests nécessitant une interaction forte avec les utilisateurs finaux. Une situation qualifiée par ce même critère indique que l'ingénieur méthode recherche des fragments de méthode prenant en compte le fait que des utilisateurs finaux sont impliqués dans le projet de développement. `Virtual User Involvement` et `Real User Involvement` sont des exemples de raffinement (ou de spécialisation) du critère `User Involvement`. Ils permettent, sur cet exemple, de préciser si ce sont les utilisateurs réels qui participent au projet de développement ou s'il s'agit d'acteurs du projet qui jouent ce rôle. Ces critères plus précis permettent éventuellement de mieux qualifier les directives présentées dans un fragment de démarche ou une situation de réutilisation. Nous distinguons trois façons de raffiner les critères dans l'arbre qui constitue le Reuse Frame : un raffinement simple, un raffinement en des critères ordonnés les uns par rapport aux autres et un raffinement en des critères exclusifs les uns par rapport aux autres.

La *relation de raffinement ordonnée* permet de préciser un classement entre les différents nœuds qui partagent le même nœud père direct. Prenons l'exemple du critère `Code Reuse`. Comme nous l'avons expliqué dans la section 2.1.3, il peut être pertinent de distinguer des degrés de réutilisation de code. Le critère `Code Reuse` est donc raffiné en `Strong Code Reuse`, `Medium Code Reuse` et `Weak Code Reuse`. Dans ce cas de figure, il est intéressant de préciser que ces trois critères sont ordonnés les uns par rapport aux autres (dans un sens ou dans l'autre). Ce classement permet par exemple dans le cas où une situation de recherche est qualifiée par le critère `Weak Code Reuse` de proposer des fragments qualifiés par ce même critère bien sûr, mais aussi des fragments éventuellement qualifiés par le critère `Medium Code Reuse`, voire `Strong Code Reuse`, en ayant également la possibilité de préciser que les fragments qualifiés par ce dernier critère correspondent moins bien à la situation de réutilisation que ceux qualifiés par le critère `Medium Code Reuse`. De façon plus générale, la relation de raffinement ordonnée permet de rapprocher des fragments ou des situations ayant été qualifiés par des critères classés *après* ou *avant* le critère pris comme référence. Comme nous le verrons pas la suite, une distance de voisinage permet de quantifier l'éloignement entre les critères.

La troisième forme de raffinement est la *relation de raffinement exclusive*. Le critère `Time Pressure` est un exemple de critère raffiné à l'aide d'une relation de raffinement exclusive.

Des directives ou des situations de travail peuvent être caractérisées par le fait que le temps de développement imparti au projet est court ou bien, au contraire, par le fait que le temps de développement ne contraint pas particulièrement la façon dont le projet est mené. La pression temporelle apparaît alors comme une espèce de variable booléenne qui prendrait la valeur vraie quand le temps de développement imparti est court (il y a pression) et la valeur fausse quand au contraire le temps de développement imparti est long (il n'y a pas de pression). Comme il n'est pas possible dans notre structure d'associer plusieurs valeurs à un critère, nous raffinons ce critère en autant de sous-critères qu'il y a de valeurs possibles en indiquant que ces sous-critères sont exclusifs les uns par rapport aux autres. Ici nous proposons les sous-critères `High Time Pressure` et `Low Time Pressure`. Si un fragment de méthode n'est pas qualifié par le critère `High Time Pressure`, par exemple, cela ne signifie pas qu'il est alors dédié à un développement où les temps de développement sont relativement longs mais seulement que le critère de temps n'est pas déterminant pour qualifier le fragment. De même si un fragment n'est pas qualifié par le critère `Low Time Pressure`, cela ne signifie pas qu'il est alors dédié à un développement dont les temps de développement sont courts mais encore une fois que le critère de temps n'est pas déterminant pour qualifier le fragment. Nous voyons bien sur cet exemple qu'il est parfois nécessaire de faire apparaître dans le REUSE FRAME des critères (comme `Low Time Pressure` et `High Time Pressure`) qui représentent des caractéristiques opposées et qu'il est donc nécessaire d'indiquer explicitement qu'ils ne peuvent pas être utilisés dans un même contexte de réutilisation ou dans une même situation de réutilisation. Cette qualification de la relation de raffinement permet également d'indiquer que `Time Pressure` n'est pas un critère en lui même puisqu'il est raffiné de façon exclusive et qu'il ne peut donc être utilisé en tant que tel pour qualifier un contexte ou une situation. Un critère est donc en fait défini par le chemin de la racine jusqu'au nœud qui porte son label, sachant que le nœud en question ne doit pas être raffiné par une relation de raffinement exclusive. En effet, un nœud raffiné à l'aide d'une relation de raffinement exclusive n'est pas un critère en lui même. Il a pour but de rassembler des caractéristiques qui s'opposent.

Il n'y a pas de contrainte sur les nœuds fils de nœuds raffinés de façon exclusive ou ordonnée. Ils peuvent à leur tour être raffinés ou pas, à l'aide des relations de raffinement que nous proposons. Le critère `Dividing Project` est un autre exemple de critère pouvant être raffiné à l'aide d'une relation de raffinement exclusive. Deux sous-critères peuvent par exemple être proposés pour son raffinement : `One Single System`, pour qualifier des situations ou des fragments spécifiques à un projet géré dans son ensemble, et `Establishing Sub-projects`, pour qualifier des situations ou des fragments spécifiques à un projet découpé en sous-projets. `Establishing Sub-projects` peut à son tour être raffiné, à nouveau de façon exclusive, en `Establishing Processoriented Subprojects`, `Establishing System-oriented Sub-projects` et `Establishing Hybrid Sub-projects`. Il peut aussi être raffiné de façon simple uniquement à l'aide des critères `Establishing Process-oriented Sub-projects` et `Establishing System-oriented Sub-projects`. Dans ce cas, les situations ou les fragments liés à un découpage hybride du projet seront qualifiés par les deux sous-critères.

Enfin, la *relation de raffinement simple* permet de représenter des critères qui ne sont ni ordonnés ni exclusifs comme c'est le cas, par exemple, des critères `User Involvement`, `Real User Involvement` et `Virtual User Involvement`. En effet, des directives ou des situations peuvent être appropriées à la fois pour des utilisateurs finaux réels et virtuels. Ces deux critères ne sont donc pas exclusifs. Il n'y a pas d'ordre entre le fait de faire participer des utilisateurs

réels ou virtuels. La relation de raffinement est donc simple. Elle peut néanmoins être exploitée lors de la recherche de fragments de méthodes car elle permet de proposer dans le résultat de la recherche des fragments qualifiés par des critères plus précis ou plus généraux que ceux suggérés dans la requête. Nous pouvons par exemple imaginer une situation de réutilisation dans laquelle des utilisateurs finaux de l'application à développer sont présents (`Real User Involvement`). En plus des fragments de la bibliothèque de fragments éventuellement qualifiés par ce critère, les fragments qualifiés par le critère plus générique `User Involvement` pourraient également être inclus dans le résultat car ils peuvent également présenter un intérêt.

Le contenu du Reuse Frame est géré par les ingénieurs méthode qui le font évoluer en raffinant les critères existants ou en réorganisant les nœuds intermédiaires.

Une proposition de contenu pour le Reuse Frame

Le contenu du Reuse Frame que nous proposons est un raffinement des trois dimensions principales caractérisant le développement d'un système d'information [23] : la dimension humaine, la dimension organisationnelle et la dimension technique. Nous avons proposé un contenu établi à partir de différents travaux présentés dans la littérature sur la notion de contexte pour l'ingénierie des méthodes situationnelles.

En ce qui concerne la dimension technique, nous nous sommes appuyés sur nos travaux dans le cadre du projet Jecko (voir section 2.1.3). Dans ce projet, nous avions défini des aspects critiques pour la rédaction de spécifications durant le processus de développement logiciel que nous avons intégrés au contenu du Reuse Frame. Nous avons également pris en considération les différents types d'application pouvant être développés et susceptibles de nécessiter l'utilisation de directives adéquates : le développement d'applications dédiées aux échanges inter-organisations (*business to business*), aux échanges à l'intérieur d'une organisation (intra-organisation) et au commerce en ligne entre une organisation et un particulier (*business to customer*).

En ce qui concerne la dimension humaine, également prise en considération dans [38], nous avons recensé différents acteurs impliqués dans le développement d'un système d'information : analystes, développeurs, testeurs, etc. Nous avons distingué différents niveaux d'expertise pour chacun d'eux.

Enfin, pour ce qui est de la dimension organisationnelle, nous sommes partis d'un ensemble de facteurs proposé dans la littérature [86] pour caractériser les projets de développement de systèmes d'information. Nous distinguons trois familles de facteurs :

- les facteurs de contingence,
- les activités du cycle de développement,
- les facteurs liés à la gestion de projet.

Parmi les facteurs de contingence, citons l'implication des utilisateurs dans le projet de développement, comme les critères `User Involvement`, `Virtual User Involvement` et `Real User Involvement` que nous avons déjà introduits. C'est également dans cette famille de facteur que l'on trouve les critères liés au temps de développement (`High Time Pressure` et `Low Time Pressure`) et les critères liés à l'engagement de la direction de l'organisation.

Des éléments comme la complexité du projet, son niveau d'innovation, la formalisation avec laquelle il doit être décrit, son impact, sa taille, les ressources dont il dispose sont également rassemblés dans cette branche du Reuse Frame. La branche dédiée aux activités du cycle de développement rassemble toutes les activités de l'étude de faisabilité jusqu'aux tests. La gestion des risques et la gestion des versions apparaissent également dans cette partie du Reuse Frame. Ces activités sont plus ou moins précisément décrites. L'activité d'analyse est par exemple décomposée en analyse structurelle, fonctionnelle et comportementale. Cette classification constitue un point de départ. Elle peut être raffinée par les ingénieurs méthode de l'organisation dans laquelle elle est utilisée afin de décrire plus exactement les différentes activités de l'organisation en fonction de la connaissance méthodologique qui a été capitalisée dans l'organisation. Enfin la branche gestion de projet rassemble des critères liés à l'organisation du projet, à sa division en sous-projets, aux stratégies de développement, de réalisation et de livraison possibles, aux produits et aux activités du projet ainsi qu'au niveau de traçabilité souhaité. Le contenu de notre Reuse Frame est illustré ici au travers d'un détail présenté dans la figure 3.8. J'y détaille la partie du Reuse Frame dédiée aux critères liés à l'organisation du projet, à sa division en sous-projets et aux stratégies de développement, de réalisation et de livraison possibles. Aussi, nous distinguons l'organisation standard et l'organisation adaptée d'un projet, qui sont des critères exclusifs. En ce qui concerne la division en sous-projets, nous distinguons les sous-projets organisés en fonction du processus de développement de ceux organisés en fonction du système à développer et de ceux hybrides. Ces critères sont exclusifs du fait de la présence du critère d'organisation hybride. En matière de stratégie, nous distinguons les stratégies de développement, les stratégies de réalisation et les stratégies de déploiement. En ce qui concerne les stratégies de développement, nous distinguons de façon exclusive le guidage par les phases du processus de développement qui se suivent ou bien qui se superposent, l'aspect itératif du développement, le guidage par le développement d'un prototype et l'externalisation du développement. En terme de stratégie de réalisation, nous distinguons quatre stratégies exclusives : en un seul composant, en plusieurs composants en parallèle, en plusieurs composants de façon incrémentale et en plusieurs composants avec recouvrement des développements. Enfin, en terme de déploiement, nous distinguons trois stratégies exclusives : la stratégie en une fois, la stratégie incrémentale et la stratégie évolutionnaire.

La description complète du contenu du Reuse Frame a été publiée dans [65]. Elle est également disponible sur le site web du Laboratoire I3S[2].

Le Reuse Frame a été défini pour rassembler des critères permettant de spécifier le contexte de réutilisation des fragments de méthode d'une part et les situations de réutilisation de la connaissance méthodologique d'autre part afin de les apparier. Je vais maintenant détailler les notions de contexte de réutilisation et de situation de réutilisation.

[2]http://www.i3s.unice.fr/ mirbel/reuse-frame/html/rf.html.

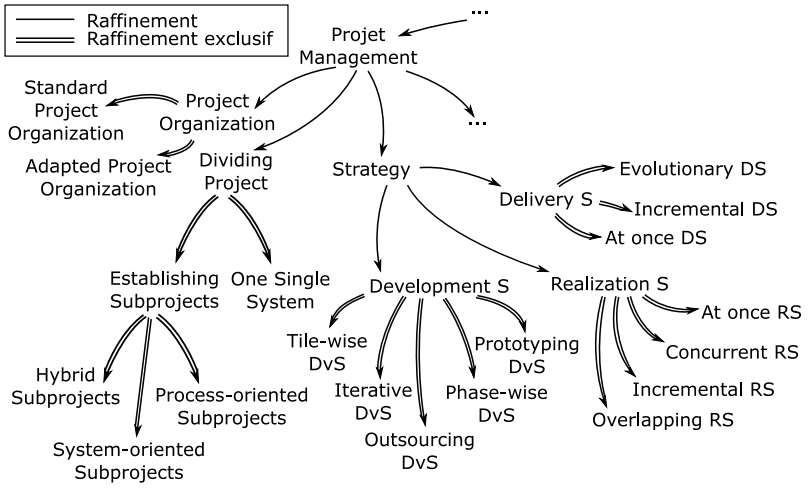

FIG. 3.8 – Une partie de la proposition de contenu pour le REUSE FRAME

3.2.2 Contexte et situation de réutilisation

Contexte de réutilisation

Dans le domaine de l'ingénierie des méthodes, les moyens de classification et les techniques de recherche sur les fragments de démarche qui ont été proposés sont basés sur les relations structurelles qui existent entre les fragments (spécialisation, composition, alternative, etc.) et sur les intentions de réutilisation [72]. Dans notre travail, nous avons cherché un moyen d'expliciter les similarités entre fragments de démarche de façon indépendante de leur structure à l'aide du REUSE FRAME.

Chaque fragment est doté d'un contexte indiquant comment il peut être réutilisé. Ce contexte est spécifié au moyen d'une liste d'au moins un critère pris dans le REUSE FRAME. Ce contexte de réutilisation permet de qualifier de façon pertinente les fragments de démarche dans le but de permettre leur réutilisation. Les fragments qui proposent des directives générales sont plutôt qualifiés à l'aide de critères généraux (c'est-à-dire correspondant aux nœuds proches de la racine de l'arbre). Les fragments plus spécifiques et donnant des directives plus précises sont plutôt qualifiés à l'aide de critères précis (c'est-à-dire correspondant à des nœuds proches des feuilles du REUSE FRAME ou aux feuilles elles-mêmes). C'est la responsabilité de la personne qui ajoute le fragment dans la bibliothèque de le qualifier de façon pertinente. La situation de réutilisation du fragment de méthode `RequirementAnalysis::Core::UI-View` présenté dans la figure 2.15 du chapitre 2 (section 2.2.2) et extrait de l'approche JECKO, par exemple, peut être décrite comme suit: `Requirement analysis`[3].

Situation de réutilisation

Lorsqu'il recherche des fragments de méthode durant le processus de construction ou d'adaptation d'une démarche spécifique à un projet, l'ingénieur méthode détermine une situation pour laquelle il souhaiterait trouver des fragments de démarche adaptés. Cette situation est

[3]Ce critère est extrait du contenu que nous avons proposé pour le REUSE FRAME.

spécifiée à l'aide des critères proposés dans le Reuse Frame. La *situation de réutilisation* est un ensemble de (au moins un) critère(s) compatible(s) et nécessairement présent(s) dans le contexte des fragments sélectionnés en réponse à la requête de l'ingénieur méthode. La *situation de réutilisation* est également constituée d'un ensemble de critères interdits, c'est-à-dire des critères qui ne doivent pas apparaître dans les fragments sélectionnés comme réponse à une requête de l'ingénieur méthode.

Un ingénieur méthode qui recherche par exemple des directives concernant la phase d'analyse des besoins, de façon à formaliser le processus de développement et à en assurer une certaine traçabilité, sachant qu'il travaille dans un environnement dans lequel les projets sont de petite taille, indiquera comme situation de réutilisation: `Requirement analysis`, `High Project Formality`, `Small Project Size`, `Strong Tracing`[4]. Comme il sait également qu'il travaille dans un environnement où il est difficile d'impliquer les utilisateurs finaux dans le processus de développement, il indiquera comme critère interdit : `Real User Involvement`.

3.2.3 L'appariement entre contexte et situation

Afin de permettre aux ingénieurs méthode de rechercher dans la bibliothèque des fragments qui correspondent à leur besoin, nous avons proposé un ensemble de mesures qui permettent de comparer :

- une situation et des contextes de réutilisation,
- des contextes de réutilisation entre eux.

La comparaison entre une situation de réutilisation et des contextes de réutilisation est utile lorsque des ingénieurs méthode cherchent dans la bibliothèque des fragments susceptibles de répondre à un besoin méthodologique auquel ils font face pendant le processus de création ou d'adaptation d'une démarche de développement. La comparaison entre des contextes de réutilisation permet de comparer des fragments les uns avec les autres. Cela est utile lorsque des ingénieurs méthode cherchent une alternative à une façon de faire. Cela leur permet d'enrichir la démarche de développement en y intégrant des façons de faire alternatives.

Nous avons défini une mesure de similarité qui permet de quantifier la proximité entre les contextes de réutilisation de deux fragments ou entre le contexte de réutilisation d'un fragment et une situation de réutilisation. L'intérêt de rassembler des fragments de méthode dans une bibliothèque est de les rendre disponibles au plus grand nombre d'ingénieurs méthode et de permettre à ces derniers de trouver le plus de fragments correspondant à leurs attentes. Les fragments dont le contexte de réutilisation ne correspond pas parfaitement au contexte ou à la situation défini par l'ingénieur méthode doivent donc, dans une certaine mesure, aussi être pris en considération. Dans ce cas de figure, la proximité entre le contexte ou la situation proposé et les contextes trouvés doit être quantifiée afin de permettre à l'ingénieur méthode de comprendre que les fragments trouvés ne correspondent pas parfaitement à ce qui est recherché. Ce type de situation doit d'autant plus être pris en considération que les démarches de développement ne sont pas formellement spécifiées, ce qui peut conduire à des définitions et à des interprétations légèrement différentes d'un individu à l'autre. De ce fait, les directives encapsulées dans les fragments de méthode peuvent être décrites de façon plus

[4]Ces critères sont extraits du contenu que nous avons proposé pour le Reuse Frame.

ou moins détaillée, plus ou moins précise et être qualifiées par des critères plus ou moins précis. Aussi, pensons-nous pertinent d'inclure dans le résultat de la recherche de fragments de méthode ceux dont la situation de réutilisation ne correspond qu'en partie à la situation ou au contexte défini par l'ingénieur méthode. Un contexte de réutilisation qui ne correspond pas parfaitement au contexte ou à la situation proposés par l'ingénieur méthode peut aussi, par exemple, correspondre à un contexte dans lequel les critères sont moins précis que ceux de la situation ou du contexte proposés par l'ingénieur méthode. Dans ce cas aussi, l'éloignement entre le contexte ou la situation proposés et les contextes des fragments trouvés doit être quantifié afin de permettre à l'ingénieur méthode de comprendre que les fragments retrouvés ne correspondent pas parfaitement à ce qui était recherché. Notre mesure de similarité est basée sur la comparaison d'ensembles, comme l'indice de Jaccard ou le coefficient de Dice. La comparaison porte ici sur les critères caractérisant soit les contextes de réutilisation de deux fragments soit le contexte de réutilisation d'un fragment et une situation de réutilisation.

Mesure de similarité entre deux contextes de réutilisation

En examinant le nombre de critères communs dans les deux contextes de réutilisation, une mesure de similarité, qui varie entre 0 et 1, est calculée pour indiquer à l'ingénieur méthode combien le contexte du fragment trouvé correspond à celui du fragment proposé. La valeur nulle correspond à des contextes qui n'ont aucun critère commun. La valeur 1 correspond à des fragments ayant des contextes identiques. La définition de la mesure est la suivante :

$$sm(fp, ft) = \frac{\Sigma^{i=1..n} d(c_{NC_{fp_i}}, NC_{ft})}{max(card(NC_{fp}), card(NC_{ft}))}$$

où fp est le fragment examiné par l'ingénieur méthode, ft est le fragment trouvé, $NC_{fp} = \{c_{NC_{fp_1}}, .., c_{NC_{fp_n}}\}$ est l'ensemble des critères qui composent le contexte de réutilisation du fragment fp, NC_{ft} est l'ensemble des critères qui composent le contexte de réutilisation du fragment ft et $d(c, C)$ est une *distance de voisinage* définie comme suit: $Si\ c\ \in C,\ d(c, C) = 1,\ sinon\ d(c, C) = 0$.

Mesure de similarité entre situation de réutilisation et contexte de réutilisation

La mesure de similarité est dans ce cas fonction du nombre de critères communs entre les critères spécifiés comme nécessaires dans la situation de réutilisation et le contexte de réutilisation du fragment examiné, du nombre de critères communs entre les critères spécifiés comme interdits dans la situation de réutilisation et le contexte de réutilisation du fragment examiné et du nombre de critères spécifiés comme nécessaires dans la situation de réutilisation.

Une valeur positive de cette mesure indique qu'il y a plus de critères nécessaires que de critères interdits présents dans le contexte de réutilisation du fragment examiné. Une valeur négative indique au contraire qu'il y a moins de critères nécessaires que de critères interdits. L'adéquation parfaite correspond à la valeur 1. La définition de la mesure est la suivante:

$$sm(s, f) = \frac{\Sigma^{i=1..n} d(c_{NC_{s_i}}, NC_f) - \Sigma^{j=1..m} d(c_{FC_{s_j}}, NC_f)}{card(NC_s)}$$

où s est la situation de réutilisation proposée, f est le fragment examiné, $NC_s = \{c_{NC_{s_1}}, .., c_{NC_{s_n}}\}$ est l'ensemble des critères spécifiés comme nécessaires dans la situation de réutilisation

s, $FC_s = \{c_{FC_{s_1}}, .., c_{FC_{sm}}\}$ est l'ensemble des critères spécifiés comme interdits dans la situation de réutilisation s, NC_f est l'ensemble des critères qui caractérisent le contexte de réutilisation du fragment f et $d(c, C)$ est une *distance de voisinage* définie comme suit : $Si\ c\ \in C,\ d(c, C) = 1,\ sinon\ d(c, C) = 0$.

Des exemples de contexte de réutilisation et de situation de réutilisation sont montrés dans la figure 3.9. Tous les critères utilisés sont extraits du contenu de Reuse Frame que nous avons proposé. Le `Fragment 1` est caractérisé par les critères `Real User Involvement`, `Low Time Pressure` et `Huge Project Size` [5]. Le `Fragment 2` est caractérisé par les critères `Real User Involvement`, `Low Time Pressure` et `Small Project Size` [6]. Deux situations de réutilisation sont également montrées. La `Situation A` est caractérisée par :

- les critères nécessaires `Real User Involvement` et `Low Time Pressure`,
- les critères interdits `Small Project Size` et `Low Project Formality` [7].

La `Situation B` est caractérisée par :

- les critères nécessaires `Real User Involvement` et `Strong Project Traceability`[8] ,
- les critères interdits `Huge Project Size` et `Low Time Pressure`.

Les mesures de similarités ont été calculées et montrent que les deux fragments sont plus proches de la situation de réutilisation `A` que de la situation de réutilisation `B`. Le premier fragment correspond parfaitement la situation de réutilisation `A`.

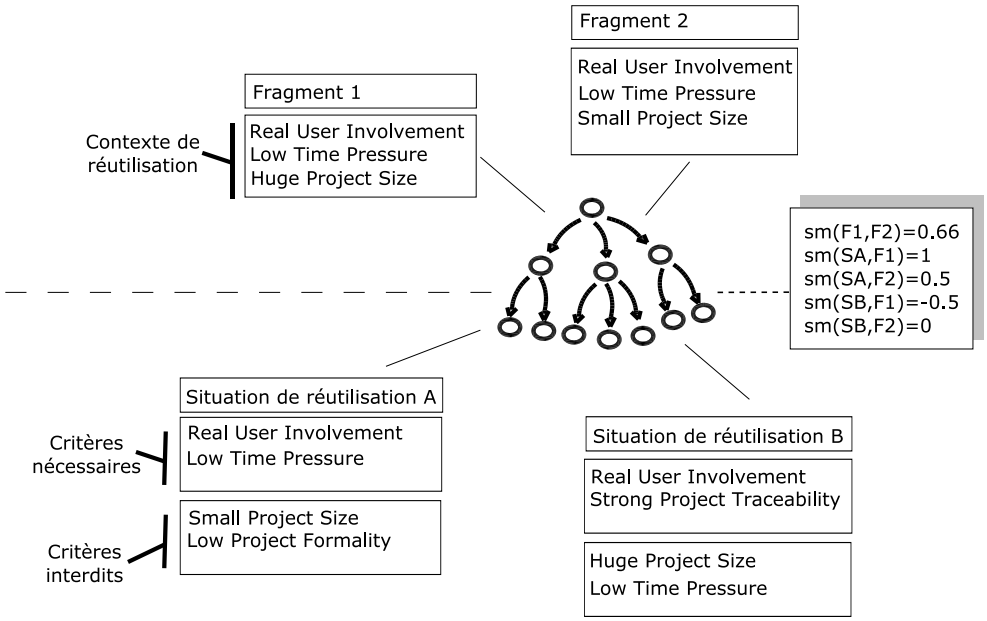

Fig. 3.9 – Exemples de calcul de mesure de similarité

[5]Ces critères sont extraits du contenu que nous avons proposé pour le Reuse Frame.
[6]Ce critère est extrait du contenu que nous avons proposé pour le Reuse Frame.
[7]Ce critère est extrait du contenu que nous avons proposé pour le Reuse Frame.
[8]Ce critère est extrait du contenu que nous avons proposé pour le Reuse Frame.

Mesure de similarité étendue

Les fragments de méthode qui incluent dans leur contexte de réutilisation des concepts plus généraux, plus spécifiques, avant ou après les critères spécifiés dans le contexte ou la situation spécifiés par l'ingénieur méthode doivent également être pris en considération, comme cela a été expliqué plus haut. Les fragments incluant des critères plus spécifiques dans leur contexte peuvent être intéressants parce qu'ils sont censés proposer des directives plus spécifiques. Ils ne portent vraisemblablement que sur une partie seulement du problème méthodologique de l'ingénieur méthode mais ils y répondent vraisemblablement aussi plus en détail. Si l'ingénieur méthode cherche par exemple des fragments dédiés à la réutilisation de code source et utilise pour cela le critère `Code Reuse` dans la spécification de la situation de réutilisation, il peut être également intéressé par des fragments qualifiés par les critères `Weak Code Reuse`, `Medium Code Reuse` ou `Strong Code Reuse` qui sont plus spécifiques que le critère qu'il a proposé. Un ingénieur méthode peut également être intéressé par des fragments qualifiés par des critères plus généraux que ceux employés dans le contexte ou la situation de référence. De tels fragments proposeraient des directives plus générales susceptibles d'intéresser l'ingénieur méthode. De la même façon que je viens d'illustrer l'intérêt d'exploiter la relation de raffinement simple du Reuse Frame, la relation de raffinement ordonné peut être exploitée pour prendre en considération des fragments qualifiés par des critères se situant, dans le Reuse Frame, avant ou après les critères proposés par l'ingénieur méthode pour qualifier la situation de réutilisation ou le contexte de réutilisation. L'exploitation des relations de raffinement du Reuse Frame peut également être intéressante pour les critères interdits. Élargir l'ensemble des critères interdits à des critères plus généraux permet d'interdire des branches complètes du Reuse Frame. L'extension de l'ensemble des critères interdits à des critères plus spécifiques permet de ne pas proposer dans l'ensemble des fragments résultat des fragments qualifiés par des critères trop spécifiques, c'est-à-dire probablement des fragments donnant des directives également trop spécifiques. De la même façon, élargir l'ensemble des critères interdits aux critères avant ou après ceux indiqués dans la situation ou le contexte de référence (en exploitant les relations de raffinement ordonnées) permet d'éviter d'inclure dans le résultat des fragments dont la portée dépasse celle des critères donnés dans la situation ou le fragment de référence.

L'extension de la requête de l'ingénieur méthode en permettant l'inclusion de critères plus généraux, plus spécifiques, avant ou après les critères indiqués par l'ingénieur méthode dans la situation ou le contexte de référence lui permettent d'élargir ou de réduire l'ensemble des fragments résultat. Si la recherche ne donne pas suffisamment de fragment résultat, l'ingénieur méthode peut choisir d'étendre l'ensemble des critères nécessaires de la situation de réutilisation ou le contexte de réutilisation du fragment (afin de prendre en compte les critères plus généraux, plus spécifiques, avant et après) dans le but de trouver plus de fragments. Si la recherche, au contraire, donne un résultat contenant trop de fragments, l'ensemble des critères interdits peut être étendu (en prenant en compte les critères plus généraux, plus spécifiques, avant et après) et ainsi permettre de réduire le nombre de fragments trouvés et présentés dans le résultat de la recherche. Ces possibilités d'extension sont résumées dans la table de la figure 3.10. Lorsque la mesure de similarité est calculée en étendant l'ensemble des critères nécessaires et/ou interdits, une distance de voisinage étendue ed, remplaçant d dans les mesures de similarité précédentes, a été proposée pour quantifier l'éloignement entre les critères du contexte ou de la situation de référence et les critères éventuellement plus

	Correspondance exacte	**Correspondance étendue**		
		Critères moins précis	**Critères plus précis**	**Critères avant/après**
Critères nécessaires	**Pour inclure des fragments**	**Pour trouver plus de fragments**		
		plus généraux	plus spécifiques	en intersection
Critères interdits	**Pour exclure des fragments**	**Pour trouver moins de fragments**		
		plus généraux	plus spécifiques	en intersection

FIG. 3.10 – Possibilités d'extension pour la mesure de similarité

généraux, plus précis, avant ou après les contextes des fragments examinés et susceptibles d'être présentés dans le résultat de la recherche. Pour cela, nous avons défini l'*extension* d'un critère. Il s'agit de l'ensemble de tous les critères plus généraux, plus spécifiques, avant ou après le critère examiné. La distance de voisinage étendue, *ed*, est calculable pour tous les éléments de l'extension d'un critère. Elle prend la valeur 1 s'il y a une correspondance parfaite entre deux critères (identiques dans la situation ou le contexte de référence et le contexte examiné). La valeur de la distance de voisinage étendue tend vers 0 au fur et à mesure que la distance dans le REUSE FRAME entre le critère examiné dans la situation ou le contexte de référence et le critère examiné dans le contexte trouvé augmente. La valeur 0 correspond à la racine du REUSE FRAME dans la cas de comparaison avec des critères plus généraux. Dans le cas de comparaison avec des critères plus spécifiques, la valeur 0 est associée au critère correspondant au label du nœud feuille de la branche la plus profonde du sous-arbre qui prend racine au nœud correspondant au critère examiné. En ce qui concerne les critères avant et après le critère considéré, la valeur 0 est associée respectivement aux critères dont le label correspond aux nœuds dont les rangs sont respectivement le plus bas et le plus élevé parmi les nœuds qui partagent le même nœud père direct que celui examiné. Les valeurs associées aux nœuds intermédiaires sont réparties régulièrement sur l'intervalle. La figure 3.11 résume les différentes situations possibles. Il peut y avoir plusieurs correspondances entre un critère du contexte ou de la situation de référence et le contexte du fragment examiné. Dans ce cas, c'est le critère qui maximise la mesure de voisinage qui est retenu comme résultat.

La mesure de similarité que nous avons proposée plus haut peut donc être étendue grâce à cette distance de voisinage étendue. Quand les critères qui sont présents dans le contexte de réutilisation du fragment examiné ne correspondent pas exactement aux critères de la situation ou du contexte de référence, la distance de voisinage étendue est utilisée pour calculer les mesures de similarité.

Des exemples de calcul de mesure de similarité étendue sont montrés dans la figure 3.12. Dans cette figure, une situation de réutilisation très simple, SU, caractérisée par un singleton, `a`, est comparée aux contextes de réutilisation des fragments F1 et F2. Il n'y a pas d'intersection entre l'extension de `a` et les critères du contexte de réutilisation de F2, la mesure de similarité entre SU et F2 est donc nulle. Il y a un critère commun entre l'extension de `a` et le contexte de réutilisation de F1. La mesure de similarité peut donc être calculée pour ce critère `2` à l'aide de la distance de voisinage étendue. La mesure de similarité entre SU et F1 est de 0.5. Nos travaux sur les mesures de similarités ont été publiés dans [65,61].

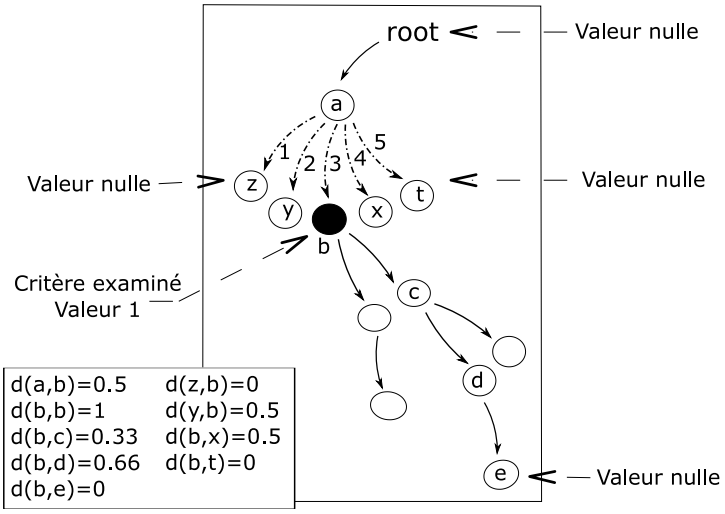

FIG. 3.11 – Décroissance de la distance de voisinage

Le REUSE FRAME est au cœur de l'environnement que nous avons proposé dans le cadre du projet SESAME que je vais maintenant présenter plus en détail.

3.2.4 La construction de méthode dans le projet SESAME

Le projet SESAME a porté sur le développement d'un environnement pour l'ingénierie des méthodes situationnelles dans lequel deux approches complémentaires ont été proposées. Ces dernières couvrent deux étapes fondamentales dans l'ingénierie des méthodes : la construction d'une méthode pour répondre aux spécificités d'un projet et la configuration d'une méthode pour répondre aux spécificités d'une situation de travail d'un membre du projet. Nous avons proposé une approche par assemblage de fragment de méthode pour construire une méthode spécifique à un projet et une approche par configuration pour personnaliser l'accès de la méthode à chacun des membres du projet. Ces deux étapes de construction et de configuration partagent une même bibliothèque de fragment de méthode et un même REUSE FRAME. Ce dernier aide à caractériser la situation de projet d'une part et la situation de travail d'autre part. Dans cette section, je vais présenter plus précisément la démarche de construction d'une méthode et l'utilité du REUSE FRAME dans une telle démarche. L'étape de configuration d'une démarche sera détaillée dans le chapitre 4 dédié à la flexibilité dans l'ingénierie des systèmes d'information.

Dans notre approche, une méthode est considérée comme un ensemble de fragments faiblement couplés et exprimés à des niveaux de granularité différents. Un fragment de méthode [72] est une partie autonome et cohérente de démarche qui supporte la réalisation d'activités spécifiques. Un exemple de fragment, selon le formalisme proposé par J. Ralyté, est montré dans la figure 3.13. Sur cet exemple, nous voyons qu'un fragment est caractérisé dans sa partie supérieure par une intention que le fragment doit permettre de réaliser (`Construct a use-case model`) et la situation de départ nécessaire pour pouvoir utiliser le fragment

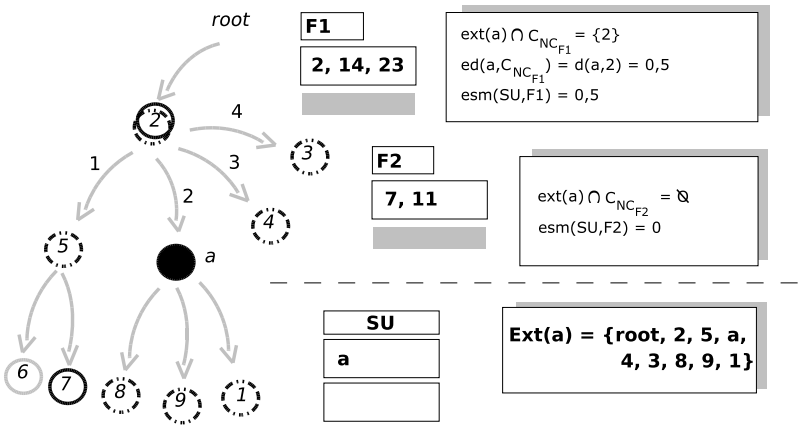

FIG. 3.12 – Exemples de distances de similarité étendues

(`Problem description`). Le corps du fragment contient une description du concept de cas d'utilisation et de sa structure (partie droite) et une description des directives permettant de réaliser l'intention but du fragment (partie de gauche). Les directives utilisées ici sont dites tactiques et sont présentées sous forme d'un plan composé de quatre sous-directives : `<(Prob. description),Identify an actor>*`, `<(Actor),Identify a use-case>*`, `<(Use case),Describe use-case>` et `<(Use-case model),Refine use-case model>`. Chaque sous-directive est également constituée d'un plan ou d'un choix entre plusieurs sous-directives. L'étoile indique que la directive peut être répétée plusieurs fois. Pour plus de détail sur la spécification du contenu d'un fragment de méthode, voir [65,72]. Dans le projet SESAME,

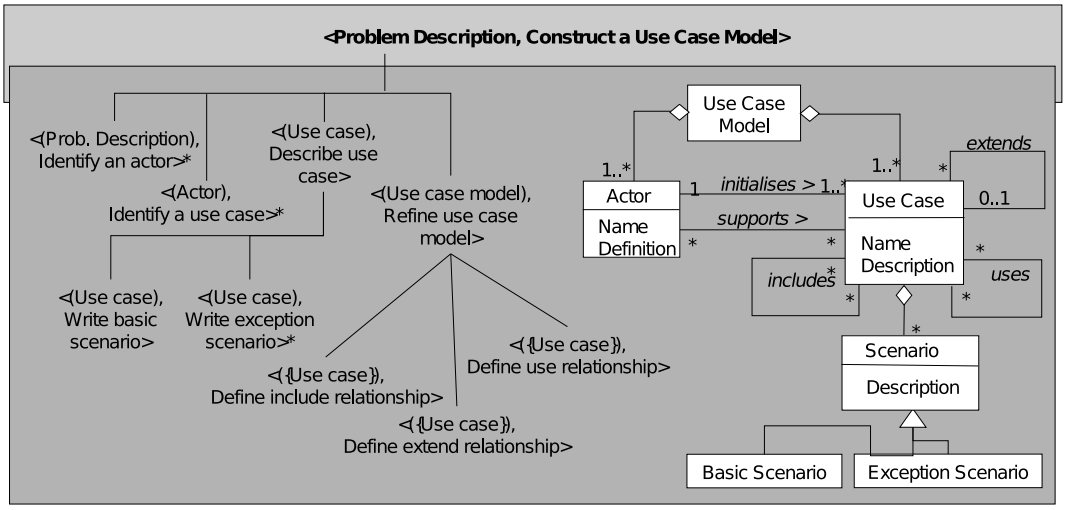

FIG. 3.13 – Un exemple de fragment de démarche [72]

nous avons plus particulièrement travaillé sur la notion de descripteur qui permet une description compacte du fragment pour aider à le retrouver durant le processus de construction d'une démarche. Un *descripteur* est associé à chaque fragment de démarche. Il permet de

définir le contexte dans lequel le fragment peut être réutilisé. Il est constitué d'un *contexte de réutilisation* et d'une *intention de réutilisation*. Le contexte de réutilisation est un ensemble de critères pris dans un Reuse Frame. Le contexte de réutilisation du fragment présenté dans la figure 3.13 doit, par exemple, permettre de décrire le fait que le fragment est applicable pendant l'activité d'élicitation des besoins. L'intention de réutilisation exprime l'objectif que le fragment aide à satisfaire. Elle est définie par un verbe, une cible et un ensemble de paramètres, tous spécifiés dans un glossaire. Par exemple, l'intention de réutilisation du fragment présenté dans la figure 3.13 est définie comme « `(Construire)`$_{\texttt{verb}}$`(Modèle de cas d'utilisation)`$_{\texttt{target}}$`(suivant stratégie` *`OOSE`*`)`$_{\texttt{manner}}$ ». Le descripteur contient également un *nom*, un *identifiant*, une description textuelle de son *objectif*, un *type* (atomique ou agrégat), une *origine* (la méthode dont il est extrait), ses composants s'il est de type agrégat. Les fragments qui correspondent à des *alternatives* ou qui ne lui sont *pas compatibles* peuvent également être précisés, ainsi que des *exemples* et des *expériences* d'utilisation. Le modèle du descripteur est donné dans la figure 3.14.

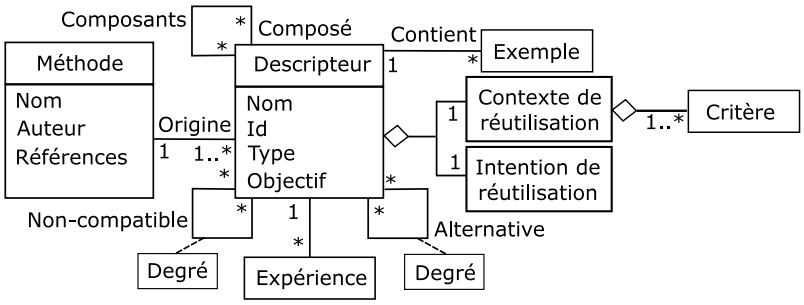

Fig. 3.14 – Modèle du descripteur d'un fragment de démarche

La phase de construction de méthode situationnelle par assemblage de fragment que nous avons proposée dans le cadre du projet Sesame a pour but de construire une méthode qui corresponde au mieux à la situation du projet. Cela consiste à sélectionner des fragments de méthode réutilisables, en fonction des besoins du projet considéré, parmi tous ceux présents dans la bibliothèque et à les assembler. Nous distinguons trois phases dans le processus de construction d'une démarche :

- la phase d'élicitation des besoins,
- la phase de recherche des fragments (correspondant aux besoins)
- la phase d'assemblage des fragments sélectionnés.

Le Reuse Frame est utilisé dans les deux premières phases du processus. Dans la phase d'élicitation des besoins, il permet de caractériser la situation de projet pour laquelle la méthode doit être construite. Durant la phase de recherche des composants, l'ingénieur méthode recherche des fragments qui satisfont chacun une partie au moins des besoins élicités durant la première étape du processus de construction. La recherche s'effectue donc en fonction des intentions des fragments présents dans la bibliothèque et de la situation de projet qui a été explicitée lors de la phase d'élicitation des besoins. Cette dernière est comparée aux contextes de réutilisation des fragments afin de sélectionner les fragments pertinents à la vue de la situation du projet. À l'issue de ce processus de construction dont plus de détails

peuvent être trouvés dans [65,72], une démarche spécifique au projet est obtenue.

3.3 Synthèse

Le projet WIDE m'a permis d'appréhender le processus de réutilisation dans sa totalité en contribuant à définir un langage pour représenter des patrons d'exceptions, une classification adaptée pour les organiser, des moyens de les spécialiser et de les instancier et enfin une démarche dédiée pour les mettre en œuvre.

Le projet SESAME m'a permis d'approfondir certains aspects du processus de réutilisation et notamment l'étape de sélection de fragment. Pour cela, nous avons proposé une technique d'appariement très simple basée sur l'utilisation d'une structure dédiée pour représenter les critères de qualification des fragments et des situations de réutilisation. Nous avons également proposé une façon différente d'organiser et de retrouver des fragments de méthode pour aider à construire et faire évoluer les démarches de développement.

En ce qui concerne le **processus de réutilisation de patrons de situations exceptionnelles dans les workflows**, nous avons proposé des mécanismes originaux de classification pour capitaliser et rechercher des patrons dans une bibliothèque structurée de patrons. La structure de la bibliothèque et les mécanismes proposés, qui sont le fruit d'un travail réalisé en collaboration avec des industriels, à partir d'étude cas fournies par eux, ont contribué à une meilleure connaissance sur les situations exceptionnelles considérées dans les workflows. Nous avons également proposé des mécanismes originaux d'adaptation de ces patrons durant le processus de modélisation d'un workflow : la spécialisation et l'instanciation.

En matière de **réutilisation de fragment de méthode** également, notre travail a porté sur des mécanismes de classification pour capitaliser et rechercher des composants de méthode dans une bibliothèque dédiée. Nous avons pour cela proposé des moyens originaux de spécifier le contexte d'un fragment et d'une situation de projet. Le point fort de notre proposition est de permettre de rechercher les fragments autrement que sur leurs intentions ou par mot-clé (comme c'est le cas dans la plupart des approches de la littérature), grâce au REUSE FRAME, qui est un cadre fédérateur pour les différents facteurs proposés dans les contextes de la littérature. L'originalité de ce travail réside également dans le fait de proposer des mécanismes pour exploiter les relations du REUSE FRAME afin de trouver des fragments proches de ceux recherchés et dans le fait de quantifier le résultat d'une recherche approximative.

Je vais maintenant présenter mon travail selon le troisième angle d'approche que j'avais choisi pour évoquer mes activités de recherche : celui de la flexibilité dans l'ingénierie des système d'information. J'ai déjà évoqué ce thème dans ce chapitre dans le sens où le développement par réutilisation permet dans une certaine mesure d'adapter des éléments réutilisables

pour concevoir un système d'information particulier. Les mécanismes d'instanciation et de spécialisation proposés dans le cadre du projet WIDE que j'ai présentés dans la section 3.1 de ce chapitre permettent la modélisation de spécification de workflow de façon flexible. De même l'approche que nous avons proposée dans le cadre du projet SESAME et que j'ai présentée dans la section 3.2 de ce chapitre permet la construction et l'évolution de démarches de développement afin d'adapter leurs caractéristiques aux spécificités de projets auxquels elles sont dédiées. Mais ce thème n'est pas central à ces deux approches. Dans le chapitre qui suit, je vais maintenant présenter plus en détail comment j'ai abordé le thème de la flexibilité dans le domaine de l'ingénierie des méthodes.

Chapitre 4

La flexibilité dans les systèmes d'information

Différentes approches ont été successivement proposées pour supporter de façon appropriée le développement de systèmes d'information. Elles l'ont été suivant des paradigmes différents [50] : certaines ont eu pour objectif de formaliser le processus de développement comme un processus de transformation ; d'autres ont cherché à le décrire comme un processus de décision ; d'autres encore ont proposé de formaliser le processus de développement comme un processus de résolution de problème. Des approches récentes l'ont appréhendé comme un processus d'apprentissage. Toutes ces démarches ont été développées dans le but de couvrir un spectre large de situations de projet et semblent finalement trop généralistes pour être appliquées telles quelles à un projet en particulier. En effet, les projets diffèrent les uns des autres par leur contexte de développement, de déploiement, de par la constitution des équipes, etc. Même les projets qui semblent similaires (même taille, même technologie) nécessitent des adaptations de leur démarche de développement du fait de leur contexte organisationnel.

Les approches dites agiles ont été une réponse à ce besoin d'adaptation dans le domaine de l'ingénierie logicielle. L'expérience montre que les ingénieurs d'application, c'est-à-dire les personnes qui mettent la démarche en pratique dans leurs tâches quotidiennes, préfèrent des démarches de développement légères parce qu'ils s'y sentent plus impliqués. Ces démarches dites agiles augmentent l'implication des ingénieurs d'application au contraire des démarches plus rigides où tous les choix significatifs sont faits lors de la construction de la démarche par les ingénieurs méthode avant que la démarche ne soit utilisée. Cependant les démarches agiles sont la plupart du temps des processus empiriques dérivés par catégorisation des entrées et des sorties et en définissant des contrôles pertinents pour contraindre le processus à rester dans les limites prescrites [74]. Dans la modélisation de démarches empiriques, les modèles sont strictement basés sur les données obtenues en entrée et en sortie des processus de façon expérimentale, sans recours à aucune loi concernant la nature fondamentale et les propriétés du système à développer, du projet ou de la démarche elle-même. Cela rend difficile la transposition de ce type de démarche d'une organisation à une autre. De plus, quand les processus agiles augmentent leur faculté d'adaptation et de flexibilité, il n'y a plus de garantie que le processus garde sa stabilité [83].

Un autre courant de travaux a cherché à proposer des mécanismes, des techniques et des ou-

tils pour construire des démarches de développement adaptées aux spécificités d'un projet en réutilisant et en assemblant des fragments de démarches existantes. Nous parlons d'ingénierie des méthodes situationnelles [42,11,73]. Dans ces approches, l'ingénieur méthode est responsable de la construction d'une bibliothèque de fragments après avoir identifié et extrait des fragments réutilisables de démarches existantes ou bien après les avoir générés à partir d'un métamodèle. Il est également en charge de construire une nouvelle méthodologie adaptée au projet et cela en assemblant des fragments réutilisables qui conviennent parmi tous ceux conservés dans la bibliothèque. Un spectre [42] a été proposé pour positionner les approches en ingénierie des méthodes les unes par rapport aux autres selon leur degré de flexibilité. L'échelle varie d'une flexibilité dite « faible » à une flexibilité dite « élevée ». Au niveau faible de ce spectre se trouvent les approches dites *rigides* tandis qu'au niveau élevé se situent les approches de *construction modulaire de méthode*. Les méthodes rigides sont complètement prédéfinies et laissent peu de place à l'adaptation alors que les méthodes modulaires peuvent être améliorées ou modifiées pour mieux convenir à une situation donnée. La sélection d'une méthode rigide permet de choisir la méthode la plus adaptée à un projet à partir d'un ensemble de méthodes rigides prédéfinies. Enfin, la sélection et l'adaptation d'une méthode permettent à chaque projet de sélectionner une méthode parmi différentes approches et de l'adapter aux besoins du projet.

L'augmentation de l'externalisation des développements et les nouveaux contextes de développement créent des besoins imprévus dans la gestion et le déploiement de méthodes de développement. Une démarche doit donc être également considérée comme un processus continu et évolutionnaire, du fait des évolutions techniques et organisationnelles, ainsi que des besoins des nouveaux développements. Les travaux sur le *method rationale* et sur les méthodes évolutionnaires [83] sont des réponses actuelles à ce besoin.

Un autre cadre de comparaison de la flexibilité dans le domaine de l'ingénierie des méthodes a été proposé dans [38]. Quatre axes de comparaison sont proposés par les auteurs : la flexibilité de *la structure de la méthode*, la flexibilité de *l'organisation du processus*, la flexibilité de *la nature du processus* et la flexibilité du *produit de la méthode*. La structure de la méthode rassemble les modèles, les langages, la démarche et les outils nécessaires à la méthode. Les auteurs distinguent une nature *figée* (l'ensemble des modèles, des langages, de la démarche et des outils est fixe), une nature *adaptable* (la méthode propose un cadre général permettant, par exemple, d'étendre un formalisme) et une nature *évolutive* quand la méthode peut être enrichie de nouveaux éléments. L'organisation du processus indique si l'enchaînement des activités qui constituent le processus de la méthode est entièrement défini (nature *prescriptive*), défini de façon générale pour être configuré (nature *rationnelle*) ou n'est pas décrit a priori (nature *dynamique*). La nature du processus décrit le processus de raisonnement pouvant être mis en œuvre durant l'utilisation de la méthode. Les auteurs distinguent une nature *libre* (sans guide), une nature *opératoire* (accomplissement de tâches prédéfinies), une nature *décisionnelle* (l'utilisateur doit prendre des décisions tout au long du processus) et une nature *exploratoire* (les alternatives de développement ne font pas partie de la définition du processus). Le produit de la méthode caractérise le degré de généricité des produits issus de l'utilisation de la méthode : *spécifiques*, *génériques* ou *méta produit*.

Le principe de flexibilité est donc au cœur de l'ingénierie des méthodes dont la problématique centrale est aujourd'hui celle de l'adaptation des démarches [76]. Cette adaptation peut être

fonction des contingences du projet [86], des besoins spécifiques d'un ou plusieurs acteurs du processus de développement ou de l'adaptation dynamique [76,38] dans le contexte du processus d'ingénierie des méthodes lui-même.

Dans ce contexte, nous avons essayé de répondre à cette problématique dans les deux projets portant sur des démarches de développement auxquels nous avons participé : JECKO et SESAME. Nous nous sommes plus particulièrement concentrés sur deux points. D'une part nous avons proposé des approches dans lesquelles nous avons placé la personnalisation de l'accès à l'information pour les acteurs du processus de développement au cœur de notre travail. En effet, ce point a été abordé dans le domaine de l'ingénierie logicielle au travers des approches agiles [74] et des entrepôts de bonnes pratiques partageables et utilisables par les différents acteurs du processus [43] mais il a peu été traité dans le domaine de l'ingénierie des méthodes. Les méthodes de développement doivent devenir des documents vivants et dynamiques, alors que les pratiques actuelles tendent à simplement mettre les documents de description des démarches sur les intranets comme des documents monolithiques qui ne sont par conséquent pas exploités durant les activités conceptuelles et ne permettent pas d'être facilement accédés, exploités et mis à jour [43]. Cette nécessité de construire des infrastructures de gestion de la connaissance relative aux pratiques accessibles et partageables par tous les acteurs du processus de développement est également explicitée dans [93]. Dans la section 4.1, je présenterai nos contributions sur ce point.

D'autre part, l'étude de la littérature sur les méthodes situationnelles montre que les approches proposées ont été clairement conçues pour être adaptables aux spécificités d'un projet en particulier. Mais peu de travaux expliquent comment adapter une méthode afin d'en faire une démarche partagée par tous les projets d'une organisation, ce qui est pourtant nécessaire dans les grandes organisations où plusieurs projets de développement sont menés en parallèle. Dans [34], les auteurs arrivent à cette même conclusion. Je montrerai dans la section 4.2 comment nous avons répondu à ce besoin.

4.1 Un accès personnalisé à la connaissance méthodologique

Des travaux sur la personnalisation de l'accès à l'information existent dans le domaine de la recherche d'information et dans celui des bases de données [10]. Les distinctions entre ces deux domaines se font sur les informations à partir desquelles la personnalisation est réalisée : il s'agit de document dans le domaine de la recherche d'information ou de table structurée dans le domaine des bases de données. Les modes d'accès sont également différents : ils sont généralement basés sur l'utilisation de mots-clés et un fonctionnement pas à pas dans le domaine de la recherche d'information alors qu'ils sont basés sur des expressions logiques et un fonctionnement global dans le domaine des bases de données [10]. Les auteurs de [10,89] distinguent la notion de profil de la notion de requête. La requête est l'expression du besoin d'un utilisateur. Le profil est un modèle personnalisé d'accès à l'information. Dans les approches en recherche d'information, la requête et le profil sont généralement mélangés alors que dans le domaine des bases de données une distinction est clairement faite entre la requête et le profil, qui vient enrichir la requête de l'utilisateur [10]. Dans [10], les auteurs présentent une synthèse des différentes approches en matière de personnalisation de l'accès à l'information. Ils proposent un modèle générique de représentation d'un profil. Les contenus

varient d'une approche à l'autre et tous les éléments ne sont pas systématiquement nécessaires, mais ils entrent tous dans le cadre suivant :

- données personnelles,
- centre d'intérêt,
- ontologie du domaine,
- qualité attendue des résultats délivrés,
- adaptation de la présentation des informations,
- sécurité et confidentialité,
- retour et préférences et
- informations diverses.

Les informations personnelles sont utiles dans le contexte du commerce électronique. Les informations sur les centres d'intérêt constituent le cœur du profil. L'ontologie du domaine complète la définition des centres d'intérêt en explicitant la sémantique des termes utilisés dans les requêtes et dans les profils. Les informations sur la qualité attendue permettent d'exprimer des préférences comme l'origine de l'information, sa précision, sa fraîcheur ou sa durée de validité. Les informations liées à l'adaptation de la présentation permettent d'indiquer les préférences de l'utilisateur en matière d'affichage des informations résultat de la requête. La dimension sécurité du profil peut concerner les données que l'on interroge ou que l'on modifie, les informations que l'on calcule, les requêtes de l'utilisateur ou les autres dimensions du profil. Les retours et préférences regroupent les informations collectées sur le comportement de l'utilisateur. Les informations diverses rassemblent des informations spécifiques à certaines applications.

Notre travail a porté sur la personnalisation de l'accès à l'information concernant les démarches de développement de systèmes d'information. Nous avons cherché à proposer une solution pour améliorer l'accès à la connaissance méthodologique contenue dans une démarche de développement, voire à l'ensemble de la connaissance méthodologique capitalisée dans une organisation. En effet, dans les démarches actuelles de développement de systèmes d'information, les ingénieurs d'application doivent connaître et comprendre la totalité de la méthode de développement et tous les concepts qui la composent pour être capables de l'exploiter, la plupart du temps seulement en partie. Les retours d'expérience des ingénieurs d'application montrent que les méthodes de développement sont perçues comme trop prescriptives et trop rigides [6]. Notre proposition s'appuie sur une description d'une démarche de développement à l'aide de fragment. Nous avons proposé une façon d'exploiter l'interface de ces fragments pour permettre aux utilisateurs de méthode un accès personnalisé à la démarche en sélectionnant les fragments adaptés à leur situation de réutilisation. Dans ce contexte, nous avons clairement distingué d'une part la requête de l'utilisateur (exprimée à l'aide d'une intention) indiquant l'objet des fragments de démarche susceptibles de l'intéresser et d'autre part le profil de l'utilisateur (exprimé au travers d'une situation de travail). Dans le modèle générique de profil proposé dans [10], plusieurs dimensions sont dédiées à l'accès aux informations disponibles sur le Web (données personnelles, qualité attendue, sécurité et confidentialité). Nous n'avons donc pas pris en compte ces dimensions dans notre travail. Nous nous sommes concentrés sur les centres d'intérêt et l'ontologie du domaine. Dans notre approche, les centres d'intérêt de l'utilisateur sont représentés sous forme d'une situation de réutilisation, comme cela a été présenté dans la section 3.2.2. Quant à l'ontologie du domaine, c'est le Reuse Frame que nous avons décrit dans la section 3.2.1 qui nous permet d'expliciter les termes décrivant les

centres d'intérêt de l'utilisateur.

Dans la plupart des approches en ingénierie des méthodes situationnelles, des fragments de méthode préétablis sont sélectionnés en fonction des caractéristiques du projet et assemblés pour former la méthode spécifique au projet. Une autre stratégie consiste à choisir, parmi un ensemble de méthodes supportant une grande variété de projets, la plus adaptée au type de projet. Une dernière phase de configuration permet d'adapter la méthode choisie aux caractéristiques du projet en question. Un exemple bien connu de ce type d'environnement est le RUP (Rational Unified Process [47]). Ces approches, même si elles différent les unes des autres, sont toutes construites selon un principe de séparation dans le temps entre la phase de construction de la méthode et la phase de mise en application de la méthode. Les auteurs de [38], qui considèrent les composants de méthode comme des services, proposent deux types de composition de services « méthodes », la *composition semi-statique* et la *composition dynamique*, qui tentent de réduire cette distinction entre la phase de construction et la phase d'utilisation de la méthode. Dans la composition semi-statique, le choix des services retenus est fait au moment de l'utilisation de la méthode et seule l'organisation des services est préétablie. Dans le scénario de composition dynamique aucune organisation n'est prédéfinie. Dans notre approche, nous avons cherché à mettre une partie de la construction de la méthode sous la responsabilité des utilisateurs de la méthode, c'est-à-dire durant la phase de mise en application de la méthode. Ce sont eux qui vont choisir, parmi les fragments préétablis par les ingénieurs méthode, dans la méthode spécifique au projet, les fragments qui correspondent à ce dont ils ont besoin dans leur travail quotidien. Dans ce sens, nous nous situons au même niveau de flexibilité que la stratégie semi-statique décrite dans [38]. Nos travaux diffèrent de ceux de [38] d'une part du fait que nous nous adressons à des ingénieurs d'application alors que le travail présenté dans [38] est plutôt destiné à des ingénieurs méthodes et d'autre part du fait que nous nous intéressons à l'accès à la connaissance relative aux pratiques alors que les auteurs de [38] cherchent à construire une méthode opérationnelle. Cette façon de faire nous semble être un bon moyen de rendre les méthodes plus attractives pour les utilisateurs, de faire en sorte que la perception qu'ils en ont soit moins négative, qu'elles soient ressenties comme moins rigides et moins prescriptives, tout en restant dans un cadre prédéfini.

Enfin, les environnements qui ont été proposés dans la littérature ne permettent pas de répondre à la question : comment exploiter la connaissance capitalisée dans les processus de développement, comment transformer ces derniers en des entrepôts de bonnes pratiques partageables et utilisables par les différents acteurs du processus [43] ? Les démarches de développement ne doivent pas être perçues comme de simples processus à suivre mais comme un point focal pour un processus d'apprentissage organisationnel [43]. Une bibliothèque de fragments de méthode peut être vue comme rassemblant des savoir-faire sur les façons de mettre en œuvre des démarches de développement. Des moyens doivent être proposés pour accéder à ce savoir-faire et cela de façon pragmatique et adaptée aux activités des acteurs du processus de développement.

Je vais maintenant montrer comment nous avons répondu à cette problématique, d'abord dans le cadre du projet Jecko puis dans le cadre du projet Sesame.

4.1.1 L'approche Jecko

Dans l'approche Jecko la fragmentation de la démarche (présentée dans la section 2.2.2 de ce mémoire) et la proposition de critères techniques pour caratériser les fragments de démarche (détaillés dans la section 2.1.3) permettent une certaine adaptabilité de l'approche. Lorsqu'un ingénieur d'application souhaite de l'aide pour mener à bien une activité conceptuelle du processus de développement, il peut rechercher parmi les fragments rattachés à la phase dans laquelle se situe son travail et en fonction de son contexte de travail s'il y a des directives susceptibles de l'intéresser. Dans notre approche, l'ensemble des directives qui correspondent au contexte et à la situation de travail d'un ingénieur d'application est appelé un itinéraire. La partie gauche de la figure 4.1 montre l'ensemble des fragments proposés couvrant la phase d'analyse des objets et du domaine métier du processus de développement. La partie droite de cette figure montre un itinéraire particulier correspondant au critère de développement d'une application incluant une interface homme-machine graphique (IHM). Grâce à cette façon de faire, l'ingénieur d'application va pouvoir accéder aux directives qui constituent la démarche de développement au travers du filtre qu'il a choisi. Dans l'exemple de la figure 4.1, le filtre est le critère : développement d'une application incluant une IHM. Ainsi, il n'aura pas à appréhender la méthode dans son ensemble et accédera directement au savoir-faire suceptible de l'intéresser.

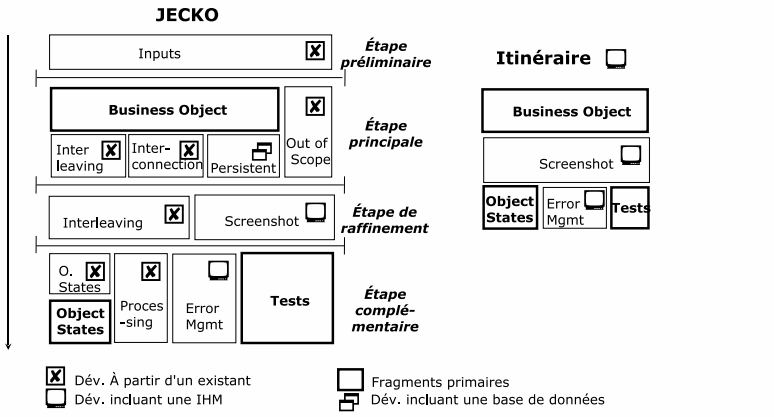

Fig. 4.1 – Exemple de phase et d'itinéraire dans l'approche Jecko

4.1.2 L'approche Sesame

Le projet Sesame a porté sur le développement d'un environnement pour l'ingénierie des méthodes situationnelles dans lequel deux approches complémentaires ont été rassemblées. Elles couvrent deux étapes fondamentales de l'ingénierie des méthodes : la construction d'une méthode pour répondre aux spécificités d'un projet et la configuration d'une méthode pour

répondre aux spécificités d'une situation de travail d'un ingénieur d'application. Cette façon de procéder permet de réduire les divergences potentielles dans la définition du processus de développement en retardant le moment de l'adaptation qui est alors effectuée au moment de l'utilisation de la démarche par les ingénieurs d'application et non pas au moment de la conception de la méthode par les ingénieurs méthode. La figure 4.2 schématise notre approche. L'étape de construction d'une démarche par assemblage a déjà été détaillée dans le chapitre 3 (voir section 3.2). Nous nous concentrons ici sur l'étape de configuration. Elle permet de

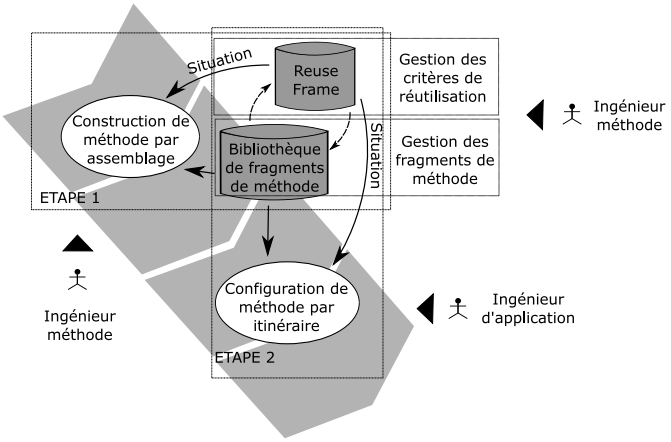

FIG. 4.2 – L'environnement proposé dans le projet SESAME

proposer un itinéraire personnalisé à chaque acteur du processus de développement qui le souhaite. Elle consiste à sélectionner les fragments de démarche qui satisfont les besoins de l'acteur à l'intérieur du cadre défini par la méthode spécifique au projet. En effet, plusieurs façons de faire ont pu être proposées dans la méthode pour réaliser une activité du processus de développement. De plus, la façon dont les différents fragments qui constituent une démarche peuvent ou doivent être utilisés n'est pas toujours complètement prédéterminée : les fragments peuvent être ou ne pas être liés par des relations de précédence. Plusieurs itinéraires sont donc possibles au sein d'une même démarche. Un itinéraire est composé d'une ou plusieurs séquences cohérentes de fragments de méthode prises dans la démarche du projet et correspondant à l'usage de la méthode par un acteur particulier du processus de développement. C'est une vue personnalisée du processus de développement. Le but d'un itinéraire n'est pas de proposer à l'acteur une description complète et détaillée de toutes les tâches qu'il va réaliser tout au long du processus de développement car il est nécessaire de laisser également de la place pour la créativité, mais de lui donner des conseils au moment où il en demande ou quand des directives pour des situations similaires à la sienne ont été accumulées dans la bibliothèque de fragment. Nous avons proposé trois stratégies pour construire cet itinéraire :

- **La stratégie guidée** : C'est la façon la plus simple de procéder. Elle consiste à sélectionner les fragments de méthodes qui correspondent à la situation de travail de l'ingénieur d'application parmi ceux qui constituent la méthode spécifique au projet. Cette façon de faire est recommandée aux néophytes ou aux ingénieurs d'application qui souhaitent suivre scrupuleusement la démarche construite pour le projet.
- **La stratégie libre** : Cette autre façon de faire consiste à rechercher les fragments directement dans la bibliothèque. Elle est recommandée lorsqu'un ingénieur d'application

cherche une réponse à un problème très précis (correspondant à l'utilisation d'un ou deux fragments mais pas à une démarche proprement dite).
- **La stratégie mixte** : Cette dernière façon de faire consiste à rechercher les fragments de méthode dans la démarche spécifique au projet et à compléter ensuite le résultat obtenu en recherchant des fragments directement dans la bibliothèque. Des fragments complémentaires peuvent en effet être pertinents s'ils représentent des alternatives à des fragments de la démarche spécifique. Dans ce cas de figure, après avoir utilisé cet itinéraire enrichi, l'ingénieur d'application peut donner son retour d'expérience aux ingénieurs méthode afin que la démarche spécifique au projet soit éventuellement enrichie par les fragments découverts et utilisés par les ingénieurs d'application.

Ces trois stratégies de configuration permettent de couvrir des besoins variés en matière de recherche de connaissance méthodologique. Elles permettent également de couvrir des profils différents d'ingénieurs d'application (néophytes, experts, etc.) et contribuent, dans une certaine mesure, à faire évoluer la démarche du projet (stratégie mixte). Ces trois stratégies s'apparentent aux stratégies de composition de services « méthodes » proposées dans [38] (statique, semi-statique et dynamique) en terme de niveaux de flexibilité.

Quelle que soit la stratégie de construction de l'itinéraire, nous distinguons trois phases dans le processus de configuration d'une démarche :
- la phase d'élicitation des besoins,
- la phase de recherche des fragments et
- la phase de construction de l'itinéraire.

La phase d'élicitation des besoins consiste à définir une *intention* et/ou une *situation de réutilisation* (voir section 3.2.2). L'intention est renseignée uniquement lorsque l'ingénieur d'application cherche une aide méthodologique sur un point précis et souhaite trouver des fragments qui concernent ce problème (stratégies libre et mixte). Une intention est constituée d'un ensemble significatif de verbes, de cibles et de paramètres pris dans un glossaire associé aux fragments de la bibliothèque (voir section 3.2.4). Le renseignement de la situation de réutilisation a deux fonctions :
- Dans le cas d'une recherche sur un point précis à l'aide d'une intention, la situation de réutilisation permet de préciser le contexte du problème et donc de réduire l'ensemble des fragments résultats de la recherche uniquement à ceux qui sont qualifiés par un contexte de réutilisation proche de celui donné par l'ingénieur d'application.
- Dans le cas d'une stratégie guidée, la recherche se fait uniquement sur la situation de travail qui permet d'exprimer le point de vue souhaité par l'ingénieur d'application sur la démarche pour la mettre en œuvre.

Un exemple de besoin peut être la recherche de fragments pour l'`écriture de scénario`. C'est un exemple d'*intention*. L'ingénieur d'application au travers de la définition d'une situation de réutilisation précise son contexte de travail : la phase du processus de développement (capture des besoins) et un développement rapide (les critères sont tirés de la proposition de contenu que nous avons faite pour le Reuse Frame) par exemple :
- critères nécessaires : `Requirement Elicitation`, `Low Time Pressure`.
- critères interdits : `Test`, `Analysis`.

La deuxième étape du processus de configuration consiste à rechercher les fragments de démarche qui correspondent à la situation et/ou à l'intention indiquées par l'ingénieur d'ap-

plication. Pour cela, nous avons proposé une mesure de similarité (voir section 3.2.3) qui permet de comparer une situation à un contexte de réutilisation. Un *seuil de similarité* permet à l'ingénieur d'application d'indiquer à partir de quelle valeur de mesure de similarité les fragments sont retenus pour faire partie de l'itinéraire résultat. C'est dans cette étape du processus de configuration que l'ingénieur d'application peut exploiter la mesure de similarité étendue en permettant ou non l'inclusion de critères plus généraux, plus spécifiques, avant ou après les critères indiqués dans la situation de réutilisation (voir section 3.2.3). Si la recherche ne donne pas suffisamment de fragments résultat, l'ingénieur d'application peut choisir d'utiliser la version étendue de la mesure de similarité (afin de prendre en compte les critères plus généraux, plus spécifiques, avant et après) dans le but de trouver plus de fragments de démarche. Si la recherche, au contraire, donne un résultat contenant trop de fragments, l'ensemble des critères interdits peut être étendu (en prenant en compte les critères plus généraux, plus spécifiques, avant et après le critère examiné) et ainsi permettre de réduire le nombre de fragments trouvés et présentés dans le résultat de la recherche. Les fragments de méthodes sélectionnés peuvent ne pas tous être reliés les uns aux autres. Ils peuvent très bien couvrir des étapes différentes et disjointes de la démarche. La recherche de fragments s'effectue parmi les fragments qui constituent la démarche (stratégie guidée et mixte) et/ou parmi l'ensemble des fragments présents dans la bibliothèque (stratégie libre et mixte).

La dernière étape, qui permet d'assurer la cohérence de l'itinéraire proposé, consiste à :

- Compléter l'ensemble des fragments résultats en y ajoutant des fragments complémentaires (*fragments associés*, voir section 3.2.4) de ceux trouvés dans l'étape précédente. Un *seuil de complémentarité* permet à l'ingénieur d'application de définir à partir de quelle valeur de *degré de nécessité* ou de *degré de proximité* (voir section 3.2.4) les fragments associés sont ajoutés à l'ensemble des fragments résultats.
- Supprimer de l'ensemble des fragments résultats ceux dont les fragments associés ajoutés sont incompatibles avec les critères donnés par l'ingénieur d'application dans la situation de réutilisation (critères interdits) ou avec les autres fragments résultats (fragments incompatibles). Un *seuil de compatibilité* permet à l'ingénieur d'application de définir à partir de quelle valeur du *degré d'incompatibilité* (cf section 3.2.4) les fragments sont considérés comme incompatibles les uns avec les autres au sein d'un même itinéraire.

La figure 4.3 synthétise les trois stratégies que nous avons proposées pour construire un itinéraire ainsi que les différentes tâches réalisées durant les trois étapes du processus de configuration pour chacune des stratégies.

Dans l'approche Sesame, nous avons également tenté de répondre au besoin de flexibilité dans le cadre d'une organisation dans laquelle sont menées à bien plusieurs projets en parallèle. C'est ce que je vais décrire dans la section suivante.

4.2 La flexibilité dans les démarches pour les organisations

Dans une organisation, plusieurs développements peuvent être menés en parallèle par des personnes appartenant à différentes équipes et participant à plusieurs projets. Dans ce cas, il est nécessaire, au sein d'une même organisation, d'avoir un cadre de travail commun dans lequel une certaine flexibilité est possible en fonction des projets et des individus. Comme cela

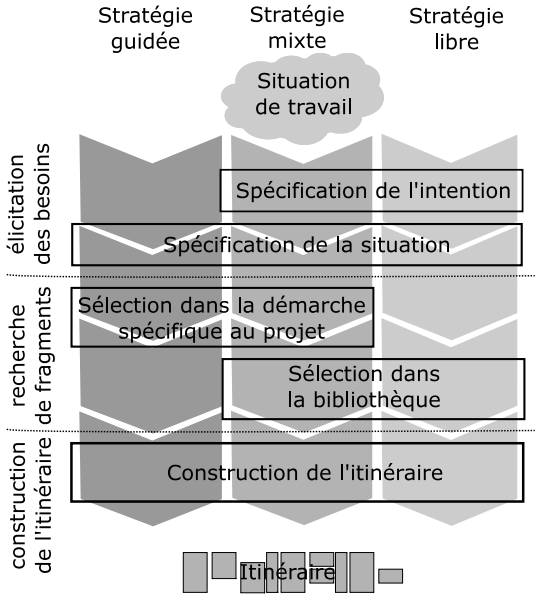

FIG. 4.3 – Les stratégies de configuration dans l'approche SESAME

a été souligné dans [34], peu de travaux de recherche ont porté sur comment utiliser l'approche situationnelle, dans une organisation, pour construire et utiliser une méthode de référence (*organization-wide method*) pour l'ensemble des projets de l'organisation, par opposition aux méthodes adaptées aux spécificités d'un projet en particulier (*project-wide method*). Une méthode conçue pour une organisation tout entière a pour vocation d'être utilisée sur une longue période de temps du fait de l'engagement et de l'investissement qu'elle demande pour être mise en place. D'un autre côté, l'évolution constante des techniques, des mécanismes et des technologies utilisées pour le développement de systèmes d'information nécessite des moyens de maintenir et de faire évoluer cette démarche.

De plus, une démarche définie pour l'ensemble de l'organisation doit permettre une certaine flexibilité au niveau de chaque projet. Or un des écueils des méthodes situationnelles actuelles est qu'elles deviennent difficilement comparables les unes par rapport aux autres du fait de leur adaptation à des contextes différents. Elles sont par conséquent également difficilement contrôlables par les responsables de projet et les responsables de l'organisation. Dans [44] une classification des approches en ingénierie des méthodes situationnelles selon quatre niveaux est présentée.

- Le premier niveau rassemble les approches dont le but est de maintenir le contenu modulaire de méthodes afin de pouvoir utiliser et réutiliser les fragments de méthode selon les perspectives des trois autres niveaux.
- Le deuxième niveau rassemble les approches dédiées à la construction de méthode pour répondre aux spécificités d'un projet. Selon les auteurs, ce type d'approche suppose un contrôle sur le développement et laisse certains choix ouverts là où un besoin d'adaptation a été anticipé. Les auteurs parlent de flexibilité planifiée.
- Le troisième niveau rassemble des approches permettant aux ingénieurs méthode de choisir les fragments qui correspondent le mieux à leurs besoins. Les auteurs parlent de

method fitting. Cette adaptation est contrainte par le cadre de la méthode spécifique au projet.
– Enfin la construction « à la volée » permet de prendre en considération des situations non anticipées, de construire un plan de travail efficace dans la situation courante et constitue le quatrième niveau.

L'approche que nous avons proposée dans le cadre du projet SESAME permet de couvrir ces différents niveaux de flexibilité. En effet, nous avons proposé de combiner deux approches complémentaires : une approche basée sur l'assemblage de fragment de démarche pour la construction de méthode (présentée dans la section 3.2.4) et une approche pour la configuration de méthode basée sur la notion d'itinéraires que nous avons présentée dans ce chapitre. Un des intérêts de notre approche est de permettre la définition d'une démarche adaptée à l'organisation qui constitue alors un cadre partagé par tous les projets et qui limite les adaptations faites au sein de ces derniers. L'environnement que nous avons proposé dans SESAME, centré autour du REUSE FRAME et d'une bibliothèque de fragments (qui constituent le niveau 1 de la classification proposée dans [44]) peut être en effet étendu afin de supporter :
– la construction d'une démarche au niveau de l'organisation (niveau 2 de la classification présentée dans [44]),
– une personnalisation de cette démarche au niveau de chaque projet (niveau 3 de la classification présentée dans [44]),
– un accès personnalisé pour chaque ingénieur d'application à l'intérieur de la démarche du projet (niveau 4 de la classification proposée dans [44]).

La figure 4.4 synthétise ces différents niveaux d'adaptation dans le cadre de l'approche SESAME.

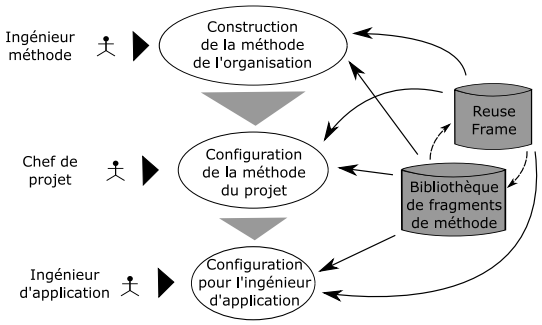

FIG. 4.4 – Niveaux d'adaptation d'une démarche de développement

4.3 Synthèse

Dans notre travail sur la flexibilité des démarches de développement de systèmes d'information, nous avons donc dans un premier temps contribué à définir des critères pertinents de caractérisation des fragments de méthode (les critères techniques proposés dans le cadre du projet JECKO). Nous avons ensuite contribué à favoriser l'accès à la connaissance relative aux pratiques aux différents acteurs du processus de développement. Dans le projet JECKO,

cela s'est concrétisé par la proposition de directives concrètes, fragmentées et associées à des critères permettant aux ingénieurs d'application de définir un itinéraire à l'intérieur de la démarche définie pour l'ensemble des projets de développement. Dans le projet SESAME, nous avons cherché à améliorer encore notre proposition en proposant une structure polymorphe et évolutive permettant aux ingénieurs méthode de définir eux-mêmes les critères de caractérisation des fragments de démarche et de personnalisation de l'accès à l'information. Enfin, nous avons également donné une réponse au besoin de flexibilité à la fois au niveau du projet et de l'organisation. Dans la première proposition que nous avons faite, dans le cadre du projet JECKO, nous avons présenté un support à une démarche partagée par tous les projets de l'organisation en terme de démarche et de profil UML. Encore une fois, cette première proposition a été étendue dans le cadre du projet SESAME où nous avons poussé plus loin nos idées en proposant un véritable environnement permettant de définir des démarches personnalisées au niveau de l'organisation et du projet.

L'approche SESAME répond d'ailleurs aux différentes formes d'adaptation que nous avons trouvées dans la littérature et que nous avons rappelées en début de ce chapitre : adaptation aux facteurs de contingence du projet, adaptation aux acteurs du processus de développement et adaptation afin de répondre à l'évolution de la méthode, elle-même fonction de l'évolution des technologies logicielles et des besoins des systèmes d'information à développer. Même si notre effort a particulièrement porté sur la proposition d'une solution concernant l'adaptation de la démarche aux acteurs du processus de développement et à la construction d'une démarche pour l'ensemble d'une organisation, nous n'en avons pas pour autant négligé la possibilité de faire évoluer la méthode dans le temps (ce que permet notre approche de construction de méthode par assemblage de fragments) et la nécessité d'adaptation en fonction des facteurs de contingence (ce que nous avons pris en compte dans la structure et le contenu de REUSE FRAME que nous avons proposé).

Nous avons aussi essayé de contribuer à changer la façon d'appréhender le processus de développement des ingénieurs d'application. La façon traditionnelle de le faire consiste à considérer le processus comme une succession de phases, éventuellement décomposées en étapes, dont chacune a pour but la production d'un produit. D'autres façons de l'appréhender ont également été proposées, comme nous l'avons indiqué au début de ce chapitre. Notre façon de percevoir le processus de développement comme un savoir-faire fragmenté et qualifié par une intention et un contexte de réutilisation nous permet de donner les moyens aux acteurs de ce processus de mieux expliciter comment et pourquoi les directives sont rattachées aux différentes activités du processus de développement et ainsi contribuer à une approche plus guidée par les intentions que par les produits à obtenir tout au long du processus.

En matière de **flexibilité**, l'originalité de notre approche se résume en deux points :

- Nous avons concentré nos efforts sur la proposition de mécanismes pour accéder aux connaissances relatives aux pratiques contenues dans une démarche de développement, voire à l'ensemble de la connaissance relative aux pratiques capitalisée dans une organisation, de façon personnalisée. À ma connaissance, ce problème de capitalisation des connaissances

relatives aux pratiques n'a pas été abordé en ces termes dans le domaine de l'ingénierie des méthodes situationnelles. Notre travail sur des mécanismes de classification et de recherche de fragments de méthode, basés sur le Reuse Frame, ainsi que la proposition de stratégies idoines pour accéder au contenu d'une démarche constituent une réponse à cette problématique.

– Sesame est une approche permettant de personnaliser une démarche de développement à plusieurs niveaux : au niveau d'une organisation, d'un projet dans une organisation et d'une personne dans un projet. Ces différents niveaux d'adaptation ont été décrits dans la littérature mais, à ma connaissance, aucune approche ne les appréhende au sein d'un même environnement, ce que nous proposons dans le cadre du projet Sesame en couplant deux approches qui couvrent deux étapes fondamentales de l'ingénierie des méthodes : la construction et la configuration d'une méthode.

Chapitre 5

Perspectives : de l'information à la connaissance

Comme je l'ai expliqué à plusieurs reprises dans ce document, la complexification et l'évolution de plus en plus rapides de l'informatique et des technologies de l'information ont considérablement changé la façon d'appréhender les systèmes d'information. Dans ce contexte, et comme cela a été montré dans ce mémoire, j'ai participé à plusieurs projets qui ont contribué à mieux appréhender certains aspects de modélisation, de réutilisation et de flexibilité dans le développement de systèmes d'information. Concevoir et implanter des systèmes d'information nécessite de prendre en considération des informations mais également des savoirs. Cela demande de choisir, de combiner et d'exploiter correctement des savoirs multiformes, difficiles à capter, à organiser et à assister, qui constituent une « intelligence pratique du monde » [27]. Dans ce contexte, les perspectives de mes travaux visent à aider les acteurs du processus de développement de systèmes d'information en améliorant l'accès à ces savoirs qui leur sont nécessaires dans leurs tâches quotidiennes. À court terme, ces perspectives portent plus précisément sur des techniques informatiques pour mieux appréhender les savoir-faire méthodologiques inhérents aux méthodes de développement, aspect que j'ai déjà abordé au travers du mécanisme de configuration de méthode proposé dans le cadre du projet Sesame. À plus long terme, je souhaite proposer, de façon plus générale, des techniques informatiques pour aider les acteurs du processus de développement d'un type particulier de systèmes d'information, les systèmes logiciel, à s'autoformer de façon collaborative aux langages et aux technologies du développement logiciel à l'aide des ressources disponibles sur le Web. Je vais détailler ces perspectives dans la suite de ce chapitre.

5.1 Le Web sémantique pour l'ingénierie des méthodes

Nos contributions dans le cadre des projets Jecko et Sesame, ont consisté à proposer des fragments d'expertise sur la mise en œuvre des méthodes de développement de système d'information, afin de rendre cette expertise réutilisable par plusieurs personnes et dans plusieurs projets au sein d'une organisation. Pour cela, nous avons proposé une structure arborescente, le Reuse Frame, pour rassembler des critères pertinents de caractérisation des fragments et ainsi permettre une organisation originale des fragments les uns par rapport aux autres. À court terme, une perspective intéressante à ce travail est d'intégrer le mécanisme de configuration de méthode que nous avons proposé dans une approche Web sémantique. Cela consiste

à modéliser le Reuse Frame comme une véritable ontologie à l'aide des standards du Web (RDFS et OWL) et à proposer, à partir du modèle de fragment présenté dans le projet Sesame, un modèle d'annotation des fragments de méthode en RDFS. Cela donnerait la possibilité de faire des recherches dans la bibliothèque de composant de méthode à l'aide d'un langage de requête standard comme SPARQL. Cette façon de faire permet de standardiser la façon de décrire les ressources (les fragments) à l'aide d'annotations sémantiques et la façon de définir les critères de qualification de ces ressources (dans une ontologie du domaine) facilitant ainsi le partage de fragments appartenant à différentes bibliothèques de composant de méthode. Cela permet également d'élargir les sortes de ressource considérées. Nous pensons par exemple proposer des modèles d'annotation de retours d'expérience informels, de savoir-faire empiriques ou de meilleures pratiques. Les requêtes porteraient alors sur l'ensemble des ressources considérées, améliorant ainsi l'accès aux savoirs méthodologiques hétérogènes que la communauté maintient dans sa bibliothèque de ressources. Le but de ce travail est de contribuer à permettre de mieux expliciter la façon dont la connaissance sur les méthodes de développement est mise en œuvre au quotidien et à faciliter son transfert. Une réflexion sur l'apport des communautés de pratiques et des techniques du Web sémantique pour l'ingénierie des méthodes a deéjà été intiée [60].

5.2 Intégration contextuelle de ressources informatiques

Les travaux en ingénierie des méthodes que j'ai présentés dans ce mémoire visaient à capitaliser et à partager des connaissances relatives aux pratiques de développement de systèmes d'information au sein d'une organisation. Aujourd'hui, l'échange et la capitalisation de savoirs dépassent le cadre de l'organisation. Le Web est devenu un acteur incontournable. Il permet à n'importe quel individu d'y rendre accessibles des informations et des services. C'est notamment vrai dans le domaine du développement logiciel. On trouve de nombreuses ressources portant sur les langages de programmation et les technologies informatiques. Des sites, dédiés à la programmation, à un langage ou à une technologie en particulier, proposent aussi bien des didacticiels que des FAQ, des documentations en ligne ou des forums de discussion. L'augmentation du nombre et de la popularité des logiciels libres a encore accentué ce phénomène. Ces sites sont régulièrement visités et enrichis par des individus participant à des activités de développement logiciel, que ce soit dans un cadre professionnel ou privé. De plus, l'évolution très rapide des langages et technologies du développement logiciel oblige les acteurs du processus de développement à constamment actualiser leurs connaissances. Mais les sites proposent des ressources de contenu et de qualité très hétérogènes. L'information pertinente peut finalement être difficile à trouver parmi toutes les sources d'information disponibles. En effet, les ressources en question s'adressent à des individus ayant des motivations et des profils différents (par exemple : amateurs ou professionnels, débutants ou experts). Elles décrivent des aspects différents du langage ou de la technologie du développement logiciel (conseils de programmation, détails techniques, algorithmes, etc.). Enfin, la recherche d'information sur ces sites est faite de façon déconnectée des activités de développement logiciel proprement dites. Dans ce contexte, le but de mes travaux à venir est d'aider les acteurs du processus de développement à accéder et à exploiter ces savoirs disponibles sur le Web et cela en liaison avec leurs activités de développement logiciel et de façon collaborative. Comme dans mes travaux précédents, je m'intéresse à l'aspect personnalisation de l'accès à l'information. Puisque les ressources sur lesquelles nous allons travailler sont celles publiées sur le Web,

nous avons choisi une approche Web sémantique. Dans la suite de cette section, je détaille plus précisément ces perspectives de travail. Je distingue les perspectives concernant l'annotation des ressources, celles concernant la contextualisation des annotations et enfin celles concernant l'autoformation.

5.2.1 Annotation des ressources

Notre première perspective concerne l'annotation des ressources sur les langages et technologies du développement logiciel. Il s'agit de proposer des modèles et des techniques pour expliciter la structure (auteur, code, commentaires, etc.) et les dimensions informatiques présentes dans les ressources disponibles sur le Web : une ressource peut être pertinente, par exemple, pour l'algorithme qui y est illustré, pour la façon dont la technologie y est mise en œuvre ou pour la fonctionnalité qui y est implémentée. Il s'agit également de proposer des modèles pour décrire les profils des acteurs du processus de développement recherchant des ressources. Un acteur peut se considérer comme néophyte ou expert ; il peut rechercher des ressources en tant qu'amateur ou bien dans un cadre professionnel. Nous souhaitons également que la recherche d'information soit corrélée aux activités de développement en cours réalisation. Un administrateur système, par exemple, ne recherchera pas les mêmes informations qu'un développeur. Il s'agit donc de proposer également des modèles pour annoter les motivations des acteurs du processus de développement logiciel.

Nos travaux sur le Reuse Frame, permettant de décrire les aspects humains, organisationnels et techniques caractérisant les activités du développement logiciel et les modèles de fragment proposés dans les projets Jecko et Sesame seront un point de départ pour ces modèles. Nous travaillerons aussi sur la proposition de techniques semi-automatiques pour annoter les ressources en fonction de ces modèles.

Nous nous plaçons dans une perspective de veille technologique collaborative au sein d'une communauté de développement en ligne, comme par exemple, des développeurs travaillant dans une même organisation et interagissant à l'aide d'outils collaboratifs sur l'intranet de leur organisation, des développeurs appartenant à une communauté de développement d'un logiciel libre ou des développeurs appartenant à un forum de discussion sur un thème particulier. Ces communautés possèdent généralement des ressources propres que leurs membres partagent sur un Intranet ou sur le Web. Il peut s'agir de documentation en ligne, de *how-to* spécifiques au domaine de la communauté (matériel, environnement, technologie utilisés) de rapports de bug ou simplement de messages échangés dans les forums de discussion par exemple. Nous parlons de ressources internes pour désigner les ressources appartenant à la communauté par opposition aux autres ressources publiées sur le Web que nous appelons ressources externes. Nous souhaitons proposer des modèles et des techniques pour annoter les ressources internes à la communauté de la même façon que nous souhaitons en proposer pour exploiter les ressources externes. Dans le projet Sesame, la recherche est faite sur l'intention et la situation de réutilisation de la ressource (en l'occurrence des fragments de méthode) qui doivent être explicitement décrites par les acteurs du processus de développement au moment de les placer dans la bibliothèque. En proposant une approche Web sémantique d'annotation semi-automatique des ressources internes, nous souhaitons d'une part alléger la tâche des acteurs du processus de développement et d'autre part exploiter la totalité de la ressource pour l'annoter (annotation limitée au descripteur et à l'intention dans le cas de fragments

de méthodes tels que nous les avons définis dans le projet SESAME par exemple). Nous ne cherchons pas seulement à décrire les ressources (internes et externes) et les acteurs susceptibles de les exploiter mais également à connecter ces différentes informations. Il s'agit pour nous de proposer des moyens d'exploiter de façon collective les savoirs disponibles sur le Web. Notre but est d'améliorer l'accès aux connaissances contenues dans les ressources externes en les accédant au travers des ressources internes décrites et comprises par les membres d'une communauté. La description de la version précise du système d'exploitation utilisée comme plate-forme de développement par les membres d'une communauté (ressource interne) peut, par exemple, aider à préciser si une ressource externe, portant sur le système d'exploitation en question, est pertinente ou non. Le lien entre les deux ressources ne sera établi que si les versions concernées par la ressource externe sont par exemple antérieures ou équivalentes à la version décrite dans la ressource interne. Ces liens seront également exploités pour relier les annotations des ressources externes entre elles. En effet, plusieurs ressources externes peuvent, par exemple, traiter d'un même problème technique et être liées les unes aux autres pour cette raison. D'autres ressources peuvent être liées parce qu'elles proposent des implantions dédiées à un même domaine d'application par exemple. On peut également imaginer des relations explicitant le fait qu'une ressource décrive plus précisément un sujet, ou au contraire le fait qu'elle le décrive d'une façon plus générale. Nous souhaitons proposer des moyens d'expliciter ces liens, à l'aide d'une typologie à définir, afin de permettre d'accéder aux ressources à travers différentes sortes de réseaux. Ces réseaux adaptables seront constitués de ressources internes ou externes en fonction des critères de recherche (ressources similaires, domaines d'application identiques, ressources plus ou moins spécifiques, etc.) et de leur utilisation (profil de l'acteur, motivation, activité de développement).

5.2.2 Contextualisation des annotations

Notre deuxième perspective concerne la contextualisation des annotations. Dans l'approche de configuration que nous avons proposée dans le cadre du projet SESAME, nous avons considéré un seul niveau de contexte. Il était modélisé pour chaque ressource ou situation de réutilisation à l'aide des critères rassemblés dans le REUSE FRAME. Nous souhaitons aller plus loin dans cette direction et distinguer deux niveaux de contextualisation :

- un niveau conceptuel décrivant de façon abstraite les informations pouvant cohabiter les unes avec les autres, ce qui correspond aux concepts décrits dans une ontologie du domaine (comme le REUSE FRAME dans le projet SESAME),
- un niveau objet correspondant aux informations contenues dans les ressources et décrivant de façon concrète le contexte d'une instance d'un concept dans une ressource donnée. Dans une page Web, par exemple, une technologie peut être illustrée dans un contexte particulier : une version de la technologie, un système d'exploitation particulier, une configuration matérielle, etc.

Nous souhaitons proposer des moyens d'expliciter ces relations définies au niveau des objets de façon distincte des relations entre concepts modélisées dans les ontologies. Une meilleure connaissance et explicitation de ce contexte nous semble nécessaire dans le cadre de recherches portant sur des ressources externes et hétérogènes (ce qui n'était pas le cas dans les projets JECKO et SESAME où un contexte partagé par tous les projets de développement de l'organisation pouvait être supposé). Ce contexte permettra :

- de préciser la connaissance contenue dans les ressources annotées : l'annotation d'une ressource sur une faille de sécurité précisant le(s) système(s) d'exploitation et la/les

version(s) incriminées permet de préciser l'intérêt de la ressource ;
- un guidage plus précis durant la recherche d'information dans les ressources annotées : la recherche de ressources à propos d'un système d'exploitation peut entraîner la reformulation de la requête afin de préciser la version du système d'exploitation ou la finalité de la recherche (installation d'un logiciel, programmation système ou autre).

De même nous souhaitons proposer des moyens de contextualiser les profils des acteurs ainsi que leurs motivations et leurs activités afin d'améliorer l'adéquation entre la connaissance extraite des ressources (algorithmes, technologies, etc.) et l'utilisation qui en est faite (profil de l'acteur, motivation, activité de développement, etc.).

5.2.3 Partage de connaissance et autoformation

Enfin, notre dernière perspective, toujours au sein de communautés de développeurs en ligne, est de soutenir l'activité d'autoformation, qui consiste à se former à son rythme à l'aide des ressources partagées dans une communauté. Selon [94], une communauté de pratique (CoP) est constituée d'un groupe d'individus partageant un intérêt, des problèmes et/ou une passion pour un sujet donné. Le but d'une CoP est de permettre à ses membres de coopérer et d'échanger des savoirs pour créer une valeur collective utile à chacun. La communauté leur permet de partager des ressources communes (savoirs, expériences, documents) et de collaborer dans un processus d'apprentissage collectif.

Les technologies du Web ont favorisé l'émergence de CoP virtuelles. Selon [96], les deux spécificités d'une CoP virtuelle sont d'exister en dehors de toute organisation particulière et, du fait de cette indépendance et de la dispersion géographique de ses membres, de reposer sur l'utilisation du Web. Les communautés de développement de logiciel libre, par exemple, entrent dans cette catégorie de CoP. Toujours selon [96], les CoP virtuelles, même si elles permettent le partage de connaissances tacites à un moindre degré que les CoP traditionnelles, sont particulièrement adaptées à la transmission de savoirs liés aux technologies du Web. Tirer le meilleur parti de ces outils ainsi que, de façon plus générale, des modèles et des technologies du Web sur lesquelles sont basées les CoP virtuelles, exige de proposer des services correspondant aux spécificités du domaine de connaissance de la CoP.

Nous avons présenté nos perspectives en matière d'annotation et de contextualisation des annotations de ressources sur les langages et technologies du développement logiciel. Nous souhaitons également proposer des mécanismes pour rassembler les avis des membres de la communauté sur les ressources et les liens qui les unissent, de manière à faire émerger, de façon collaborative, des ressources, des sites ou des auteurs par exemple. Nous cherchons à favoriser la construction d'une connaissance collective sur les ressources extérieures pertinentes pour la communauté.

D'autre part, nous souhaitons travailler sur les démarches mises en place par les acteurs pour retrouver des savoirs pertinents dans les ressources (internes et externes). Ces démarches se traduisent par des requêtes successives sur les ressources. Nous souhaitons proposer des moyens de conserver ses requêtes afin que les membres de la communauté puissent les améliorer et les partager. Ces requêtes pourront à leur tour être annotées et contextualisées. Elles constitueront des traces des fragments de démarches suivies par les membres de la communauté. Au delà de l'intérêt évident de conserver des requêtes en plus des ressources, nous souhaitons

supporter la construction collective de fragments de démarche par améliorations successives des requêtes et de leurs annotations par plusieurs membres de la communauté, ainsi que la capitalisation et la transmission de fragments de démarches entre membres.

Chapitre 6

Conclusion

Dans ce mémoire j'ai tenté de montrer l'étendue de mes contributions dans le domaine de l'ingénierie des systèmes d'information.

J'ai d'abord donné un aperçu des contextes dans lesquels j'ai travaillé, à savoir : le projet WIDE et le projet CHOROCHRONOS dans la cadre de mon séjour postdoctoral au Politecnico de Milan puis les projets JECKO et SESAME après mon recrutement à l'Université de Nice Sophia Antipolis. J'espère avoir montré au travers de ce mémoire la richesse et la variété de ces projets.

J'ai ensuite choisi d'illustrer mes contributions au sein de ces projets au travers de trois thèmes qui me semblent incontournables dans le domaine de l'ingénierie des systèmes d'information : la structuration de l'information, le partage et la réutilisation des informations et la flexibilité dans l'ingénierie des systèmes d'information.

En matière de structuration de l'information, j'ai distingué mes contributions sur la séparation des dimensions de celles sur la modélisation pour le partage et la réutilisation. Les travaux sur les documents multimédias interactifs et sur les situations dites exceptionnelles dans les workflows auxquels j'ai participé m'ont permis d'aborder des aspects temporels des systèmes d'information. Dans les deux cas, nous nous sommes efforcés de proposer une démarche, parfois outillée, pour faciliter le travail des concepteurs. Ma contribution au sein du projet JECKO a porté sur la définition de dimensions propres aux logiciels développés par l'entreprise Amadeus S.A.S. avec qui ce projet a été mené, dans le but d'améliorer la démarche de développement de l'entreprise en tenant compte des caractéristiques du logiciel à développer qui impactent de façon significative la façon dont les premières phases du processus de développement (analyse et conception) doivent être appréhendées. En matière de structuration de l'information pour le partage et la réutilisation, mes contributions ont porté sur la proposition de patron de situations dites exceptionnelles pour la démarche WIRES de conception de workflows et sur la proposition d'un modèle de spécification de fragment de méthode dans le cadre du projet JECKO.

En terme de partage et de réutilisation, mes contributions ont porté sur la proposition d'une démarche à base de patron pour la modélisation des situations dites exceptionnelles dans les workflows dans le cadre du projet WIDE et sur la spécification de fragments de méthode, es-

sentiellement en terme de connaissance pour la réutilisation, dans le cadre du projet Sesame.

En ce qui concerne la flexibilité dans les systèmes d'information, je l'ai essentiellement abordée dans le domaine de l'ingénierie des méthodes situationnelles où mes contributions ont cherché à apporter des réponses au besoin de personnalisation des démarches de développement à différents niveau d'une organisation : pour l'ensemble de l'organisation, pour un projet et pour un membre du projet dans un but de partage des connaissances relatives aux pratiques de développement de systèmes d'information.

Enfin, j'ai présenté les perspectives que je voyais à mes travaux dans le domaine de l'ingénierie des connaissances qui me semble particulièrement intéressant en ce qui concerne les problématiques évoquées dans ce mémoire.

Chapitre 7

References

[1] – **http://www.omg.org/technology/documents/formal/uml.htm**.

[2] – **Database Support for Workflow Management: The Wide Project** – *Kluwer Academic Publishers*, Norwell, MA, USA, 1999.

[3] – **Spatio-Temporal Databases: The CHOROCHRONOS Approach** – *Springer*, 2003.

[4] Allen, J.F. – **Maintaining Knowledge about Temporal Intervals.** – Commun. ACM, 26(11), 1983, pp. 832-843.

[5] Aydin, M.N. – **Method adaptation** – University of Twente, the Netherlands, 2007.

[6] Bajec, M.and Vavpotic, D. and Kirsper M. – **The scenario and tool-support for constructing flexible, people-focused system developement methodologies** – 13th International Conference on Information Systems Development - ISD 2004, Vilnius, Lituania, September, 2004.

[7] Baresi, L. and Casati, F. and Castano, S. and Fugini, M and Mirbel, I. and Pernici, B. – **WIDE workflow development methodology.** – International joint conference on Work Activities Coordination and Collaboration, San Francisco, California, USA, , February, 1999, pp. 19-28.

[8] Baresi, L. and Casati, F. and Castano, S. and Fugini, M. and Grefen, P. and Mirbel, I. and Pernici, B. and Pozzi, G. – **Database Support for Workflow Management: the WIDE Project** – *Kluwer Academics Publishers*, 1999.

[9] Booch, G. and Rumbaugh, J. and Jacobson, I. – **The Unified Modeling Language User Guide** – *Addison-Wesley*, 1998.

[10] Bouzeghoub, M. and Kostadinov, D. – **Personnalisation de l'information: aperçu de l'état de l'art et définition d'un modèle flexible de profils** – CORIA, 2005, pp. 201-218.

[11] Brinkkemper, S. and Saeki, M. and Harmsen, F. – **Assembly Techniques for Method Engineering** – International Conference on Advanced Information Systems Engineering, 1998.

[12] Brinkkemper, S. and Saeki, M. and Welke, R. – **Method Engineering: Principles of method construction and tool support** – *Chapman &Hall*, 1996.

[13] Brisaboa, N.R. and Mirbel, I. and Pernici, B. – **Constraints in Spatio-Temporal Databases: A Proposal of Classification** – 10th Conference on Advanced Information Systems Engineering - CAISE 1998 Workshop on Exploring Modeling Methods in Systems Analysis and Design - EMMSAD, , Pisa, Italy, June, 1998.

[14] Brusoni, V. and al. – **Later: Managing Temporal Information Efficiently.** – IEEE Expert, 12(4), 1997, pp. 56-64.

[15] Bucher, T. and al. – **Situational method engineering. On the differantiation of Contexte and Project Type** – IFIP WG8.1 working conference on situational method engineering: fundamentals and experiences (ME07), Geneva, Switzerland, September, 2007.

[16] Casati, F. and Castano, S. and Fugini, M and Mirbel, I. and Pernici, B. – **WERDE: A Pattern-Based Tool for Exception Design in Workflows.** – 1998, pp. 237-252.

[17] Casati, F. and Castano, S. and Fugini, M and Mirbel, I. and Pernici, B. – **Using Patterns to Design Rules in Workflows.** – IEEE Transaction on Software Engineering, 26(8), 2000, pp. 760-785.

[18] Casati, F. and Fugini, F.G. and Mirbel, I. – **An Environment for Designing Exceptions in Workflows.** – Advanced Information Systems Engineering, 10th International Conference CAiSE'98, Pisa, Italy, June, 1998, pp. 139-157.

[19] Casati, F. and Fugini, M and Mirbel, I. and Pernici, B. – **WIRES: A Methodology for Developing Workflow Applications.** – Requirement Engineering, 7(2), 2002, pp. 73-106.

[20] Casati, F. and Fugini, M.G. and Mirbel, I. – **An Environment for Designing Exceptions in Workflows.** – Information Systems, 24(3), 1999, pp. 255-273.

[21] Castano, S. and Fugini, M, and Mirbel, I. and Pernici, B. – **Workflow reference models** – 3015-2, WIDE ESPRIT project 20280, 1997.

[22] Cauvet, C. and Rolland, C. – **Object-Oriented Conceptual Modelling.** – CISMOD, 1992.

[23] Cauvet, C. and Rosenthal-Sabroux, C. – **Ingénierie des systèmes d'information** – *Hermes*, 2001.

[24] Cauvet, C. and al. – **Réutilisation dans l'ingénierie des systèmes d'information** – *Hermes*, 2001.

[25] Ceri, S. and Fraternali, P. – **Designing Database Applications with Objects and Rules: The IDEA Methodology** – *Addison-Wesley*, 1997.

[26] Ceri, S. and Ramakrishnan, R. – **Rules in Database Systems.** – ACM Comput. Surv., 28(1), 1996, pp. 109-111.

[27] Charlet, J and Reynaud, C. and Teulier, R. – **Ingénierie des connaissances pour les systèmes d'information** – *Hermes*, 2001.

[28] Chen, P.P. – **The Entity-Relationship Model - Toward a Unified View of Data** – ACM Transactions on Database Systems (TODS), 1(1), 1976, pp. 9-36.

[29] Conte, A. and Giraudin, J.-P. and Hassine, I. and Rieu, D. – **Un environnement et un formalisme pour la définition, la gestion et l'application de patrons** – ISI Formalisme et modèle pour les systèmes d'information, 6(2), 2001.

[30] Dechter, R. and Meiri, I. and Pearl, J. – **Temporal Constraint Networks.** – Artif. Intell., 49(1-3), 1991, pp. 61-95.

[31] Deneckère, R. and Iacovelli, A. and Kornyshova, E. and Souveyet, C. – **From Method Fragments to Method Services** – 20th Conference on Advanced Information Systems Engineering - CAISE 2008 Workshop on Exploring Modeling Methods in Systems Analysis and Design - EMMSAD, 2008.

[32] Eriksson, H.E. and Penker, M. – **Business modeling with UML, business patterns at work** – *Ipsen*, 2000.

[33] Firesmith, D.G. and Henderson-Sellers, B. – **The OPEN process framework - an introduction** – *Addison-Wesley*, 2002.

[34] Fitzgerald, B. – **The use of systems development methodologies in practice: a field study.** – Inf. Syst. J., 7(3), 1997, pp. 201-212.

[35] Gamma, E. and al. – **Design patterns: elements of reusable object-oriented software** – *Addison-Wesley Professional*, 1995.

[36] Giraudin, J-P. – **Complexité des systèmes d'information et de leur ingénierie** – e-TI - la revue électronique des technologies d'information, http://www.revue-eti.netdocument.php ?id=726, (3), 2007.

[37] Grefen, P. and Pernici B. and Sanchez G. – **Database Support for Workflow Management: the WIDE Project** – *Kluwer Academics Publishers*, 1999.

[38] Guzélian, G. – **Conception de systèmes d'information : une approche orientée service** – Université Paul Cezanne - Aix Marseille III, Juillet, 2007.

[39] Güting, R. H. and al. – **A foundation for representing and quering moving objects.** – ACM Trans. Database Syst., 25(1), 2000, pp. 1-42.

[40] Habrias, H. – **Le modèle relationnel binaire** – *Eyrolles*, 1988.

[41] Harmsen, F. – **Situational Method Engineering** – *Moret Ernst Young*, 1997.

[42] Harmsen, F. and Brinkkemper, S. and Oei, J.L.H. – **Situational method engineering for informational system project approaches.** – Methods and Associated Tools for the Information Systems Life Cycle, 1994, pp. 169-194.

[43] Henninger, S.R. – **Turning Development Standards Into a Repository of Experiences** – Software Process Improvement and Practice, 6(3), 2001.

[44] Jarvi, A. and Hakonen, H. and Makila, T. – **Developer driven approach to situational method engineering** – IFIP WG8.1 working conference on situational method engineering: fundamentals and experiences (ME07), Geneva, Switzerland, September, 2007.

[45] Karlsson, F. and gerfalk, P.J. and Hjalmarsson, A. – **Process Configuration with Development Tracks and Generic Project Types** – 6th CAiSE/IFIP8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'01), 2001.

[46] Khayati, O. – **Modèles formels et outils génériques pour la gestion et la recherche de composants** – Institut National Polytechnique de Grenoble, Décembre, 2005.

[47] Kruchten, P. – **The Rational Unified Process: An Introduction** – *Addison-Wesley Longman Publishing Co., Inc.*, Boston, MA, USA, 2003.

[48] Lamsweerde, A. van and al. – **The KAOS Project: Knowledge Acquisition in Automated Specification of Software** – AAAI Spring Symposium Series, Stanford University, American Association for Artificial Intelligence, 1991, pp. 59-62.

[49] Langefors, B. – **Theoretical Analysis of Information Systems** – *Lund, Sweden: Studentlitteratur*, 1966.

[50] Leppanen, M. – **Towards an Ontology for Information Systems Development** – 18th Conference on Advanced Information Systems Engineering - CAISE 2006 Workshop on Exploring Modeling Methods in Systems Analysis and Design - EMMSAD, 2006.

[51] Lings, B. and Lundell, B. – **Method-in-Action and Method-in-Tool: Some Implications for CASE** – 6th International Conference on Enterprise Information Systems - ICEIS 2004, Porto, Portugal, April, 2004.

[52] Mirbel, I and Rivieres, V. de – **Introducing Flexibility in the Heart of Analysis and Design** – 6th world multi-conference on systemics, cybernetics, and informatics - SCI 2002, , Florida, US, July, 2002, pp. 34-39.

[53] Mirbel, I. – **A fuzzy thesaurus for semantic integration of design schemes** – LIWED, Ambleside, United Kingdom, March, 1995.

[54] Mirbel, I. – **Semantic Integration of Conceptual Schemes.** – NLDB, Versailles, France, June, 1995.

[55] Mirbel, I. – **Un mécanisme d'intégration de schémas de conception orientée objet** – University of Nice-Sophia Antipolis, December, 1996.

[56] Mirbel, I. – **Une méthode d'intégration de schémas de conception.** – Bordeaux, France, June, 1996, pp. 161-177.

[57] Mirbel, I. – **Semantic Integration of Conceptual Schemas.** – Data Knowledge Engineering, 21(2), 1997, pp. 183-195.

[58] Mirbel, I. – **A polymorphic context frame to support scalability and evolvability of information system development processes** – 6th Int. Conference on Enterprise Information Systems - ICEIS 2004, , Porto, Portugal, April, 2004, pp. 131–138.

[59] Mirbel, I. – **Rethinking ISD methods : fitting project team members profiles** – Thirteenth International Conference on Information Systems development - ISD 2004, , Vilnius, Lituania, Sept., 2004.

[60] Mirbel, I. – **Connecting Method Engineering Knowledge: a Community Based Approach** – IFIP WG8.1 Working Conference on Method Engineering, Geneva, Switzerland., September, 2007.

[61] Mirbel, I. – **Contemporary Issues in Database Design and Information Syst ems Development** – *Idea Group, Inc.*, 2007.

[62] Mirbel, I. and Cavarero, J.L. – **An Integration Method for Design Schemas.** – Advances Information System Engineering, 8th International Conference, CAiSE'96, Heraklion, Crete, Greece, May, 1996, pp. 457-475.

[63] Mirbel, I. and Pernici, B. and Sellis, T. and Vazirgiannis, M. – **Checking the Temporal Integrity of Interactive Multimedia Documents.** – VLDB Journal, 9(2), 2000, pp. 111-130.

[64] Mirbel, I. and Pernici, B. and Vazirgiannis, M. – **Temporal Integrity Constraints in Interactive Multimedia Documents.** – ICMCS, Vol. 2, 1999, pp. 867-871.

[65] Mirbel, I. and Ralyte, J. – **Situational method engineering : combining assembly-based and roadmap-driven approaches** – Requirement Engineering Journal, 11(1), 2006, pp. 58–78.

[66] Mirbel, I. and Rivieres, V. de – **Adapting Analysis and Design to Software Context: The JECKO Approach.** – 8th International Conference on Object-Oriented Information Systems - OOIS 2002, , Montpellier, France, Sept., 2002, pp. 223-228.

[67] Mirbel, I. and Rivieres, V. de – **Adapting Analysis and Design to Software Context: the JECKO Approach** – I3S/RR-2002-14-FR, I3S Laboratory, 2002.

[68] Mirbel, I. and Rivieres, V. de – **Conciliating User Interface and Business Domain Analysis and Design** – 9th International Conference on Object-Oriented Information

Systems - OOIS 2003, , Geneva, Switzerland, Sept., 2003, pp. 383-399.

[69] Mirbel, I. and Rivières, V. de – **UML and the unified process** – *IRMA Press*, 2003.

[70] Prakash, N. – **On method statics and dynamics** – Information Systems, 34(8), 199, pp. 613-637.

[71] Pujalte, V. and Ramadour, P. and Cauvet, C. – **Recherche de composants réutilisables: une approche centrée sur l'assistance à l'utilisateur.** – INFORSID, 2004, pp. 211-227.

[72] Ralyté, J. – **Ingenierie des methodes a base de composants** – Universite Paris I - Sorbonne, January, 2001.

[73] Ralyté, J. and Rolland, C. – **An Assembly Process Model for Method Engineering.** – Advanced Information Systems Engineering, 13th International Conference, CAiSE 2001, 2001, pp. 267-283.

[74] Rising, L. and Janoff, N.S. – **The Scrum Software Development Process for Small Teams.** – IEEE Software, 17(4), 2000.

[75] Rochfeld, A. – **MERISE, an Information System Design and Development Methodology** – ER, 1986.

[76] Rolland, C. – **L'ingénierie des méthodes : une visite guidée** – e-TI - la revue électronique des technologies d'information, http://www.revue-eti.netdocument.php ?id=726, (1), 2005.

[77] Rolland, C. – **Method engineering : achievements, trends and challenges** – IFIP WG8.1 Working Conference on Method Engineering, 2007.

[78] Rolland, C. – **Method Engineering: Towards Methods as Services** – International Conference on Software Process (ICSE-ICSP), 2008.

[79] Rolland, C. and Plihon, V. and Ralyté, J. – **Specifying the Reuse Context of Scenario Method Chunks.** – CAiSE, 1998, pp. 191-218.

[80] Rolland, C. and Proix, C. – **A Natural Language Approach for Requirements Engineering.** – CAiSE, 1992, pp. 257-277.

[81] Rolland, C. and Richard, C. – **The REMORA Methodology for Information Systems Design and Management** – IFIP Working Conference on Information System Design Methodologies : A Comparative Review, 1982.

[82] Ross, D.T. and Schoman, K.E. Jr. – **Structured Analysis for Requirements Definition.** – IEEE Trans. Software Eng., 3(1), 1977, pp. 6-15.

[83] Rossi, M. and al. – **Managing evolutionary method engineering by method rationale** – Journal of the Association for Information Systems, 5(9), 2004, pp. 356-391.

[84] Rumbaugh, J. and al. – **OMT. Modélisation et conception orientées objet.** – *Prentice Hall London*, 1995.

[85] Shanon, C.E. – **A mathematical theory of communication** – Bell System Technical Journal, 271948, pp. 379–423 and 623–656.

[86] Slooten, K. van and Hobbes, B. – **Characterizing IS development projects** – IFIP WG8.1 Working Conference on Method Engineering: Principles of method construction and tool support, Great Britain, 1996, pp. 29-44.

[87] Störrle, H. – **Describing Process Patterns with UML.** – 8th European Workshop on Software Process Technology, Lecture Notes in Computer Science, 2001, pp. 173-181.

[88] Sugumaran, V. and Storey, V.C. – **A semantic-based approach to component retrieval** – SIGMIS Database, 34(3), New York, NY, USA, 2003, pp. 8–24.

[89] Tchienehom, P.L. – **Modèle générique de profils pour la personnalisation de l'accès à l'information.** – INFORSID, 2005, pp. 269-284.

[90] Tolvanen, J.P. – **Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence** – University of Jyvskyl (Finlande), 1998.

[91] Tryfona, N. and Pfoser, D. and Hadzilacos T. – **Modeling Behavior of Geographic Objects: An Experience with the Object Modeling Technique.** – Advanced Information Systems Engineering, 9th International Conference CAiSE'97, 1997, pp. 347-359.

[92] Vazirgiannis, M. – **Interactive Multimedia Documents: Modeling, Authoring, and Implementation Experiences** – *Springer*, 1999.

[93] Weerde, I. van de and Brinkkemper, S. – **Meta-modeling for situational analysis and design methods** – 2007.

[94] Wenger, E. and McDermott, R. and Snyder, W.M. – **Cultivating communities of practice** – *Harvard Business School Press*, 2002.

[95] Yu, E. – **Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering.** – 3rd IEEE International Symposium on Requirements Engineering (RE'97), January 5-8, 1997, Annapolis, MD, USA, 1997, pp. 226-235.

[96] Zarb, M.P. – **Modelling participation in Virtual Communities of Practice** – London, UK, , 2006.

[97] Zhang, Z. and Lyytinen, K. – **A Framework for Component Reuse in a Metamodelling-Based Software Development.** – Requirement Engineering, 6(2), 2001, pp. 116-131.