Flexibilité des processus de développement à la conception et à l'exécution : application à la plasticité des interfaces homme-machine

Eric Ceret

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Eric CÉRET

Thèse dirigée par **Sophie Dupuy-Chessa**
et codirigée par **Gaëlle Calvary**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG)**
dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (MSTII)**

Flexibilité des processus de développement à la conception et à l'exécution : application à la plasticité des Interfaces Homme-Machine

Thèse soutenue publiquement le **4 juillet 2014**,
devant le jury composé de :

Madame Sylvie Pesty
Professeur, Université Pierre Mendès-France, Présidente

Monsieur Jean Vanderdonckt
Professeur, Université Catholique de Louvain, Rapporteur

Monsieur Óscar Pastor López
Professeur, Universidad Politécnica de Valencia, Rapporteur

Madame Corinne Cauvet
Professeur, Aix - Marseille Université, Membre

Madame Gaëlle Calvary
Professeur, Grenoble INP, Membre

Madame Sophie Dupuy-Chessa
Maître de Conférences, Université Pierre Mendès-France, Membre

Monsieur Philippe Renevier-Gonin
Maître de conférences, Université de Nice Sophia Antipolis, Membre

REMERCIEMENTS

De nombreux remerciements, dans les premières pages des manuscrits de thèse, pointent l'aventure intellectuelle et humaine vécue par les auteurs. Leur lecture m'a amené à me questionner… et à identifier l'importance de nombreuses personnes dans mon aventure.

Tout d'abord, j'allais dire évidemment, j'ai bénéficié de l'encadrement et du soutien de mes directrices, Sophie Dupuy-Chessa et Gaëlle Calvary. Notre aventure commune était au départ une gageure : un doctorant plus âgé qu'elles, avec 20 ans d'expérience comme ingénieur, est un profil très atypique. Elles ont relevé le défi, et je les remercie très profondément pour tout ce qu'elles m'ont apporté, tant en matière de méthodologie scientifique qu'en conseils divers ou en rigueur, parmi tant d'autres domaines. Leur complémentarité, leur niveau d'exigence, leurs questions (pour ne pas dire leurs remises en cause, parfois !), leur soutient, leurs encouragement, sont des éléments décisifs de cette thèse. Je garde en mémoire quelques moments inoubliables, de rires partagés, de confrontation(s !) et d'élaboration commune.

Ensuite, je voudrais remercier Marc. Je n'aurais jamais imaginé pouvoir préparer une thèse s'il n'avait pas été là depuis quelques années, armé d'une certitude que je pouvais y arriver. Bien plus que moi, il croyait. Encourageant dans les moments de doute, questionnant dans les moments de certitude, il a su être à la fois présent dans le soutien et délicat dans la présence. Cette thèse n'aurait jamais vu le jour sans lui.

Il y a bien sûr ma famille. Persuadés de n'y pouvoir rien comprendre, ils m'ont poussé à trouver des métaphores pour parler de mon travail. Construire une IHM ou une méthode, c'est (un peu) comme construire une maison ou une voiture. Il faut des plans (des modèles !) des règles (d'ergonomie !), une prise en compte du terrain et des envies des futurs habitants (le contexte…), etc. Mais tout en comprenant difficilement pourquoi il y avait besoin de faire de la recherche pour construire une maison (on sait faire depuis des siècles, non ?), ils ont aussi su trouver une place, émotionnelle et pragmatique, dans le soutien personnel aussi bien que matériel (ah, les paniers repas livrés à domicile pendant la rédaction du manuscrit !).

Mon neveu, Augustin, a fort bien su me rappeler la « vraie vie ». Ses textos, ses appels, ses visites continuent à me faire sourire…

Les amis, les collègues, les discussions dans un café ou sur la terrasse du LIG, les questions partagées avec les autres doctorants de l'équipe, les recommandations reçues au détour d'une conversation, ont aussi contribué à m'aider à maturer ma réflexion. Je voudrais ici remercier l'équipe IIHM pour son accueil, son soutien et ses critiques. Plus particulièrement, je pense à Joëlle pour son franc-parler et son bon sens jamais démenti. En dehors de l'équipe, j'ai aussi une pensée reconnaissante pour l'équipe SIGMA, qui m'a encouragé tout au long de ce travail, et pour Nadine Mandran, pour sa disponibilité, ses conseils et sa formation.

Enfin, je voudrais aussi remercier tous les membres du jury pour leur présence, leurs conseils, leurs idées, leurs encouragements. Soumettre l'ensemble des publications qu'ils m'ont suggérées est un nouveau défi !

TABLE DES MATIERES

PARTIE II

PARTIE III

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

ATL : Atlas Transformation Language

AUI : Abstract User Interface (Interface utilisateur abstraite)

CD : Compact Disk

CMPF : Component-oriented Process Modeling Framework

CTT : ConcurTasks Tree (Paternò et al. 1997)

CUI : Concrete User Interface (Interface utilisateur concrète)

DBMS : DataBase Management System

DTD : Document Type Definition

EICS : Engineering Interactive Computing Systems (conférence)

EMF : Eclipse Modelling Framework

FDD : Feature Driven Development (Coad et al. 1999)

DUI : Final User Interface (Interface utilisateur finale)

IHM : Interface Homme-Machine

IHM : Conférence Francophone sur l'Interaction Homme-Machine

IM²AG : Informatique, Mathématiques et Mathématiques Appliquées de Grenoble

Inforsid : INFormatique des ORganisations et Systèmes d'Information de Décision

ISO : International Organization for Standardization

JRD : Joint Requirements Development Sessions

M2Flex : double sens, Models to Flexibility ou Metamodel of Flexibility

MB-IDE : Model-Based Interface Design Environment

OMG : Object Management Group

Promote : Process Models Taxonomy for Enlightening choices

RAD : Rapid Application Development (Martin 1991)

RCIS : International Conference on Research Challenges in Information Science

SIGAL : Systèmes d'Information et Ingénierie Avancée du Logiciel

SPEM : Software and Systems Process Engineering Meta-Model

UML : Unified Modeling Language

WPP : Work Product Pool (Gonzalez-Perez & Henderson-Sellers 2008)

XML : eXtensible Markup Language

XP : Extreme Programming (Beck 1999a)

1 INTRODUCTION

Au tournant des années 2000, D. Thevenin et J. Coutaz (Thevenin & Coutaz 1999) (Thevenin 2001) pointaient la diversité croissante des contextes d'utilisation des applications informatiques. Ils notaient en particulier la diversification des dispositifs, tant dans leurs formes que dans leurs finalités, ainsi que la nécessité, pour l'utilisateur final, que les applications soient capables de les accompagner dans ces différents contextes. Ils envisageaient une « *utilisation mixte, du gros calculateur [au] petit dispositif, du fixe comme du mobile, de l'ordinateur palpable à l'ordinateur évanescent* ».

Depuis 2000, la multiplication des dispositifs s'est encore amplifiée. Par exemple, le taux d'équipement des foyers en micro-ordinateurs est passé de 48% en 2003 à 78% en 2011, 31% des foyers ayant même plusieurs ordinateurs à cette date (Credoc 2011). L'ordinateur fixe cède peu à peu la place aux dispositifs mobiles : sur douze mois glissants en juin 2013, 65% des ventes d'ordinateurs concernaient des ordinateurs portables (Nguyen Van 2013). L'ordinateur perd de son hégémonie dans les technologies de l'information et de la communication pour le grand public : en 2011, 44,4% des Français étaient équipés d'un téléphone intelligent et 18,4% des foyers avaient une tablette (Jaimes 2013), alors que ces équipements étaient confidentiels ou inexistants en 2001. Cette « explosion », pour reprendre le mot de P. Farago, n'est pas propre à la France : le taux de croissance de l'équipement en téléphones intelligents équipés d'Android (le système mobile de Google) ou iOS (celui d'Apple) est de 401% en Chine, 279% au Chili ou encore 171% en Inde (Farago 2012).

Les usages aussi se sont diversifiés. Ainsi, 12% des Français déclaraient avoir une connexion à domicile en 2000, alors qu'ils étaient 64% en 2010 (INSEE 2011). Les dispositifs mobiles sont omniprésents : 66% des possesseurs d'un téléphone intelligent s'en servent pour consulter leurs courriers électroniques, 62% pour télécharger des applications et 28% pour regarder la télévision (Credoc 2011), une activité qui était réservée à un poste fixe quelques années auparavant. D'après (Meeker & Wu 2013), les utilisateurs mobiles consultaient leur téléphone environ 150 fois par jour en 2013 : 23 fois pour consulter des messages, 22 fois pour passer ou recevoir des appels, 18 fois pour regarder l'heure, 13 fois pour écouter de la musique, 12 fois pour jouer, 9 fois pour se connecter à des réseaux sociaux, 8 fois pour filmer ou photographier, 6 fois pour consulter l'actualité, 5 fois pour consulter un agenda, 3 fois pour faire une recherche sur Internet etc. De tous ces usages, combien existaient en 2001 ? Des internautes affirment que l'intégration d'un appareil photographique dans un téléphone a été commercialisée en novembre 2000 au Japon (Debjit 2010) ; Sony revendique la paternité de l'intégration des lecteurs audio de haute qualité avec un téléphone mobile en 2005 (Sony-Ericsson 2005). Ces utilisations se retrouvent dans l'usage d'Internet : chaque mois, en 2012, les sites de Google recevaient 1,2 milliards de visiteurs uniques (Meeker & Wu 2013) alors qu'ils étaient 36 millions en

2002 (Fox 2002). Le volume de données partagées sur Internet est passé de 130 à 1800 exaoctets[1] entre 2005 et 2011, la prévision pour 2015 étant de 7910 exaoctets (EMC 2011).

Le constat de D. Thevenin et J. Coutaz sur la nécessité qu'une application soit capable de s'adapter à son contexte d'usage est donc plus vrai que jamais. Il est en effet inimaginable de créer des versions spécifiques pour tous les dispositifs, tous les systèmes dont ils sont équipés et plus encore, pour toutes les utilisations possibles : la combinatoire des paramètres rend cette solution totalement impossible économiquement.

1.1 La plasticité des Interfaces Homme-Machine, une réponse au besoin d'adaptation aux contextes d'usage

Depuis quinze ans, les travaux portant sur la plasticité des interfaces homme-machine (IHM) ont posé les concepts pour relever ce défi (Thevenin & Coutaz 1999) (Calvary et al. 2001) (Paternò & Santoro 2002) (Limbourg et al. 2005) (Sottet et al. 2007) (Vanderdonckt 2008) (Aquino et al. 2010). La plasticité d'une IHM est définie comme sa capacité à s'*adapter* à son *contexte d'usage* dans le respect de ses *bonnes propriétés* pour l'humain (Coutaz 2007). Ces « bonnes propriétés » sont définies selon deux points de vue : fonctionnel et non-fonctionnel, c'est-à-dire à la fois les services attendus par l'utilisateur et la qualité de ces services. Ce sont les exigences de l'utilisateur, la définition du problème selon sa propre perspective. Le contexte d'usage permet de capturer les contraintes qui pèsent sur la solution. Il est défini par le triplet <utilisateur, plateforme, environnement> (Thevenin 2001), où

- L'utilisateur représente l'utilisateur du système interactif. Il peut être décrit par des données générales telles que son âge, sa taille, etc. mais aussi par ses compétences dans son métier, en informatique, etc. Ses compétences peuvent évoluer au fil du temps et militer, en conséquence, pour une adaptation de l'IHM ;
- La plateforme dénote les équipements matériels et logiciels disponibles pour l'interaction. Toute évolution peut de même questionner la pertinence d'une adaptation de l'IHM. Par exemple, lorsque la batterie du téléphone faiblit, l'application courante peut migrer du téléphone à la plate-forme la plus proche ;
- L'environnement se réfère à l'espace physique qui accueille l'interaction. Il peut être décrit par ses conditions lumineuses, sonores, sociales, etc.

Le défi de la plasticité est alors de calculer la « meilleure solution » pour l'utilisateur compte tenu du problème et des contraintes sur la solution. Le cadre de référence élaboré au cours du projet Cameleon (Calvary et al. 2001) définit les concepts nécessaires pour la conception d'IHM plastiques. En effet, pour que le système qui calcule cette « meilleure solution » puisse raisonner sur la structure de l'IHM et sur son adaptation, il est nécessaire de lui fournir des descriptions de la situation et du problème à résoudre. La situation et le

[1] Un exaoctet représente 10^{18} octets, soit un milliard de gigaoctets. Le chiffre exact de 2011 est 1.987.262.613.861.770.000.000 octets…

problème sont alors représentés par des modèles : modèle de contexte et modèle de tâches (Breedvelt-Schouten et al. 1997), auxquels est ajouté un modèle de domaine, qui représente les concepts manipulés par l'IHM. Une série de transformations de modèles permet, comme le montre la figure 1, de réifier ces modèles en IHM Abstraite (« *Abstract UI* »), puis en IHM concrète (« *Concrete UI* ») et enfin en IHM finale (« *Final UI* ») c'est-à-dire exécutable ou interprétable. Nous parlons ici de réification car ces différents modèles sont considérés comme plus concrets au fur et à mesure qu'ils sont plus proches du code. L'IHM Abstraite est « *l'expression du rendu des concepts et fonctions du domaine d'une façon indépendante des interacteurs disponibles sur la* [plateforme] *cible* » (Calvary et al. 2003). Elle permet, par exemple, de représenter les espaces de travail et les relations entre ces espaces. L'IHM concrète réifie ces espaces de travail en interacteurs. Il ne s'agit toujours pas de code, mais d'une maquette qui rend explicite l'aspect final de l'IHM (Calvary et al. 2003). L'IHM finale incarne l'IHM concrète sous la forme de code dédié à

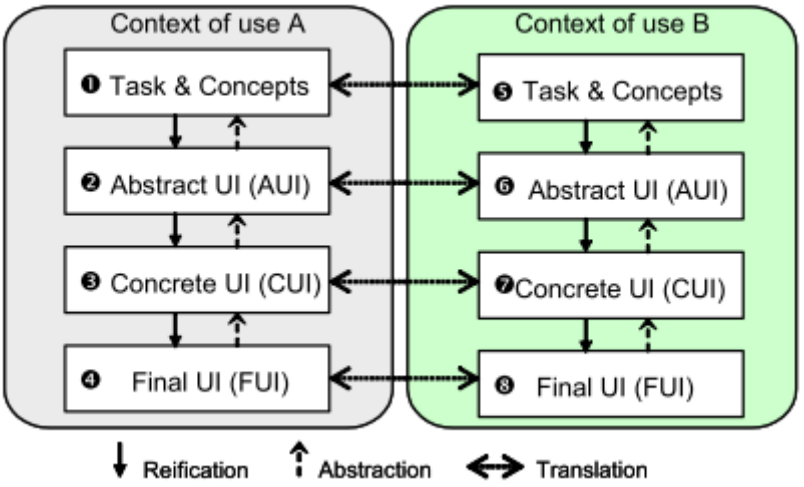

Figure 1 - Les différents chemins de transformation (Limbourg & Vanderdonckt 2004)

la(les) plateforme(s) ciblée(s).

L'IHM ainsi générée est adaptée au contexte d'usage pour lequel elle a été calculée. Mais si un changement survient dans ce contexte, il est nécessaire de recalculer la solution pour prendre en compte les nouvelles contraintes. Cette adaptation peut alors être faite en exécutant de nouveau toutes les transformations, ou en recalculant la solution à l'un des niveaux d'abstraction. On parle dans ce dernier cas de translation d'un contexte d'usage à un autre.

La plasticité des IHM repose ainsi sur des modèles et des transformations : c'est une approche basée sur les modèles (Paternò 2000). Malgré leur puissance, les approches d'ingénierie dirigée par les modèles pour le développement d'interfaces homme-machine ne sont pas encore largement diffusées dans l'industrie informatique. La section suivante propose une explication à cet état de faits.

1.2 Besoin de guidage adapté pour la mise en œuvre de la plasticité

Selon B. Myers, les approches basées sur les modèles présentent un coût d'accès (« *threshold of use* ») élevé (Myers et al. 2000). Dans l'acception de B. Myers, le coût d'accès est la difficulté à apprendre à utiliser un système. Selon l'auteur, les systèmes couronnés de succès présentent soit un faible coût d'accès et un apport faible (« *low ceiling* »), soit un coût d'accès élevé et un apport important. Selon son analyse, la motivation principale pour mettre en œuvre des approches basées sur les modèles dans le développement d'IHM est l'espoir de diminuer l'effort de développement (des parties des IHM sont générées automatiquement). D'autres bénéfices attendus, présentés comme secondaires, sont la portabilité automatique sur de nombreuses plateformes et la génération automatique des systèmes d'aide pour l'utilisateur final.

B. Myers note que le paradigme des approches basées sur les modèles est peu connu. Sa mise en œuvre nécessite l'apprentissage d'un nouveau langage pour spécifier les modèles, apprentissage qui entraîne un coût d'accès élevé. Face à la difficulté d'apprendre une nouvelle technique, la réponse classique, proposée en Systèmes d'Information et en Génie Logiciel, est de fournir du guidage aux concepteurs et aux développeurs, sous la forme d'une méthode de développement.

Cependant, plusieurs études comme (Hardy et al. 1995), (Fitzgerald 1998), (Avison & Fitzgerald 2003), (Mannaro et al. 2004) ou (Garzotto & Perrone 2007) montrent que les concepteurs et les développeurs sont peu satisfaits des méthodes de développement qu'ils utilisent. Lorsqu'on les interroge, ils estiment en effet que les méthodes (a) ne couvrent pas suffisamment la diversité des projets et des contraintes imposées par les clients, (b) sont difficiles à apprendre et à mettre en œuvre et (c) imposent des processus complexes, linéaires et rigides, rédigés en termes peu accessibles. Offrir une méthode complexe et inadaptable ne permettrait guère de simplifier l'accès au développement d'IHM plastiques.

Nous nous trouvons donc ici face à une difficulté : comment offrir un guidage adapté aux concepteurs et aux développeurs, alors qu'ils perçoivent les guidages existants comme peu adaptés ?

L'ingénierie des méthodes de développement logiciel s'est depuis longtemps préoccupée de l'adaptation des méthodes. Cette capacité d'adaptation à la situation est classiquement nommée flexibilité (Boehm et al. 1975) (Humphrey & Kellner 1989) (Harmsen et al. 1994) (Rising & Janoff 2000) (Mulyar et al. 2007) (Hug et al. 2008). Ces travaux ont porté sur différents niveaux et formes de flexibilité, comme :

- la construction monolithique du produit final, qui sous-entend la mise en œuvre d'un processus rigide (Mills 1971), linéaire et séquentiel (Prakash 1997) ;
- l'adaptation à la maturité des besoins (Boehm 1989) et à leur « *volatilité* » (Boehm 1979), c'est-à-dire leurs variations au cours du projet, qui a amené des auteurs comme (Norman & Draper 1986) ou (Curtis et al. 1987) à interroger les différentes

possibilités de prise en compte des utilisateurs finaux dans le processus de développement ;
- la portabilité (Goldberg 1974) et l'évolutivité (Parnas 1978) du produit final ;
- la modularité du produit final et la réutilisabilité de ses composants (Henderson-Sellers & Edwards 1990) ;
- la structure itérative (Mills 1971), (Basili & Turner 1975), (Dowson 1987) et/ou incrémentale (Glass 1969), (Cockburn 1995) du cycle de développement, son organisation autour de prototypes (Griffiths et al. 2001) et de points de validation (Boehm 1976) ;
- la durée variable du cycle (Messager Rota 2009), (Beck 1999b), (Martin 1991) ;
- l'adaptation à la taille (Boehm et al. 1975) et aux contraintes (Humphrey & Kellner 1989) du projet ;
- la prise en compte de la taille de l'équipe (Cusumano & Selby 1997), (Rising & Janoff 2000), (Laanti 2008) ;
- le partage de connaissances entre équipes de développement et l'évolution des modèles de processus au fil du temps (Smith 1971) ;
- la réutilisabilité des composants méthodologiques (Harmsen et al. 1994), (Guzelian 2007), (Bendraou et al. 2007), (Hug 2009) ;
- les différentes orientations possibles des approches (Rolland 2005), (Gonzalez-Perez & Henderson-Sellers 2008) ;

Pourtant, des études comme (Barry & Lang 2001), (Kalay 2006), (Garzotto & Perrone 2007) ou (Guzman et al. 2012) identifient toujours une demande pressante de flexibilité de la part des équipes de développement ou des critiques sur la difficulté de prise en main des méthodes. Il reste donc des verrous dans la flexibilité des méthodes, comme le manque de choix parmi les activités proposées (Stolterman 1992) (Gonzalez-Perez & Henderson-Sellers 2008), (Martínez-Ruiz et al. 2013). Les concepteurs et les développeurs mentionnent aussi la difficulté d'apprendre le processus proposé (Introna & Whitley 1997), (Guzman et al. 2012) ou d'assimiler un vocabulaire spécifique (Fitzgerald et al. 2003), en particulier lorsque les méthodes ne comportent pas (ou trop peu) d'exemples (Pohl 2010) réalistes (Barry & Lang 2001), (Garzotto & Perrone 2007) et intuitifs (Guzman et al. 2012). Des auteurs comme (Hug et al. 2008) ou (Gonzalez-Perez & Henderson-Sellers 2008) estiment aussi qu'un modèle de processus centré sur les activités, qui relègue les autres points de vue (les buts, les décisions, les artefacts,…) au second plan ne correspond pas aux besoins des organisations qui les mettent en œuvre.

Dans cette thèse, notre question de recherche est donc :

> Comment permettre, à la conception comme à l'exécution, toutes les dimensions de la flexibilité au sein des modèles de processus, en particulier pour le développement d'interfaces homme-machine plastiques.

Cette question se subdivise en (1) comment évaluer les formes de flexibilité offertes par un modèle de processus ou le potentiel de flexibilité apporté par un métamodèle de processus, (2) quelle est la structure nécessaire pour définir des modèles de processus

flexibles et (3) quel modèle de processus flexible est applicable au développement d'interfaces homme-machine plastiques ?

La section suivante présente la méthode de recherche que nous avons mise en œuvre dans cette thèse.

1.3 Méthode de recherche

Nous avons basé nos travaux sur une approche hypothético-déductive (Willig 2001). Cette approche repose sur un cycle alternant des observations à partir desquelles une hypothèse (un modèle, par exemple) est construite par induction. Cette hypothèse est alors utilisée pour faire des prédictions sur les résultats de futures observations : c'est la phase de déduction. Ces prédictions sont ensuite confrontées à des observations concrètes. Les éventuels écarts entre les prédictions et les résultats réels des observations permettent de procéder à un raffinement des hypothèses et d'entrer ainsi dans un nouveau cycle d'induction et de déduction.

Concrètement, nous avons mis en œuvre de tels cycles dans trois phases de nos travaux :

1) lors de la phase d'analyse des modèles de processus et de construction de la taxonomie des modèles de processus. Il s'agissait ici, en partant d'observations sur les structures des différents modèles, d'élaborer des hypothèses sur les critères permettant de caractériser ces structures, puis, par déduction, de confronter ces critères avec des classifications de modèles de processus et ainsi de vérifier si ces critères permettaient réellement de catégoriser les différents modèles ;
2) pendant l'élaboration de notre métamodèle de processus flexibles : en partant des critères qui caractérisent la flexibilité d'un modèle de processus, nous avons élaboré un métamodèle, que nous avons instancié en modèle de processus pour le développement d'IHM plastiques, afin d'observer la flexibilité obtenue ;
3) pendant le développement des outils de conception et d'exécution des modèles de processus : en partant des modèles de processus flexibles élaborés lors de la phase précédente, nous avons défini et implanté une architecture logicielle permettant de représenter et d'exécuter ces modèles. Nous avons ensuite confronté en particulier l'exécution de ces modèles aux formes de flexibilité attendues.

La section suivante présente le plan de cette thèse.

1.4 Plan de la thèse

Nous avons structuré ce document en trois parties.

Dans la première partie, nous présentons notre étude des besoins des concepteurs et des développeurs en flexibilité. Nous montrons ensuite qu'aucune taxonomie existante ne définit les critères concrets permettant de mesurer la flexibilité apportée par un modèle de

processus en réponse aux besoins des concepteurs et des développeurs. Nous proposons donc Promote, notre propre taxonomie, que nous mettons en œuvre pour évaluer la flexibilité de douze modèles et métamodèles de processus représentatifs des approches existantes. Nous montrons ainsi qu'aucun d'entre eux n'offre toute la flexibilité souhaitée.

Ce constat nous amène, dans la seconde partie, à proposer M2Flex, un métamodèle de processus qui permet de créer des modèles intégrant toutes les formes de flexibilité. Afin de faciliter la mise en œuvre de ce métamodèle, nous présentons D2Flex, un outil de conception de modèles de processus, et R2Flex, un outil qui permet l'exécution des modèles définis dans D2Flex.

Nous présentons dans la troisième partie l'utilisation de M2Flex pour la création d'un modèle de processus flexible dédié à la conception d'interfaces homme-machine plastiques. Ce modèle de processus repose sur la méthodologie proposée par le Consortium UsiXML (UsiXML Consortium 2011), que nous augmentons pour intégrer les différentes formes de flexibilité. Nous présentons aussi l'environnement de support pour la mise en œuvre concrète de la flexibilité, ainsi que la valorisation de notre approche.

Un rappel des contributions est proposé en conclusion avec une ouverture sur de nombreuses perspectives.

Première Partie

Première Partie

La première partie de cette thèse se focalise sur la flexibilité attendue par les concepteurs et des développeurs et la flexibilité offerte par les modèles de processus existants.

Elle comporte les chapitres suivants :

- Le second chapitre présente l'analyse des besoins des concepteurs et des développeurs et l'étude des taxonomies existantes sur la caractérisation des modèles de processus, dans l'objectif d'identifier une taxonomie permettant de mesurer les réponses apportées par un modèle de processus aux besoins des concepteurs et des développeurs.
- Le constat qu'il n'existe pas de taxonomie permettant de qualifier la flexibilité d'un modèle de processus au regard des attentes des concepteurs et des développeurs nous amène à présenter Promote, la taxonomie que nous avons créée, dans le chapitre trois. Dans ce chapitre, nous présentons aussi les expériences menées pour l'évaluation de Promote et un site Internet qui outille cette taxonomie.
- Le quatrième chapitre présente l'application de Promote à douze modèles et métamodèles de processus représentatifs des différentes approches existantes. Nous montrons qu'aucun de ces modèles ne répond de façon satisfaisante aux besoins des concepteurs et des développeurs tels que nous avons identifiés dans le chapitre 2.

2 FLEXIBILITE DANS LES MODELES DE PROCESSUS : ETAT DE L'ART EN BESOINS ET TAXONOMIES

Ce chapitre a pour objectif d'analyser les formes de flexibilité souhaitées par les concepteurs et les développeurs au sein des modèles de processus, à la conception comme à l'exécution. Nous étudions aussi les taxonomies existantes afin d'identifier les critères qu'elles proposent pour évaluer les différentes formes de flexibilité dans les modèles de processus. La première section est consacrée aux définitions des termes que nous utilisons dans cette thèse.

2.1 Définitions

2.1.1 Méthode

G. Booch définit une méthode comme un « *processus discipliné pour générer un ensemble de modèles qui décrivent, à l'aide d'une notation bien définie, les divers aspects d'un système logiciel en cours de développement* » (Booch 1993). Selon F. Harmsen, une méthode est une « *collection intégrée de procédures, de techniques, de descriptions de produits et d'outils qui offre un soutien efficace, efficient et cohérent au processus d'ingénierie* » (Harmsen 1997). Ces deux définitions se rejoignent pour considérer qu'une méthode comporte une démarche structurée, ce qui nous amène à retenir cet élément dans notre définition. A la démarche structurée, G. Booch ajoute un ensemble de modèles, qui peuvent être considérés comme le langage de description des produits de la définition de Harmsen. Nous reprenons ce concept sous le terme de modèle de produits. Enfin, nous considérons que les *outils* de la proposition de F. Harmsen sont une généralisation de la *notation bien définie* incluse dans la définition de G. Booch. Nous reprenons cette notion en parlant de collection d'outils. A partir de ces deux définitions, nous retenons donc la définition suivante :

> Une méthode est un triplet composé d'un modèle de processus, d'un modèle de produit et d'une collection d'outils.

2.1.2 Modèle de processus

Nous retenons le terme de *modèle de processus* car il nous semble mieux représenter ce sur quoi nous souhaitons nous concentrer. Aux côtés de *processus discipliné* et de *collection de procédures*, que l'on trouve dans les définitions de Booch et de Harmsen, on peut trouver dans la littérature de nombreuses autres dénominations, comme *processus de développement* (McDermid & Ripken 1984), *modèle de développement* (Rolland 1996), *processus logiciel* (Boehm 1986) ou *processus d'ingénierie logicielle* (Osterweil 1986), qui mettent en exergue l'une ou l'autre facette du concept. Nous ne souhaitons pas retenir le terme de processus seul, car il donne l'impression que l'on décrit directement ce que font les

concepteurs et développeurs, alors qu'il s'agit en réalité d'une représentation abstraite de ces activités, comme le note Osterweil : « *un processus est un support pour réaliser une tâche, une description de processus spécifie comment la tâche doit être réalisée* » (Osterweil 1986). Le terme de *modèle de développement* fait perdre cette notion de processus. Le terme de *modèle de processus*, mentionné initialement dans (Kolodner et al. 1985), présente le double avantage de rappeler que la démarche proposée est un modèle de la démarche qui sera finalement réalisée, et que cette démarche se présente comme un processus, c'est-à-dire comme « *l'enchaînement ordonné de faits ou de phénomènes, répondant à un certain schéma et aboutissant à quelque chose* » (Larousse).

Kroeger et Davidson définissent un modèle de processus comme « *la description d'activités partiellement ordonnées, avec des rôles associés, des actifs sous-jacents et des artefacts en entrée et en sortie, destinés à produire ou à améliorer des logiciels dans un contexte déterminé* » (Kroeger & Davidson 2009). Cette définition présente l'inconvénient d'imposer les activités comme l'élément structurant du modèle de processus, alors qu'il existe d'autres possibilités : les produits, les stratégies, le contexte... (Rolland 2005). C'est pourquoi nous préférons proposer la définition suivante :

> Un modèle de processus est la description structurée et ordonnée d'une démarche de conception et de développement, qui orchestre, à l'aide d'opérateurs, des éléments comme des activités, des artefacts et des rôles.

2.1.3 Rigidité des modèles de processus

Dans la littérature, les modèles de processus ont d'abord été qualifiés de rigides lorsqu'ils conduisaient à construire un système lui-même perçu comme rigide. Différents facteurs ont été identifiés pour expliquer cette rigidité : (Boehm 1979) l'impute aux études précoces et figées des besoins des utilisateurs, selon lui utopiques car ne prenant pas en compte les 25% de besoins qui changent en moyenne au cours d'un projet. (Mills 1971) l'associe à l'absence de structure itérative et (Glass 1969) au manque de structures incrémentales, tandis que (Griffiths et al. 2001) incrimine la carence en prototypage précoce, et (Boehm 1976) l'insuffisance de points de validation. Selon (Parnas 1978), le problème est plus général : les équipes de développement ne sont pas suffisamment guidées pour créer des systèmes modulaires et extensibles.

La rigidité n'est cependant pas seulement interrogée sous l'aspect du produit final. Elle porte aussi sur la structure propre du modèle de processus, et la nécessité qu'il puisse s'adapter à la situation locale (Brinkkemper et al. 1998, Karlsson & Ågerfalk 2004, Fitzgerald et al. 2003, Rossi et al. 2004). Les modèles de processus rigides sont alors ceux pour lesquels cette adaptation est complexe, soit parce qu'ils sont complètement prédéfinis (Rolland 1998), soit par manque de modularité (Mirbel & Ralyté 2006), soit parce qu'ils sont linéaires et

séquentiels, ce qui impose une série d'étapes par lesquelles il faut obligatoirement passer (Prakash 1997). On peut alors avoir un modèle de processus qui, bien que conçu pour produire un logiciel adaptable, est cependant lui-même rigide.

Nos travaux portent sur la rigidité de la structure du modèle de processus. En nous inspirant de (Glass 1969), nous considérons donc qu'un modèle de processus est rigide lorsque sa structure est réfractaire aux changements, c'est-à-dire lorsqu'elle ne permet pas d'être adaptée localement aux besoins.

Nous apportons cependant une nuance à cette définition, qui ne spécifie pas si la flexibilité attendue se situe au moment de la conception du modèle de processus ou au moment de son exécution. Les approches de construction situationnelle de méthodes (Brinkkemper et al. 1998, Ralyté et al. 2003, Bucher et al. 2007) proposent ainsi une forte flexibilité à la conception, puisque le modèle de processus est défini spécifiquement pour le projet dans lequel il est ensuite mis en œuvre. L'adaptation locale aux besoins est donc censée être prise en compte. Cependant, nous notons que la définition du modèle de processus est faite avant le démarrage du développement, ce qui requiert d'anticiper les besoins du projet. Or, les besoins sont changeants (Boehm 1979) : ceux des utilisateurs, mais aussi, par induction, ceux du projet. Nous considérons donc que les modèles de processus conçus en amont de la phase active du projet et ensuite appliqués dans le projet risquent, au fil du temps, de s'avérer inadaptés à la situation changeante.

Pour contrer ce potentiel de rigidité, des auteurs comme (Rolland 2005) ou (Guzelian 2007) proposent de construire le modèle de processus « *à la volée* », par exemple en s'appuyant sur un ensemble de composants ou de services méthodologiques assemblés dynamiquement, l'assemblage pouvant évoluer au fil du temps. Cependant, ces composants ou ces services doivent (1) avoir été identifiés préalablement dans des modèles de processus existants et (2) avoir été alignés sémantiquement pour assoir leur compatibilité (Hug 2009). L'équipe de développement n'a donc pas toute la liberté d'orienter son processus comme elle l'entend : elle dépend d'une personne capable d'enrichir le dictionnaire de composants ou de services. C'est ce que nous nommons la rigidité à l'exécution. Nous définissons donc :

> Un modèle de processus est dit rigide à l'exécution lorsque sa structure ne permet pas son adaptation dynamique par l'équipe qui le met en œuvre.

2.1.4 Flexibilité des modèles de processus à l'exécution

Dans l'objectif de permettre la création de modèles de processus flexibles, en particulier dans le cadre du développement d'IHM plastiques, nos travaux portent sur l'éviction de la rigidité des modèles de processus à l'exécution, c'est-à-dire sur la flexibilité des modèles de processus à la conception aussi bien qu'à l'exécution, que nous définissons comme :

> La flexibilité d'un modèle de processus est sa capacité plus ou moins importante à être adapté, par ceux qui l'utilisent et pendant son exécution, au contexte dans lequel il est mis en œuvre.

Tout comme un matériau physique peut être flexible dans certaines dimensions spatiales et pas dans d'autres, un modèle de processus peut être flexible d'une façon et pas d'une autre. Ainsi, un modèle de processus pourrait s'adapter à la taille du projet (en proposant des activités optionnelles qui ne seraient réalisées que pour des grands projets), mais pas au domaine traité : un modèle de processus pourrait être adéquat pour créer un système d'information mais pas pour créer un logiciel embarqué. Ces considérations nous amènent à parler de formes de flexibilité.

2.2 Analyse des besoins des concepteurs et des développeurs

Selon (Stolterman 1994), « *les approches rigides*[2] *partent de l'hypothèse qu'à l'inverse des concepteurs et développeurs, les modèles de processus sont rationnels* » et qu'il convient donc de contraindre les équipes à structurer leur travail. Ces approches reposent ainsi sur l'espoir que les projets (voire les clients) pourront être forcés à se conformer aux « *prescriptions du modèle de processus* » (Bell 1981). Pourtant, L. Osterweil a mis en évidence qu'un modèle de processus est une entité statique, généralement conçue à partir de projets réels, qui sont des entités dynamiques au cours desquelles les ingénieurs imaginent comment résoudre un problème. Le modèle de processus est ensuite reproduit dans d'autres projets, qui se retrouvent être « *des entités dynamiques contraintes par une entité statique* » (Osterweil 1986). Le risque est alors important de contraindre la créativité, pourtant nécessaire pour identifier la meilleure solution possible (Booch 1993), et d'imposer un carcan, constitué de solutions prédéfinies et pourtant potentiellement mal adaptées. D'après C. Gonzalez-Perez et B. Henderson-Sellers, ce paradigme prescriptif peu adaptable entraîne alors le rejet des méthodes par ceux qui devraient les utiliser : ils estiment que l'utilisation d'un modèle de processus n'a de sens que si ce dernier s'adapte aux besoins leurs « *utilisateurs et bénéficiaires ultimes* », c'est-à-dire les concepteurs et les développeurs (Gonzalez-Perez & Henderson-Sellers 2008). Il nous paraît donc nécessaire

[2] Voir définition en section 2.1.3

d'analyser les besoins des concepteurs et des développeurs pour pouvoir proposer des modèles de processus permettant les adaptations qu'ils souhaitent. En d'autres termes, nous nous proposons d'étudier les besoins des concepteurs et des développeurs pour définir la flexibilité, ou, plus précisément, les formes de flexibilité, que nous incorporerons dans les modèles de processus.

Ce chapitre est consacré à l'étude de ces besoins. Il est structuré de la façon suivante : nous commençons par expliciter la démarche que nous avons mise en œuvre, puis nous présentons notre analyse des besoins des concepteurs et développeurs à partir d'un ensemble de publications.

2.3 Méthode de recherche

L'objectif de cette première partie de notre travail est d'étudier la flexibilité des modèles de processus en réponse aux besoins exprimés par les concepteurs et les développeurs. Notre démarche a donc été d'explorer la littérature pour identifier les analyses et études portant sur les besoins des concepteurs et développeurs, soit directement par des enquêtes, soit indirectement à travers l'analyse des manques dans les modèles de processus.

Nous avons collecté plusieurs dizaines de publications sur traitant de la flexibilité des modèles de processus, comme (Boehm 1986), (Humphrey & Kellner 1989), (Stolterman 1992), (Hardy et al. 1995), (Sommerville 1996), (Kabbaj et al. 2008) ou (Agerfalk & Fitzgerald 2006). Notre premier travail a consisté à sélectionner celles d'entre elles qui portaient sur les besoins des équipes de développement. Nous avons ainsi identifié trois études qui consacrent de larges sections aux besoins des concepteurs et des développeurs : (Fitzgerald 1998), (Barry & Lang 2001) et (Garzotto & Perrone 2007). Ces trois études sont complémentaires : (Fitzgerald 1998) et (Barry & Lang 2001) donnent des résultats chiffrés sur les questions qu'ils étudient, mais comme leur objectif principal est d'analyser la façon dont sont mises en œuvre les méthodes de développement et leur adoption, les questions sur les besoins des concepteurs et des développeurs sont parfois peu détaillées. A l'inverse, l'étude de (Garzotto & Perrone 2007), qui porte sur l'acceptabilité des méthodes de développement, porte un regard sensiblement plus aigu sur ces questions et analyse de façon détaillée les besoins exprimés. La principale limite de cette étude est qu'elle porte sur les processus de développement pour le Web. Cependant, les remarques et besoins exprimés par les professionnels interrogés ne sont pas spécifiques à ce domaine. Ainsi, quand ils expriment vouloir « *se sentir libres de leurs choix* », ils ne parlent pas de développement Web, mais des contraintes et des prescriptions imposées par les modèles de processus qu'ils utilisent.

Ces trois études sont complétées par d'autres publications, comme (Hess 1980), (Osterweil 1986), (Humphrey & Kellner 1989), (Curtis et al. 1987) ou (Hardy et al. 1995) qui, sans s'intéresser directement aux besoins des concepteurs et développeurs, apportent des arguments complémentaires sur certains points.

Nous avons donc estimé que l'ensemble de ces publications est intéressant à considérer et permet d'obtenir une vision d'ensemble des critiques, des attentes et des besoins des concepteurs et des développeurs. Nous avons ensuite analysé les publications retenues pour identifier les arguments critiques et les besoins exprimés par les concepteurs et les développeurs ou synthétisés par les auteurs. Nous avons ensuite regroupé les différents besoins par catégorie, en nous basant sur leurs similitudes. Ces différentes catégories sont présentées ci-dessous.

2.4 Analyse des besoins des concepteurs et des développeurs lors de l'exécution des modèles de processus

2.4.1 Besoin de chemins variés et de possibilités d'évolution

D'après (Booch 1993), un trop fort respect des préconisations du modèle de processus peut peser sur la créativité et sur l'exploration de solutions alternatives et finalement limiter la dynamique créative (Curtis et al. 1987, Hess 1980), qui est un facteur clé de réussite d'un projet (Fitzgerald 1998). Une application trop stricte du modèle de processus peut aussi conduire à une résistance à l' *« identification précoce des difficultés* » (Humphrey & Kellner 1989), de peur que ces problèmes soient attribués à « *la paresse et à la négligence des utilisateurs et des ingénieurs* » (Bell 1981).

Dans la pratique, les équipes de développement refusent de « *suivre aveuglément un processus* » (Fitzgerald 1998) et sont amenées à transgresser les recommandations du modèle de processus (Osterweil 1986, Potts 1989), en raison des contraintes des projets (Hardy et al. 1995), des préférences des concepteurs et des développeurs (Lanza et al. 2005) ou des lacunes dans les modèles de processus (Hardy et al. 1995). (Fitzgerald 1998) rapporte ainsi que 58% des informaticiens ne respectent pas toujours les prescriptions des modèles de processus. Ainsi, elles évitent de réaliser des activités jugées inutiles (Barry & Lang 2001), comme des études de risques sur des projets jugés peu ou pas risqués, ou trop consommatrices de temps ou de ressources (Garzotto & Perrone 2007), comme la séparation de la conception générale et de la conception détaillée dans des domaines qu'ils estiment bien maîtriser. Lorsque plusieurs entreprises collaborent, les modèles de processus sont aussi modifiés pour convenir à toutes les parties prenantes (Kalay 2006) et permettre la diffusion d'informations sous des formes diverses (Lee 2010).

Les choix des parties mises en œuvre ou modifiées par la ou les équipes sont alors purement subjectifs et basés sur leur expérience ; les concepteurs et développeurs se revendiquent en effet comme compétents pour faire ces choix et expriment une « *certaine irritation* » face aux approches trop prescriptives (Stolterman 1992). D'après (Fitzgerald 1998), les équipes de développement voudraient que les modèles de processus les laissent plus largement exercer leur jugement pour décider de ce qui est approprié, ce qui rejoint la proposition de (Yourdon 1999), qui suggère d'appliquer uniquement les recommandations que l'équipe de développement estime les meilleures. T. Martinez-Ruiz propose aussi de définir des modèles de processus avec de nombreux variants, afin que l'équipe puisse choisir les propositions les mieux adaptées (Martínez-Ruiz et al. 2013).

Cependant, laisser l'équipe choisir entre plusieurs options prédéfinies n'offre encore qu'une adaptation partielle (Harmsen 1997), puisqu'il n'y a pas de garantie que l'une des options soit adaptée aux besoins. J. Bach propose donc la mise en œuvre de processus dynamiques, qui « *changent avec la situation* » (Bach 1995). Yourdon propose que, dans certains cas, les modèles de processus soient amendés localement pour correspondre aux besoins (Yourdon 1999). M. Kabbaj suggère de laisser les concepteurs et développeurs sortir du modèle de processus et utiliser des « *déviations* » (Kabbaj et al. 2008), c'est-à-dire des actions qui ne correspondent pas à ce qui est proposé par le modèle de processus. Les déviations correspondent à des changements dans le modèle de processus pendant son exécution, changements qui peuvent prendre la forme d'ajouts, de modifications ou de suppressions d'éléments.

Nous retenons de ces arguments qu'un modèle de processus flexible doit comporter des mécanismes pour permettre aux concepteurs et aux développeurs (1) de faire des choix parmi plusieurs propositions et (2) de pouvoir modifier le modèle de processus pendant son exécution, pour prendre en compte des possibilités perçues comme mieux adaptées.

2.4.2 Besoin de capitalisation-réutilisation

Une des critiques principales faite par les ingénieurs à l'encontre des approches académiques est leur manque de documentation, en particulier le manque d'exemples « *adaptés et intuitifs* » (Guzman et al. 2012) et d'études de cas complètes et réalistes, dont les concepteurs et développeurs pourraient s'inspirer (Barry & Lang 2001, Garzotto & Perrone 2007, Fitzgerald et al. 2003). La présence de tels exemples est par ailleurs un point fort mis en avant par les auteurs, comme (Pohl 2010), qui explicite le besoin et l'intérêt de ces exemples. (Paulk et al. 1993) estime d'ailleurs que toute description d'une pratique importante devrait inclure des exemples.

Mais les exemples, s'ils permettent une meilleure compréhension, ne suffisent pas pour faciliter l'accomplissement des tâches. La réutilisation de composants logiciels apporte un complément intéressant : des auteurs comme (Selby 2005) ou (Krueger 1992) considèrent qu'elle est un levier important pour l'amélioration de la productivité et de la qualité des logiciels. R. France et B. Rumpe estiment que la réutilisation de composants permet en outre la capitalisation de l'expérience, ce qui est selon eux particulièrement indiqué dans le cas des développement complexes, comme l'ingénierie dirigée par les modèles (France & Rumpe 2007). La réutilisation apporte de la flexibilité : le développeur a le choix entre créer une solution originale et s'appuyer sur une solution (ou un embryon de solution qu'il doit adapter et enrichir) déjà existante. Il peut ainsi économiser des efforts dans la création d'une solution complète et il peut être amené à explorer des solutions inconnues.

A un plus haut niveau d'abstraction, les patrons de conception proposent aussi de réutiliser des solutions éprouvées (Gamma et al. 1995) et permettent d'économiser une partie de l'effort nécessaire pour inventer la solution à un problème récurrent.

M. Rossi note qu'il faut conserver les traces et les justifications de ces modifications apportées au modèle de processus (Rossi et al. 2004). Le premier objectif est d'identifier les contextes dans lesquels chaque méthode a été mise en œuvre et d'argumenter pour l'utilisation de cette même méthode dans des contextes similaires. Le second objectif d'enrichir le modèle de processus avec les modifications qui ont contribué aux succès des projets et ainsi de partager l'expérience entre équipes de développement, qui est une troisième forme de réutilisation.

Un modèle de processus flexible doit ainsi (3) permettre de pouvoir s'inspirer d'exemples, d'études de cas, ou de composants logiciels, conceptuels ou procéduraux préexistant. Il doit aussi permettre la capitalisation de ces éléments.

2.4.3 Besoin de différentes formes d'information

D'après L. Introna et E. Whitley, la mise en œuvre d'un modèle de processus complexe détourne l'attention de ses utilisateurs, qui doivent plus se concentrer sur l'apprentissage de la méthode que sur la tâche qu'ils ont à résoudre (Introna & Whitley 1997). D'après (Guzman et al. 2012), le manque de documentation destinée aux professionnels ou la difficulté d'apprendre une approche sont des problèmes mentionnés par les ingénieurs logiciels. La facilité d'apprentissage d'une méthode est donc un facteur clé de réussite (Cockburn & Highsmith 2001), la nouveauté étant perçue comme une difficulté à surmonter (Fichman & Kemerer 1993).

Pourtant, d'après les études menées auprès de concepteurs et développeurs, la plupart des modèles de processus ne semblent pas avoir privilégié la facilité d'apprentissage : dans l'étude menée par F. Garzotto et V. Perrone auprès d'une quarantaine d'entre eux, 80% des participants désirent que les modèles de processus soient plus faciles à apprendre (Garzotto & Perrone 2007). Ils estiment en effet que les méthodes sont complexes, qu'elles proposent trop d'informations ou trop de détails (Barry & Lang 2001, Heitmeyer 1998), qu'elles reposent souvent sur un vocabulaire académique mal connu des développeurs et concepteurs (Garzotto & Perrone 2007). Ils souhaitent que les modèles de processus permettent un apprentissage progressif (Fitzgerald 1998) ou au moins une prise en main rapide grâce à un « *sous-ensemble basique* » facilement accessible lorsqu'ils commencent à étudier une méthode, les concepts élémentaires devant pouvoir être transmis facilement (Garzotto & Perrone 2007).

Au delà de la phase d'apprentissage, les concepteurs et développeurs expérimentés continuent à exprimer le besoin que les modèles de processus offrent différents niveaux de détails. En effet, d'après E. Stolterman, les ingénieurs les plus expérimentés trouvent utile une vue synthétique, qui sert de rappel ou de liste de contrôle (Stolterman 1992). Dans le même temps, selon F. Garzotto et V. Perrone, les développeurs et concepteurs apprécient une documentation complète, voire exhaustive, et de haute qualité, orientée vers les problématiques industrielles, quand leur compréhension du problème s'affine (Garzotto & Perrone 2007, Noll 2003) ou lorsqu'ils arrivent sur des phases plus complexes du processus. Ainsi, les modèles de processus doivent offrir, selon les phases du processus, différents niveaux de complexité, de rigueur et de détails. Le modèle de processus

pourrait par exemple proposer des phases de prototypage rapide pour valider l'étude des besoins et la conception avec un guidage léger, suivies si besoin d'une phase d'étude plus poussée accompagnée cette fois par un processus plus riche (Garzotto & Perrone 2007).

Les développeurs et concepteurs attendent aussi qu'une méthode propose différents types de langages, en fonction de l'étape du processus (Fitzgerald et al. 2003), de la culture, des connaissances et du modèle mental individuel de l'ingénieur (Feiler & Humphrey 1993, Garzotto & Perrone 2007, Ambler 2011). Ces vocabulaires variés, qui permettraient différentes formulations, permettraient de répondre à la demande d'intelligibilité notée par C. Barry et M. Lang (Barry & Lang 2001) ainsi qu'à la critique sur l'utilisation d'un vocabulaire parfois trop académique (Garzotto & Perrone 2007).

La réponse à ces différents besoins relève de la flexibilité du modèle de processus, qui doit offrir (4) différents niveaux de détails et (5) différents niveaux de complexité, par exemple sous la forme de différents langages, c'est-à-dire en utilisant de registres de vocabulaire différents, plus ou moins techniques, permettant d'aborder peu à peu les notions les plus complexes.

2.4.4 Besoin de perspectives variées

C. Gonzalez-Perez et B. Henderson-Sellers estiment que, la plupart des modèles de processus étant trop rigides, c'est-à-dire inadaptables localement, et trop prescriptifs, c'est-à-dire laissant peu de place aux choix, ils entraînent une résistance des développeurs, qui se voient « *soumis à l'injonction de quelqu'un qui leur dit quoi faire et quand* », en particulier dans les modèles de processus centrés sur les activités. Ces auteurs font un parallèle entre le processus composé de séquences d'étapes à réaliser et les programmes, qui sont des séries d'instructions à exécuter, ramenant ainsi symboliquement le développeur au rang de « *machine qui exécute les méthodes d'une façon pratiquement mécanique* » (Gonzalez-Perez & Henderson-Sellers 2008). Ils doutent de l'efficacité d'une amélioration des processus, car, selon eux, cette quête a conduit à la définition de processus complexes qui « *ont plus souvent augmenté la distance entre les personnes et les processus qu'amené une amélioration du produit final* ».

Ils proposent donc de changer de perspective et d'adopter une vision centrée sur les produits, considérés comme plus tangibles, partageables et visibles que les processus. Ils estiment qu'en demandant aux développeurs d'atteindre l'objectif de fabriquer un produit au lieu de chercher à leur imposer une procédure pour cette fabrication, leur résistance à l'adoption des méthodes pourrait être amoindrie. Concrètement, il s'agit de mettre en œuvre une approche itérative et déductive qui consiste à répéter la recherche des produits nécessaires pour obtenir facilement et directement le produit souhaité, en partant du produit final à construire. Cependant, comme il faut bien fournir de l'aide aux développeurs quand ils ne savent pas comment réaliser le produit attendu, les auteurs intègrent aussi des activités, ou plus précisément proposent que des outils extérieurs définissent ces activités (Gonzalez-Perez & Henderson-Sellers 2008). Au final, la proposition comporte donc les mêmes éléments (activités, produits, séquencement,...) que les approches orientées sur les activités. C'est donc bien la perspective sur le processus et

non les « ingrédients » du modèle qui change dans cette approche. Cependant, cette proposition d'une approche centrée sur les produits ne propose à son tour qu'une seule perspective. Or, C. Hug montre qu'un modèle de processus qui ne présente qu'un seul point de vue (une orientation exclusive sur les activités, les produits, les décisions,...) ne correspond pas aux besoins des organisations qui l'utilisent (Hug et al. 2008).

On peut donc retenir (6) qu'il serait intéressant de proposer différentes perspectives sur le modèle de processus. Par exemple, le modèle de processus peut présenter les buts à atteindre, les produits à confectionner ou les activités à réaliser. Ces différentes perspectives permettent ainsi une adaptation à l'expérience du concepteur ou du développeur, qui peut être assez à l'aise avec les notions manipulées dans une présentation centrée sur les produits, ou bénéficier du guidage offert par les activités.

2.4.5 Synthèse des besoins

De ces différentes études, il ressort plusieurs points particulièrement importants aux yeux des concepteurs et des développeurs :

- Le besoin de variantes :
  - le modèle de processus doit (1) **proposer différents chemins ou différents enchaînements**, pour éviter une trop grande linéarité, s'adapter aux caractéristiques des projets, favoriser l'exploration de solutions variées ;
  - le modèle de processus doit (2) **être modifiable durant son exécution**, afin de favoriser l'exploration de solutions qu'il ne propose pas, de permettre la prise en compte de la multiplicité des situations possibles, de répondre aux changements d'orientation du projet et à l'évolution permanente de chaque tâche suite à sa découverte progressive ;
- Le besoin de capitalisation-réutilisation :
  - le modèle de processus doit être illustré par des exemples concrets ou des études de cas réalistes, afin de (3) **proposer la réutilisation de solutions ou d'embryons de solutions existantes et permettre leur capitalisation**, sous la forme de composants logiciels, conceptuels ou procéduraux, afin de soutenir l'exploration de différentes propositions et laisser le choix de repartir de zéro ou de s'inspirer de travaux antérieurs ;
- Le besoin de multi-vues :
  - le modèle de processus doit pouvoir (4) **présenter différents niveaux d'information**, et (5) **différents niveaux de complexité ou différents langages,** afin de convenir aux différentes phases d'un projet et de correspondre aux différents niveaux de connaissances de ses utilisateurs et de permettre ainsi un apprentissage progressif ;
  - enfin, le modèle de processus doit (6) **proposer différentes perspectives sur le processus** (ou être outillé dans ce but), afin de correspondre aux différents modèles mentaux de ses utilisateurs.

Nous allons maintenant étudier les différentes propositions qui permettent d'évaluer la correspondance entre les modèles de processus et les besoins que nous venons d'identifier.

2.5 Taxonomies et échelles de flexibilité des modèles de processus

Nous avons vu que les concepteurs et les développeurs sont peu satisfaits des propositions contenues dans les modèles de processus qu'ils utilisent. L'analyse des besoins que nous avons réalisée dans la section précédente permet d'interroger les modèles de processus afin d'identifier concrètement en quoi ils répondent ou non à ces besoins. Nous avons donc analysé une soixantaine de modèles de processus, comme le Stagewise Model (Bennington 1983)[3], le modèle du Waterfall (Royce 1970), le modèle en Spirale (Boehm 1986), Rapid Application Development (Martin 1991), Scrum (Schwaber 1995), Macao (Crampes 2002), Diane+ (Tarby & Barthet 1996), Feature Driven Development (Palmer & Felsing 2002), Symphony (Hassine et al. 2002), le Rational Unified Process (Kroll & Kruchten 2003), le Work Product Pool (Gonzalez-Perez & Henderson-Sellers 2008).

Cette analyse a montré que d'une part aucun de ces modèles ne répond à l'ensemble des besoins, tels que nous les avons identifiés dans la section précédente, et d'autre part, que les réponses aux besoins sont très variables. Par exemple, les réponses au besoin numéro 1 (disposer de différents chemins) sont très différentes selon les approches :

- le modèle du Waterfall (Royce 1970) repose sur sept étapes qui se succèdent séquentiellement. W. Royce avait prévu qu'en cas de projet développé pour la première fois, le développement soit divisé en deux phases (chacune réalisant les sept étapes du modèle) : un prototype pour les éléments les plus complexes, puis le système complet. Il offrait donc un choix : faire ou non le prototype. Ce modèle répond-il pour autant au besoin numéro 1 ? Dans l'absolu, nous devions répondre oui : deux chemins différents, ce sont déjà *des* chemins différents. On perçoit cependant qu'une seule option ne répond que très partiellement au besoin de disposer de différents chemins.
- Rapid Application Development (Martin 1991) propose plusieurs chemins possibles selon la taille du projet, l'investissement des utilisateurs, les outils mis en œuvre, les contraintes sur la périodicité des livraisons, etc.
- Le Work Product Pool (Gonzalez-Perez & Henderson-Sellers 2008) offre encore plus d'options: l'équipe définit, de façon quasiment récursive et en toute liberté, les artefacts requis pour construire un artefact cible. Le nombre de chemins est illimité, il dépend uniquement de ce que l'équipe évalue nécessaire pour « *construire directement et facilement* » sa cible. Cependant, il s'agit de chemins potentiels, qui ne sont pas spécifiés dans le modèle de processus mais construits dynamiquement par l'équipe. Ces chemins potentiels répondent-ils vraiment au besoin numéro 1 ?

[3] La publication datée de 1983 reprend une présentation orale faite par H. Bennington en 1956.

- Scrum (Schwaber 1995) propose un chemin très général et suggère d'adapter librement l'approche au contexte. Comme dans le Work Product Pool, les options concrètes ne sont pas proposées par le modèle de processus. La liberté de les définir lors de la mise en œuvre de Scrum correspond-elle à une proposition de chemins variés ?

On constate donc qu'il est difficile d'évaluer quels modèles de processus répondent réellement au besoin numéro 1, et même lorsqu'ils proposent concrètement des chemins différents, le besoin numéro 1 n'est pas nécessairement satisfait. Ainsi, nous avons vu que le modèle du Waterfall offre deux chemins différents, alors qu'il est analysé comme particulièrement linéaire et séquentiel (Prakash 1997). Ce constat, qui s'applique aussi aux autres besoins, montre d'une part que la formulation des besoins est ambigüe (un chemin potentiel doit-il être pris en compte ?) et d'autre part qu'un modèle de processus peut répondre plus ou moins au besoin. Le modèle du Waterfall peut être considéré comme répondant très peu au besoin numéro 1. Nous avons étudié les travaux de caractérisation des modèles de processus afin d'identifier les critères qu'ils proposent, et qui pouvaient servir à désambigüiser l'expression des besoins.

Nous avons identifié cinq études portant sur la caractérisation des modèles de processus, que nous présentons ci-dessous. Nous avons organisé cette présentation de la façon suivante : les deux premiers travaux, (Alexander & Davis 1991) et (Sharon et al. 2010) sont prédictifs, ils évaluent si un modèle de processus sera adapté à un projet. Le suivant, (Pérez et al. 1995) évalue l'adéquation entre un modèle de processus et un projet et définit comment adapter le modèle de processus en cas de besoin. Le quatrième, (Mulyar et al. 2007), porte sur la flexibilité par déviation, c'est-à-dire sur la prise en compte des activités réalisées par l'équipe qui ne correspondent pas aux préconisations du modèle. Enfin, (Harmsen et al. 1994) traite de la flexibilité par construction, c'est-à-dire de la construction d'un modèle de processus fait sur mesure pour un projet.

2.5.1 Les critères de Alexander et Davis

L. Alexander et A. Davis proposent 20 critères destinés à évaluer l'adéquation entre les besoins d'un projet et les propositions d'un modèle de processus (Alexander & Davis 1991). Chacun de ces critères est associé à une graduation comportant trois valeurs possibles. Par exemple, le problème à résoudre peut être simple, difficile ou complexe.

Le tableau 1 présente les critères proposés par Alexander et Davis. Deux de ces critères (numérotés et écrits en bleu dans le tableau) concernent l'adéquation du modèle de processus aux besoins de flexibilité exprimés par les concepteurs et les développeurs, les autres critères portant sur les utilisateurs finaux, le système à construire, la disponibilité des ressources et les besoins de l'organisation.

Ces deux critères, à priori pertinents pour notre étude, sont :

(1) Expérience des développeurs dans le domaine applicatif : il s'agit ici de mesurer « *les connaissances des développeurs dans le domaine traité, cette connaissance pouvant provenir soit de l'utilisation antérieure d'une application dans le même*

domaine, soit d'une expérience de développement dans ce domaine ». Ce critère traite donc des compétences de l'équipe et non de l'adaptation du modèle de processus à ces mêmes compétences.

(2) Expérience des développeurs en ingénierie logicielle : ce critère évalue « *l'expérience et la connaissance qu'ont les développeurs à propos des outils, des méthodes, des techniques et des langages requis par le développement* ». Il s'agit donc d'évaluer comment l'équipe pourra s'adapter à la situation et non comment le modèle de processus pourra s'adapter à la situation de cette équipe.

Critère	Val.1	Val. 2	Val. 3
Caractérisation du personnel			
Expérience des utilisateurs dans le domaine	Novice	Expérimenté	Expert
Capacité des utilisateurs à exprimer les besoins	Mutique	Communicant	Volubile
(1) Expérience des développeurs dans le domaine	Novice	Expérimenté	Expert
(2) Expérience des développeurs en ingénierie logicielle	Novice	Expérimenté	Expert
Caractérisation du problème			
Maturité de l'application	Nouvelle	Standard	Classique
Complexité du problème	Simple	Difficile	Complexe
Livraisons partielles	Indésirable	Souhaité	Urgent
Fréquence des changements	Rare	Modérée	Haute
Amplitude des changements	Mineure	Modérée	Extrême
Caractérisation du produit			
Taille du produit	Petit	Moyen	Grand
Complexité du produit	Simple	Difficile	Complexe
Exigences non-fonctionnelles	Souple	Modéré	Exigeant
Besoin en interaction	Mineur	Significatif	Critique
Caractérisation des ressources			
Profil du financement	Croissant	Stable	Décroissant
Quantité des fonds	Limitée	Adéquate	Large
Disponibilité de l'équipe	Croissante	Stable	Décroissante
Taille de l'équipe	Limitée	Adéquate	Grande
Accessibilité des utilisateurs	Aucune	Limitée	Grande
Caractérisation de la compatibilité du modèle de processus avec l'organisation			
Règles de développement en vigueur	Optionnel	Souhaité	Requis
Assurance qualité et capacité de configuration	Basique	Moyenne	Avancée

Tableau 1 - Critères et graduation de Alexander et Davis

Ces critères ne correspondent aux besoins des développeurs et des concepteurs comme nous les avons définis dans la section 2.4 : ils ne mentionnent aucun de ces besoins. Cette taxonomie ne permet donc pas d'évaluer un modèle de processus selon ces besoins, comme nous entendons le faire.

2.5.2 Les scores de Sharon et al.

I. Sharon détermine un certain de nombre de caractéristiques des processus de développement logiciel et établit leur lien avec les particularités des projets, afin, ici aussi, d'évaluer leur adéquation (Sharon et al. 2010). Dans l'analyse des caractéristiques des projets, on retrouve des critères proches de ceux de (Alexander & Davis 1991) et de (Pérez et al. 1995), comme la stabilité des besoins ou la taille du projet. La liste complète des caractéristiques étudiées est présentée dans le tableau 2. Les caractéristiques du projet

sont ensuite confrontées aux évaluations de différents modèles de processus. Pour une caractéristique donnée, l'adéquation entre le modèle de processus et le projet peut être évaluée à 0 (aucune adéquation), 0,5 (adéquation minimale), 0,75 (adéquation moyenne) et 1 (adéquation parfaite). Pour combiner différentes caractéristiques et ainsi évaluer un modèle de processus selon toutes les dimensions, les auteurs calculent la somme de l'évaluation de chaque caractéristique multipliée par le poids attribué à cette caractéristique (tableau 2, colonne 2). Les modèles de processus sont ensuite ordonnés selon leur score, et le modèle de processus qui obtient le meilleur score est celui qui semble le mieux adapté pour le projet. L'adéquation est ici considérée comme statique : une fois le meilleur modèle de processus identifié, il est mis en œuvre dans le projet. La proposition ne comporte pas de phase d'adaptation du modèle de processus, ni de possibilité de réévaluer l'adéquation au cours du projet.

Deux critères mentionnent l'équipe de développement et donc potentiellement ses besoins. Cependant, le « *relationnel dans l'équipe* », qui traite de la qualité de la communication au sein de l'équipe, et la « *taille de l'équipe* », qui dénombre les personnes incluses dans l'équipe, ne correspondent pas aux besoins des concepteurs et des développeurs tels que nous les avons identifiés.

Critère	Poids
Maturité des besoins	4,4
Stabilité des techniques	4,3
Taille du projet	4,1
Identification des risques	3,8
Externalisation	3,5
Clarté de la portée du projet	3,3
Engagement du client	3
(1) Relationnel dans l'équipe	3
(2) Taille de l'équipe	2,9
Type de contrat	2,9
Flexibilité des parties prenantes	2,8

Tableau 2 - Critères de Sharon

Aucun des critères de cette taxonomie ne fait donc référence aux besoins des développeurs et des concepteurs tels que nous les avons analysés. Cette proposition ne permet pas non plus d'identifier le modèle de processus adapté à ces besoins.

2.5.3 Les critères de Pérez et al.

G. Pérez s'est intéressée aux propriétés requises pour évaluer l'adéquation entre un modèle de processus adapté et un projet (Pérez et al. 1995), et ainsi déterminer si le modèle de processus est adapté au projet et, éventuellement, définir la personnalisation requise pour obtenir une meilleure correspondance. Le modèle de processus est alors modifié puis appliqué dans le projet.

Les auteurs proposent de mesurer l'adéquation globale à travers l'analyse de l'adéquation de 21 critères, qui sont présentés dans le tableau 3. Une adéquation se traduit par un nombre compris entre -1 (aucune adéquation) et 1 (adéquation parfaite). Les critères pris en compte pour le modèle de processus sont par exemple « *Dispositifs de compréhension et de validation de fonctionnalités* », « *Mécanisme de coordination* », « *Formalisation de la gestion des changements* », « *Effort d'analyse de l'existant* », etc. Les auteurs présentent des exemples d'adaptation du modèle de processus après la confrontation entre l'étude du modèle de processus et les besoins du projet. Ainsi, lorsque le projet ne permet pas l'utilisation d'outils de prototypage, ils proposent de remplacer ces prototypes par des modèles, utilisés alors comme des dispositifs de compréhension et de validation.

Critère	Exemple de valeur
Critère de sortie	Plus faible que le standard
Répartition des produits / livraisons	Oui
Mécanismes d'intégration	Faible
Dispositifs de compréhension / validation de fonction	Prototypes
Mécanisme de coordination	Livraison
Formalisation de la gestion des changements	Faible
Ressources pour le contrôle et la planification	Forte
Quantité de documentation pour la gestion de projet	Elevée
Importance du contrôle et de la planification	Elevée
Formalisation des vues utilisateur	Simple
Taille du modèle des vues utilisateurs	Importante
Existence de modèles	Quelques-uns
Existence d'exemples	Nombreux
Effort d'analyse du système existant	Elevé
Âge de la méthode	Vieille
Type de modélisation en PA	Fonctionnel et conceptuel
Orientation composants	Composants utilisés
Description du processus dans la méthode	Peu claire
Spécialisation des rôles dans le modèle	Elevée
Différence relative du modèle	Différent
Réalisation incrémentale	Oui

Tableau 3 - Critères de Pérez

Cette proposition présente plusieurs limites par rapport à notre besoin. D'une part, elle ne comporte pas de définition des critères ni de spécification des différentes valeurs possibles pour chaque critère, ce qui rend son application concrète difficile. D'autre part, si les critères proposés mènent bien à l'adaptation (« *customization* ») d'un modèle de processus à un ensemble de besoins locaux, cette adaptation se fait lors d'une phase amont du projet, ce qui rend impossible la modification du modèle pendant son exécution (besoin numéro 2). Enfin, cette taxonomie ne propose pas de critère correspondant aux autres besoins des concepteurs et développeurs. Cette proposition ne correspond donc pas à nos attentes.

2.5.4 La taxonomie et les patrons de Mulyar et al.

N. Mulyar, W. Van der Aalst et N. Russel ont proposé une autre classification de la flexibilité des modèles de processus (Mulyar et al. 2007). Selon eux, on trouve cinq formes de flexibilité.

En premier se trouve **la flexibilité par conception**, qui est définie comme la capacité d'intégrer des chemins de réalisation alternatifs dans la définition d'un processus, au moment de la conception de ce dernier, ce qui correspond au besoin numéro 1 des concepteurs et développeurs : disposer de différents chemins (voir page 18). Le chemin le plus pertinent est alors sélectionné au moment de l'exécution du modèle de processus.

La flexibilité par sous-spécification est la possibilité d'exécuter un modèle de processus incomplètement défini, c'est-à-dire un modèle ne contenant pas assez d'information pour permettre son exécution jusqu'à son terme. Cette forme de flexibilité ne permet pas de modifier les parties existantes du modèle de processus, mais autorise à les compléter pendant l'exécution en fonction du processus qui est concrètement réalisé, ce qui correspond au besoin numéro 2 (modification pendant l'exécution).

La flexibilité par déviation est la capacité pour un processus de dévier pendant son exécution du chemin prescrit par son modèle, sans altérer ce dernier, ce qui ne répond pas au besoin numéro 2 : le modèle de processus n'est pas modifié, les concepteurs et développeurs sont simplement autorisés à ne pas le respecter.

La flexibilité par changement temporaire permet, elle, de modifier le processus pendant l'exécution (comme la forme précédente) mais impacte temporairement le modèle de processus, sans diffusion aux autres instances du modèle. Les modifications du modèle sont alors perdues lorsque le processus se termine. La correspondance avec le besoin numéro 2 est ici assurée, mais pas le besoin numéro 3 : les modifications n'étant pas capitalisées dans le modèle de processus, la réutilisation d'artefacts issus de projets précédents ou de parties de processus (considérées alors comme des patrons de développement) n'est pas prise en compte.

La flexibilité par changement permanent permet de modifier non seulement un processus mais aussi le modèle de processus. Les autres instances du modèle peuvent être abandonnées (le processus concret s'arrête), abandonnées et redémarrées (le processus redémarre, conforme au nouveau modèle), laissées telles qu'elles sont (seules les nouvelles instances utilisent le nouveau modèle), ou migrées dans un état équivalent du nouveau modèle. Cette forme de flexibilité apporte à la fois la possibilité de modifier le modèle de processus pendant son exécution (besoin numéro 2) et la capitalisation des actions menées, qui est l'un des éléments du besoin numéro 3. L'enregistrement des artefacts produits en vue d'une réutilisation n'est pas mentionné dans cette publication.

Ces cinq formes de flexibilité sont considérées comme indépendantes : les auteurs ne considèrent pas qu'il s'agit de formes croissantes de flexibilité, mais qu'au contraire un modèle de processus peut proposer plusieurs formes simultanément. La flexibilité globale

est ici évaluée en fonction du nombre de formes de flexibilité que le modèle de processus comporte.

Cette taxonomie est par ailleurs accompagnée de 34 patrons qui définissent comment mettre en œuvre concrètement la flexibilité dans chacune des catégories. Ils proposent ainsi des solutions pour flexibiliser l'initialisation, la terminaison, l'extension, la contraction et le réordonnancement du processus et de son modèle, ainsi que la sélection, la parallélisation et la répétition d'activités au sein du processus.

La proposition de Mulyar prend en compte l'exécution du modèle de processus : les déviations surviennent et sont traitées pendant l'exécution, les ajouts au processus et éventuellement au modèle de processus se font aussi pendant l'exécution. Cette taxonomie répond ainsi, comme nous l'avons vu ci-dessus, aux besoins numéro 1 (différents chemins) et 2 (modification pendant l'exécution) et, en partie, au besoin numéro 3 (capitalisation des modifications du modèle, mais pas des artefacts produits). Les autres besoins exprimés par les concepteurs et les développeurs (une partie du besoin numéro 3 - la réutilisation de solutions, 4 - différents niveaux d'information, 5 - différents langages et 6 - différentes perspectives) ne sont par contre pas couverts.

2.5.5 Le spectre de Harmsen

D'après la définition proposée par Harmsen (Harmsen et al. 1994), la flexibilité est une notion monodimensionnelle, évaluée selon une échelle comportant cinq degrés. La figure 2 présente le spectre de la flexibilité croissante.

Au niveau le plus faible se situe **l'utilisation de modèles de processus**[4] **rigides**, qui « *reposent sur un ensemble figé de techniques et de procédures de modélisation* » et « *ne laissent aucune place à l'adaptation à la situation* ». Harmsen mentionne en exemple le modèle de processus de Merise (Tardieu et al. 1986). Le modèle du Waterfall (Royce 1970) fait aussi partie de cette catégorie.

Au niveau au dessus, **la sélection de modèles de processus rigides** permet de choisir l'approche la plus adaptée à un projet dans un ensemble de modèles rigides. C'est à ce niveau que se situent par exemple les approches de Alexander et Davis (Alexander & Davis 1991) et de Sharon (Sharon et al. 2010), que nous avons vues dans les sections précédentes. Une fois le modèle de processus choisi, il n'y a plus de possibilité d'adaptation. La flexibilité reste donc limitée. De plus, l'adoption de cette approche par sélection est contrainte par le coût important lié à l'achat des méthodes et des outils, ainsi qu'aux formations requises pour faire un choix éclairé (Kumar & Welke 1992).

La sélection de chemin dans un modèle de processus repose sur l'utilisation d'une démarche qui offre plusieurs possibilités, ce qui correspond au besoin numéro 1 des concepteurs et développeurs : disposer de différents chemins (voir page 18). Un exemple est Rapid Application Development (Martin 1991), qui propose plusieurs adaptations selon

[4] Harmsen ne fait pas la distinction entre "méthode" et "modèle de processus". Pour conserver la cohérence par rapport au reste du document, nous avons transcrit ses termes en modèles de processus

le projet, comme l'utilisation ou non de la « *timebox* » (une durée d'itération fixée qui impose une régularité dans les livraisons), ou la possibilité de regrouper des séances de travail communes entre développeurs et utilisateurs (dans le cycle normal, il y a une série de séances dédiées à l'étude des besoins et une autre série dédiées à la conception. Dans la version alternative, il y a une seule série de séances qui étudient les deux aspects en même temps). Cette approche offre évidemment une forme de flexibilité, celle du choix parmi plusieurs variantes, mais ne répond pas aux autres besoins. Par exemple, la modification du modèle de processus pendant son exécution n'est pas possible.

La sélection et l'adaptation d'un modèle de processus consistent à sélectionner une démarche parmi différentes possibilités et à l'accorder aux besoins du projet. Il ne s'agit pas uniquement de faire des choix parmi différents chemins prédéfinis (besoin numéro 1), mais aussi d'affiner et d'adapter le modèle de processus, ce qui sous-entend, toujours d'après F. Harmsen, l'utilisation d'outils d'adaptation. Les travaux de Pérez (Pérez et al. 1995) se situent à ce niveau du spectre. La correspondance avec les autres besoins des concepteurs et développeurs est difficile à évaluer, car d'une part l'adaptation se situe au niveau d'un projet en général (les besoins des équipes de développement ne sont pas mentionnés) et d'autre part parce que les possibilités d'adaptation dépendent des fonctionnalités offertes par les outils.

La construction modulaire de modèle de processus consiste à assembler des fragments de modèles pour constituer un modèle situationnel, autrement dit un modèle de processus élaboré sur mesure en fonction des besoins du projet. F. Harmsen décrit la construction de la façon suivante : « *les fragments de méthodes sont formatés selon un modèle, sont stockés dans une base méthodologique et sont assemblés en fonction de règles qui guident l'ingénieur des méthodes pour construire une méthode situationnelle efficace, efficiente, complète et cohérente* ». Cette vision se rapproche très fortement de la proposition de K. Kumar et R. Welke (Kumar & Welke 1992). Elle est aussi reprise par d'autres auteurs, comme (Rolland 2005), qui précise que cette approche permet une adaptation aux contingences d'un projet et aux besoins spécifiques d'un groupe d'usagers. Cette approche est classée comme la plus flexible par ces différents auteurs. Cependant,

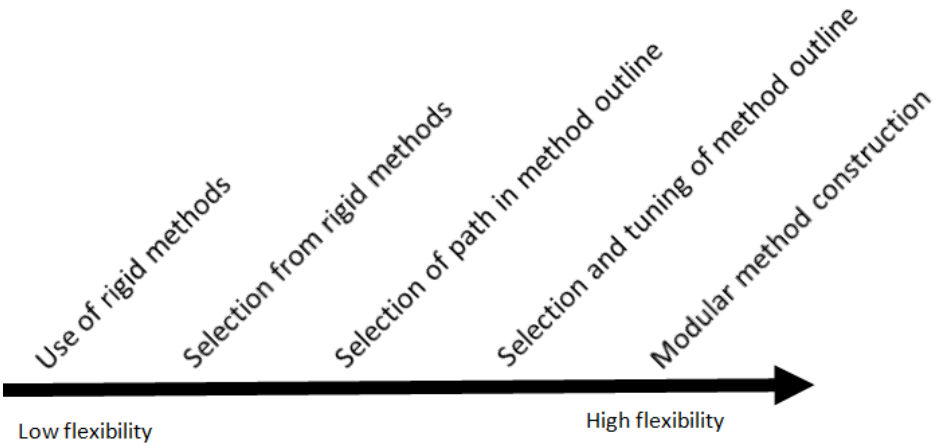

Figure 2 - La flexibilité des modèles de processus selon Harmsen (Harmsen et al. 1994)

même si les modèles de processus situationnels sont construits « *à la volée* » (Rolland 2005), ils sont construits en amont des phases de développement : c'est un *ingénieur des méthodes* qui étudie les besoins du projet et de l'équipe et qui décide de sélectionner tel ou tel fragment qui lui parait pertinent. Si les besoins changent au cours du projet, il est nécessaire de faire de nouveau appel à l'ingénieur des méthodes, et donc de revenir à cette phase de conception du modèle de processus, ce que Rossi nomme la « *disjonction spatio-temporelle de l'ingénierie des* [modèles de processus] » (Rossi et al. 2004).

Même à ce qui est considéré comme un haut niveau de flexibilité, le choix entre les différents chemins n'est donc pas fait par les concepteurs et les développeurs (besoin numéro 1). La modification pendant l'exécution n'est pas envisagée (besoin numéro 2). La réutilisation de solutions (besoin numéro 3), l'offre de différents niveaux d'information (besoin numéro 4) et de différents langages (besoin numéro 5) sont par contre possibles, à condition de pouvoir être anticipées lors de la conception du modèle de processus. (Hug 2009) montre qu'il est aussi possible d'assembler des composants proposant différents points de vue (besoin numéro 6), en abstrayant les composants méthodologiques et en alignant leurs descriptions dans un métamodèle. Cependant, ces différents besoins ne sont pas mentionnés dans le spectre de Harmsen en tant que tels, ils sont potentiellement couverts par la construction situationnelle du modèle de processus. Cette échelle ne permet donc pas d'évaluer un modèle de processus au regard des attentes et des besoins des concepteurs et des développeurs et n'est donc pas utilisable pour évaluer les modèles de processus comme nous le désirons.

Résumé du chapitre 2

Dans ce chapitre, nous avons analysé les besoins de flexibilité des concepteurs et des développeurs lors de l'exécution des modèles de processus. Cette étude nous a permis d'identifier six besoins, qui sont :

(1) disposer de différents chemins ou différents enchaînements ;
(2) pouvoir modifier le modèle de processus pendant son exécution ;
(3) pouvoir réutiliser des solutions ou des embryons de solutions existantes et les capitaliser ;
(4) pouvoir accéder à différents niveaux d'information ;
(5) disposer de différents niveaux de complexité, sous la forme de différents langages ;
(6) disposer de plusieurs perspectives sur le modèle de processus.

Nous avons montré qu'il est nécessaire de définir une caractérisation graduée des modèles de processus pour évaluer leur correspondance avec ces besoins, en raison de la diversité des propositions existantes : les modèles de processus que nous avons analysés offrent des réponses plus ou moins complètes à un nombre variable de besoins. Une taxonomie, définissant des critères précis et une évaluation quantitative, permettrait de classer les modèles en fonction des réponses qu'ils apportent aux besoins des concepteurs et des développeurs.

C'est pourquoi nous avons exploré cinq taxonomies existantes, qui mesurent l'adéquation potentielle entre un modèle de processus et un projet ou servent de base pour des propositions de gestion de l'adaptation aux besoins locaux. Nous avons établi qu'aucune des ces propositions ne permet de mesurer la correspondance entre un modèle de processus et les besoins des concepteurs et des développeurs.

Ce constat nous a amené à définir notre propre taxonomie, que nous présentons dans le chapitre suivant, avant de l'appliquer à différentes approches.

3 PROMOTE : TAXONOMIE DES MODELES DE PROCESSUS

Nous avons vu dans le chapitre 2 qu'aucune des taxonomies existantes ne définit les critères nécessaires pour évaluer des modèles de processus au regard des besoins de flexibilité exprimés par les concepteurs et les développeurs. Ce manque nous amène à définir Promote (**Pro**cess **Mo**dels **T**axonomy for **E**nlightening choices), une taxonomie des modèles de processus, c'est-à-dire un outil conceptuel de catégorisation des modèles de processus en fonction de critères, chacun de ces critères étant associé à une graduation permettant d'exprimer à quel point le modèle répond au critère. Promote offre 36 critères, portant par exemple sur la structure du cycle, la maturité du modèle de processus, sa formalisation ou sa flexibilité.

Dans ce chapitre, la première section est consacrée à une présentation de trois modèles de processus qui nous permettront par la suite d'illustrer nos propos. Le chapitre se poursuit par un exposé des motivations qui nous ont amené à proposer une taxonomie des modèles de processus et de la démarche de recherche que nous avons mise en œuvre. La section suivante traite de la flexibilité telle qu'elle est définie dans Promote. Les autres axes de la taxonomie sont rapidement présentés dans la troisième section. Nous présentons ensuite une expérimentation que nous avons menée pour valider la taxonomie et l'outillage qui l'opérationnalise.

3.1 Quelques modèles de processus qui serviront d'illustration

Pour illustrer les concepts que nous allons présenter dans les sections suivantes, nous nous appuierons sur des exemples concrets. Nous prendrons ces exemples dans des modèles de processus existants, et nous avons choisi pour ce faire trois d'entre eux, le modèle du Waterfall (Royce 1990), Rapid Application Development (Martin 1991) et Extreme Programming (Beck 1999a). Le modèle du Waterfall nous intéresse particulièrement parce qu'il est considéré comme rigide par de nombreux auteurs (Glass 1969, Boehm 1979, Mills 1971, Parnas 1978, Prakash 1997). Rapid Application Development (Martin 1991), bien qu'étant totalement prédéfini, cherche à offrir une certaine flexibilité en proposant différents chemins et différents niveaux de détails. Il propose aussi différentes visions du modèle de processus: des descriptions générale, détaillée, et sous forme de cartes méthodologiques… Il nous permettra d'illustrer ces différents éléments. Enfin, Extreme Programming (Beck 1999a) a une réputation d'approche particulièrement flexible (Gonzalez-Perez & Henderson-Sellers 2008), ce qui nous permettra d'illustrer les écarts entre cette réputation et ce que nous considérons comme une flexibilité qui répond aux besoins des concepteurs et des développeurs.

3.1.1 Le modèle du Waterfall

Le modèle du Waterfall, ou modèle de la cascade, a été proposé par W. Royce (Royce 1990). Ce modèle comporte les étapes suivantes : (1) étude des besoins, (2) spécifications, (3) analyse, (4) conception, (5) codage, (6) tests et (7) déploiement (voir figure 3).

W. Royce notait que chaque étape était susceptible de remettre en cause l'étape précédente si les résultats n'étaient pas ceux attendus. C'est ce qu'il décrivait comme une itération entre deux phases successives. Il indiquait aussi (voir figure 3) que chaque étape pouvait impliquer une remise en cause d'une étape encore plus précoce que la seule étape précédente. En particulier, la phase de tests pouvait impacter la conception, l'analyse ou même l'étude des besoins. Il précisait cependant que, selon son expérience, si les étapes préliminaires avaient été bien conduites, les corrections induites par des tests insatisfaisants ne concernaient que des éléments mineurs de programmation et n'impactaient donc pas le reste du développement.

Royce recommandait aussi de diviser les projets « *développés pour la première fois* » en deux temps, avec un « *modèle pilote* » contenant les parties critiques en termes de conception ou de déploiement. En préconisant de travailler ces parties en deux temps (« *do it twice* »), W. Royce souhaitait confronter les points clés de la conception à des tests, afin de réduire l'impact de « *l'optimisme invariable et important* » de l'esprit humain quand il s'agit de concevoir des logiciels. Le modèle de processus comportait alors une première itération - une version miniature du processus global - sur un sous-ensemble du projet, qui était ensuite complétée et améliorée dans la seconde version.

Le spectre des projets concernés par le modèle du Waterfall est posé dès cette première publication : il s'agit de « grands projets », pour lesquels les besoins sont « stables ». La difficulté, telle qu'elle est présentée par W. Royce, se situe dans les phases amont du projet, lorsqu'il s'agit de définir comment traiter le problème. W. Royce mentionne ainsi des études complexes pour le calcul d'orbites, la détermination du comportement d'un engin spatial ou l'optimisation de calculs mathématiques. La notion de « grand projet » reste floue : W. Royce mentionne quelques chiffres sur la durée (12 mois, 30 mois,...), le budget (5 millions de dollars) et le volume des spécifications (1.500 pages), mais sans jamais indiquer que ces valeurs soient caractéristiques des projets qu'il propose de gérer.

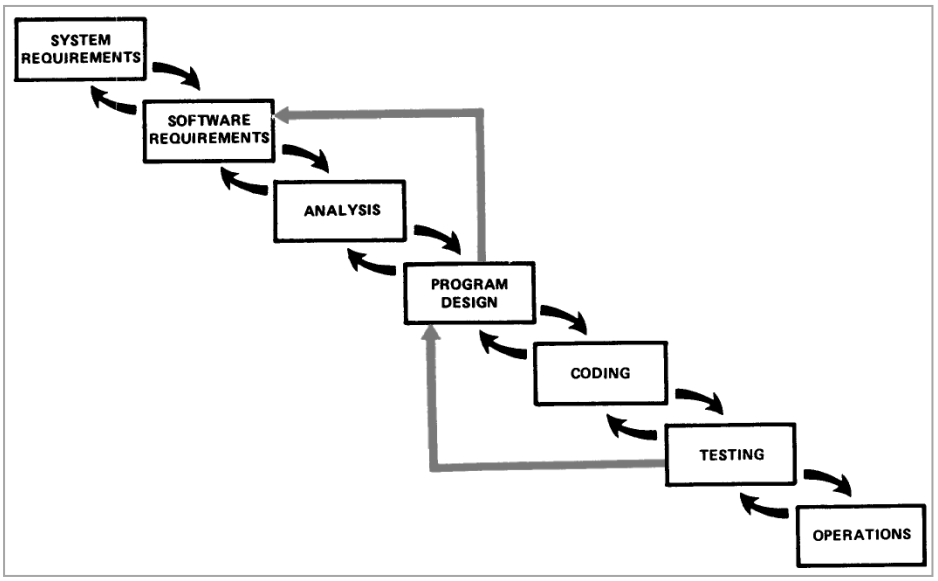

Figure 3- Le modèle du Waterfall, avec les retours en arrière potentiels

Le modèle du Waterfall sera largement utilisé dans les années 70 (Williams 1975). Par la suite, les auteurs chercheront à le compléter, en se focalisant notamment sur la fiabilité de l'étude des besoins (Jefferson 1979, Demuynck & Meyer 1979). Cette étude des besoins sera aussi interrogée sur sa complétude (Alford 1977), sa représentativité (Bell & Thayer 1976), sa désambiguïsation (Williams 1975) et sa compréhensibilité (Demuynck & Meyer 1979, Lekkos 1979). Malgré ces apports, ce modèle reste une référence en matière de rigidité : il est très régulièrement mentionné par les auteurs qui traitent de rigidité des modèles de processus (Williams 1975, Humphrey & Kellner 1989, Neighbors 1989, Booch 1993, Sommerville 1996, Helms 2001, Kroll & Kruchten 2003, Lindgaard et al. 2006, Guzelian 2007).

3.1.2 Rapid Application Development

Rapid Application Development (RAD) (Martin 1991) repose sur le prototypage comme élément de validation du système de cours de construction et propose un cycle basé sur des itération brèves (60 à 120 jours selon la complexité et la taille du projet). Cette approche est structurée selon quatre grandes phases :

1) Etude des besoins : les utilisateurs et l'équipe de développement analysent conjointement les besoins et les contraintes qui définissent le système à construire; cette phase s'arrête dès que les parties prenantes parviennent à un accord sur les principaux éléments de ce système ;
2) La conception centrée sur l'utilisateur : les concepteurs et les utilisateurs finaux détaillent les fonctions, leurs entrées et leurs sorties et les illustrent à l'aide de modèles et de prototypes ;
3) La construction : c'est la phase de programmation et de développement de l'application. Les utilisateurs sont présents et peuvent suggérer des améliorations ;
4) la coupe (« *Cutover* ») : cette phase comprend les tests et le déploiement de la version courante de l'application, ainsi que la formation des utilisateurs et la collecte de leurs retours.

RAD est décrit avec plusieurs niveaux de détails : le premier chapitre (Martin 1991) présente le modèle de processus de façon globale. Dans les chapitres qui suivent, les étapes sont décrites en langage naturel et sous la forme de cartes méthodologiques, proches d'algorithmes, qui les présentent avec plusieurs niveaux de détails et précisent les éléments à prendre en compte dans les différents choix. Enfin, les annexes offrent 250 pages d'explications et d'exemples sur certaines activités du modèle de processus, comme le calcul des points de fonction dans l'estimation des charges de travail ou la modélisation des données. J. Martin propose donc différents variants, différents niveaux de détails et de nombreux exemples.

3.1.3 Extreme Programming

Extreme Programming (XP) a été élaborée en 1999 (Beck 1999a). XP est une méthode agile, c'est-à-dire qu'elle appartient à une famille de méthodes qui adhèrent au Manifeste Agile (Fowler & Highsmith 2001) et qui, d'après (Anderson 2004), reposent sur deux idées

maîtresses : « *livrer de la valeur sous la forme de logiciel fonctionnel* » et « *accueillir le changement* ».

XP se focalise sur les besoins des utilisateurs et propose un cycle de développement basé sur des itérations brèves, qui peuvent être adaptées en fonction des changements qui surviennent dans les besoins des utilisateurs. XP promeut des valeurs de simplicité, de communication, de retours d'information, de respect et de courage. Son cycle est divisé en six phases : (1) la rédaction de scénarios utilisateurs, (2) le prototypage, qui sert de conception générale simplifiée, (3) la planification des versions, (4) le développement itératif des fonctions qui constituent une nouvelle version de l'application en cours de construction, (5) la soumission de cette version aux tests d'acceptation et (6) le déploiement de la version acceptée par les utilisateurs.

Après cette présentation de trois modèles de processus illustratifs, la section suivante présente notre taxonomie, Promote.

3.2 Présentation générale de Promote

3.2.1 Motivation et objectifs de Promote

Nous avons vu dans le chapitre précédent qu'il n'existe pas de taxonomie des modèles de processus prenant en compte les besoins des concepteurs et développeurs tels que nous les avons définis (voir section 2.4). Notre besoin initial était donc de proposer une taxonomie de la flexibilité des modèles de processus permettant d'évaluer l'adéquation entre un modèle de processus et les besoins des équipes de développement.

Cependant, une taxonomie ne portant que sur ces caractéristiques aurait eu une utilité limitée : elle n'aurait pas permis d'avoir une vision d'ensemble d'un modèle de processus, et, si une équipe de développement avait voulu l'utiliser pour choisir un modèle de processus, elle aurait été contrainte de se référer aussi à d'autres travaux, et surtout de faire un travail important d'alignement des concepts entre les différents travaux. Nous avons vu par exemple que les critères définis dans (Alexander & Davis 1991) sont au nombre de 20, deux d'entre eux seulement traitant de l'équipe de développement, sans prendre en compte les besoins de flexibilité de cette équipe. Les autres critères portent sur les utilisateurs finaux, le système à construire, la disponibilité des ressources et les besoins de l'organisation. Il semblait donc important, pour permettre une utilisation réelle de la taxonomie, d'inclure aussi d'autres critères que ceux correspondant aux besoins de flexibilité des concepteurs et développeurs.

Sans prétendre à l'exhaustivité de tous les critères imaginables répondant à tous les besoins possibles, Promote se donne comme objectif de fournir une vision globale aux spécialistes du développement de logiciels et de les aider à identifier les approches qui correspondent à leurs besoins.

3.2.2 Méthode de recherche

Comme nous l'avons indiqué dans l'introduction, nous avons procédé par cycles d'induction et déduction, en basant notre travail sur un corpus de 574 publications issues des communautés de l'Interaction Homme-Machine, pour les travaux sur le développement des IHM, et des Systèmes d'Information et du Génie Logiciel, pour les travaux sur les modèles de processus. Parmi ces publications, 263 d'entre elles présentent 65 modèles et métamodèles de processus ; 9 classifient les modèles de processus ; 185 traitent directement d'un ou de plusieurs aspects des modèles de processus, comme les itérations ou les formes de collaboration ; 41 traitent de l'ingénierie des méthodes ; 56 traitent de l'ingénierie de l'interaction homme-machine ; 23 portent sur l'ingénierie des besoins ; 49 portent sur l'ingénierie dirigée par les modèle ; 15 traitent de l'évaluation des méthodes ; 8 étudient l'adoption des méthodes ; 36 étudient différentes facettes de la flexibilité, au sens général du terme.

Dans un premier temps, nous avons analysé les différentes publications mentionnées ci-dessus, afin d'identifier les différentes caractéristiques que les auteurs intègrent dans leurs propositions ou leurs analyses (modèle de processus séquentiel ou itératif, incrémental ou non, etc.). Au cours de cette étude, nous avons noté les différentes définitions de ces caractéristiques et les différentes formes possibles de ces caractéristiques dans les (méta)modèles de processus. Nous avons synthétisé les différentes définitions identifiées dans ces publications afin d'élaborer une proposition de définition pour chacun des critères.

Nous avons ensuite défini une graduation pour chacun de ces critères. Dans quelques cas, la littérature comporte des études qui peuvent servir de base à la définition des graduations. C'est par exemple le cas pour le critère portant sur les rôles définis par le modèle de processus : deux études complémentaires, (Harrison & Coplien 1996) et (Phalp & Shepperd 2000), montrent d'une part qu'un « trop grand » nombre de rôles (au delà de 21) induit une baisse du « *ratio de communication* » (chaque rôle doit communiquer avec un grand nombre d'autres rôles), ce qui dégrade la performance de l'équipe (Harrison & Coplien 1996). A l'inverse, un « trop petit » nombre de rôles (de l'ordre de 3) conduit à la définition de rôles très génériques, comportant des activités sans lien les unes avec les autres, ce qui réduit la cohérence interne de chaque rôle (Phalp & Shepperd 2000). La combinaison de ces deux études nous a permis de créer une graduation basée sur le nombre de rôles.

Mais dans la plupart des cas, de telles études n'existent pas. Nous avons alors basé les échelons des graduations sur les différentes possibilités offertes par les modèles et métamodèles de processus.

Ainsi, pour les itérations, l'analyse des modèles existants montre qu'il existe :

- des modèles de processus non itératifs, comme le modèle du Waterfall (Royce 1970) ;

- des itérations très localisées, que nous nommons locales, comme le *Daily Meeting* de Scrum (Schwaber 1995), qui survient chaque jour mais ne comporte qu'une activité ;
- des itérations très larges, que nous appelons globales, comme le cycle du modèle en Spirale (Boehm 1986), qui répète l'ensemble des activités du cycle ;
- des itérations dont le nombre d'activités se situe entre ces deux extrêmes, et que nous appelons régionales, comme dans Feature Driven Development (Coad et al. 1999), où la phase de réalisation est faite fonctionnalité par fonctionnalité ;
- des modèles de processus comportant plusieurs types d'itérations, comme Rapid Application Development (Martin 1991), qui propose de construire le système version après version, ce qui est une itération globale, tout en répétant des séquences régionales d'activités. La phase de conception, par exemple, comporte une session de conception, un prototypage et une évaluation par les utilisateurs finaux, dont les retours servent de point de départ d'une nouvelle session de conception.

La graduation issue de cette analyse reprend alors ces différentes possibilités, ainsi que leurs différentes combinaisons possibles. Ces graduations sont donc destinées à évoluer avec les modèles de processus, pour permettre de représenter d'éventuels nouveaux modèles.

Une fois obtenus des critères et des graduations, c'est-à-dire une fois le travail d'induction de notre méthode de recherche terminé, nous avons confronté la version courante de la taxonomie à 65 modèles de processus. Notre objectif dans cette phase était de vérifier que nous pouvions associer un et un seul échelon d'une graduation à un modèle de processus et que deux modèles de processus que nous considérions comme différents étaient associés à deux échelons différents de la graduation. Les résultats de cette évaluation servaient de point de départ à un nouveau travail d'induction.

Au fil des cycles d'induction et de déduction, nous avons élaboré une première version à priori satisfaisante de Promote. Nous l'avons présentée lors de la conférence *INformatique des ORganisations et Systèmes d'Information de Décision* (Inforsid 2011), et dans le *Doctoral Consortium* de *Engineering Interactive Computing Systems* (EICS 2011). Les principaux retours ont porté sur certaines définitions trop floues. Ces remarques nous ont permis de préciser la taxonomie.

Une fois parvenus à une deuxième version, nous l'avons expérimentée auprès de concepteurs et de développeurs novices (voir en section 3.5), ce qui a mis en évidence des besoins que nous n'avions pas identifiés. Un nouveau travail d'induction nous a permis de compléter l'étude des besoins et la taxonomie, que nous avons ensuite publiée dans *Information and Software Technologies* (Céret, Dupuy-Chessa, et al. 2013). Par la suite, Promote a été utilisée en enseignement (Master 2 MOSIG de l'IM2AG[5]), présentée à des chercheurs (équipe Sigma du Laboratoire d'Informatique de Grenoble) et à des industriels

[5] Unité de Formation et de Recherche en Informatique, Mathématiques et Mathématiques Appliquées de Grenoble

(journées Neptune et Club Automation en 2013). C'est la version amendée suite aux retours collectés au cours de ces rencontres que nous présentons ci-dessous. Nous commencerons par présenter les critères de Promote qui traitent de la flexibilité des modèles de processus.

3.3 La flexibilité dans Promote

La flexibilité est définie dans Promote selon six dimensions indépendantes. Pour chacune d'entre elles, une graduation est proposée, afin de quantifier à quel point un modèle de processus offre chaque forme de flexibilité.

Pour définir chaque axe, nous nous sommes basés sur deux éléments : l'analyse des besoins des concepteurs et développeurs que nous avons présentée dans le chapitre 2, et l'étude d'un corpus de publications portant sur la flexibilité des modèles de processus. Comme nous l'avons présenté dans l'introduction de cette thèse, les travaux qui traitent de la flexibilité des modèles de processus ne se réfèrent pas tous aux besoins des concepteurs et des développeurs. Il était donc nécessaire de confronter ces travaux aux besoins de flexibilité correspondant à ceux exprimés par les concepteurs et développeurs tels que nous les avons déterminés dans le chapitre 2, à savoir :

(1) disposer de différents chemins ou différents enchaînements ;
(2) pouvoir modifier le modèle de processus pendant son exécution ;
(3) pouvoir réutiliser des solutions ou des embryons de solutions existantes et les capitaliser ;
(4) pouvoir accéder à différents niveaux d'information ;
(5) disposer de différents niveaux de complexité, sous la forme de différents langages ;
(6) disposer de plusieurs perspectives sur le modèle de processus.

La démarche que nous avons mise en œuvre dans la définition de l'axe de la flexibilité est la même que celle que nous avons utilisée des autres axes de Promote, et que nous avons présentée en section 3.2.2 : des cycles d'induction, comportant l'analyse des publications au regard des besoins des concepteurs et des développeurs et amenant à la proposition de définitions de critères et de graduations, et de déduction, comprenant une confrontation entre nos propositions et les modèles de processus, ainsi qu'une expérimentation auprès de concepteurs novices et des présentations académiques.

Les sections suivantes présentent les six critères résultant de ce travail et définissant la flexibilité au regard des besoins des concepteurs et des développeurs.

3.3.1 Variabilité

Les concepteurs et développeurs expriment le besoin que les modèles de processus offrent différents chemins ou différents enchaînements (besoin numéro 1), pour éviter une trop grande linéarité, s'adapter aux caractéristiques des projets et favoriser l'exploration de solutions variées. Le but n'est pas ici de s'assurer qu'il existera un chemin

adapté aux besoins de l'équipe de développement, mais de mesurer si le modèle de processus propose différentes possibilités. La définition que nous proposons est donc :

> La variabilité est la capacité d'un modèle de processus à proposer différents chemins parmi les éléments qui le composent.

La formulation en « éléments qui le composent », et non en « activités qui le composent » est volontaire. En effet, si les modèles de processus les plus anciens sont structurés autour d'activités, d'autres propositions ont été faites pour les structurer autour d'artefacts, comme dans le Work Product Pool (Gonzalez-Perez & Henderson-Sellers 2008), autour de buts et de stratégies, comme dans la MAP (Rolland et al. 1999), et plus rarement autour des décisions faites par l'équipe (Dowson 1987) ou autour du contexte ayant conduit à prendre ces décisions (Pohl et al. 1994).

Les modèles de processus existants offrent différents niveaux de variabilité. Ainsi, Rapid Application Development (Martin 1991) propose des choix possibles dans plusieurs étapes, comme la libre sélection des outils, la possibilité d'utiliser une « *timebox* » (une structuration temporelle imposant des livraisons à des dates fixées), la possibilité d'associer ou de dissocier des étapes dans la planification et la conception centrée sur l'utilisateur. Ces variants sont présentés explicitement, dans des cartes méthodologiques (sous une forme proche d'algorithmes), avec des critères bien définis. D'autres modèles offrent moins de choix ou avec des critères moins clairement établis, comme le modèle du Waterfall (Royce 1970), qui propose un seul variant : faire un prototype des fonctions critiques lorsque le logiciel est développé pour la première fois, ce que Royce nomme le « *do-it-twice* ».

La graduation associée à cette dimension, et destinée à prendre en compte cette diversité, est :

- Le modèle de processus ne propose **aucun variant**
- Il propose de **rares variants informels**
- Il propose de **rares variants bien définis**
- Il propose **quelques variants informels** dans certaines parties
- Il propose **quelques variants bien définis** dans certaines parties
- Il propose **de nombreux variants informels** dans la plupart des parties
- Il propose de **nombreux variants bien définis** dans la plupart des parties

La différence entre un variant informel et un variant bien défini réside dans la précision des conditions qui permettent de faire un choix. Ainsi, proposer une option « si le projet est novateur » relève d'une condition subjective et donc d'un variant informel, tandis que « si l'équipe comporte moins de 7 personnes » relève d'une condition bien définie.

3.3.2 Granularabilité

Nous avons vu dans la section 2.4.3 que les concepteurs et développeurs attendent qu'un modèle de processus contienne différents niveaux d'information et de complexité, besoin que nous avons identifié sous le numéro 4. La plupart des méthodes proposent plus d'un niveau de détail : on trouve souvent une présentation graphique du cycle, un descriptif général et un descriptif détaillé.

Nous avons identifié dans la littérature deux façons de fournir ces différents détails : d'une part, en multipliant les supports dédiés à l'une ou l'autre facette du modèle de processus et, d'autre part, en incluant différents niveaux détails dans un même support.

Un exemple de la multiplication des supports est Extreme Programming : cette approche est présentée par l'auteur initial dans des livres (Beck 1999b, Beck & Fowler 2000) et des publications scientifiques (Beck 1999a). Une grande profusion de documents complémentaires s'ajoutent à ces documents : des milliers de sites Internet et de livres[6], ainsi que des guides de poche comme (O'Reilly 2003). Certains de ces documents sont dédiés à un élément de la méthode, comme la gestion de projet (Bénard et al. 2002) ou des exemples d'applications (Newkirk & Martin 2001), complétant ainsi la documentation initiale. On peut penser que cette profusion d'études, et de recommandations permet de répondre au besoin de différents niveaux de détails : le concepteur ou le développeur qui cherche une information n'aura aucun mal à trouver un site Web ou un document qui lui apporte des éléments de réponse. Pourtant, cette profusion n'est pas satisfaisante. En effet, elle pose deux problèmes : la sélection de l'information et la vérification de sa validité. On trouve sur Internet des allégations totalement contradictoires, comme « *[Extreme Programming] est une méthodologie* ***complète et suffisante*** *qui permet de transformer une vague idée en produit utile et de l'accompagner jusqu'à sa fin de vie*[7] » tandis que Sharp estime que la même méthode donne **trop peu d'indications** sur les rôles non-techniques, ce qui imposera de trouver du guidage ailleurs (Sharp et al. 2006). Le modèle de processus est-il alors complet ou non ? On trouve dans les forums des débats sans fin, avec des arguments comme « *Scrum peut-être vu comme un framework méthodologique [...] tandis que XP concerne les activités de développement* » et quelques lignes plus loin « *Extreme Programming et Scrum [...] sont deux méthodologies de gestion de projet*[8] ». Le concepteur ou le développeur qui cherche à se faire une opinion sur l'adéquation d'une approche à son besoin trouvera tout et son contraire.

D'autres méthodes ont été présentées sous des formes qui semblent plus adaptées. Par exemple, le livre qui présente Rapid Application Development (Martin 1991) est structuré autour de quatre niveaux de détails : le premier chapitre présente une vision globale de la méthode en quelques pages, que le quatrième chapitre détaille en une vingtaine de pages. Les chapitres 7 à 24 reprennent tous les aspects de l'approche en détail et avec des

[6] Le 25/03/2014, Google trouve 34 millions de pages et 76.800 livres à propos de Extreme Programming.
[7] http://www.dominicwilliams.net/fr/just_xp.html, consulté le 31/03/2014
[8] https://www.linkedin.com/groups/Je-veux-savoir-quand-utiliser-4421531.S.119635255, 31/03/2014

explications précises. Enfin, 23 « *cartes méthodologiques* » présentent des algorithmes détaillés avec tous les critères de choix permettant de prendre des décisions éclairées.

Pour catégoriser les niveaux de détails offerts par un modèle de processus, nous avons donc choisi de restreindre les documents pris en compte à ceux publiés par le même auteur, ou la même institution, dans les cas où une méthode est proposée par un organisme, comme SPEM (Object Management Group 2008), proposée par l'Object Management Group.

A l'issue de cette étude, nous avons proposé la définition suivante :

> La granularabilité est la capacité d'un modèle de processus à offrir différentes granularités, autrement dit différents niveaux de détails dans les éléments qui le constituent.

L'analyse des modèles de processus permet de définir la graduation pour la granularabilité comme :

- Le modèle de processus comporte **une seule granularité**
- Le modèle de processus comporte **deux granularités**
- Le modèle de processus comporte **plus de 2 granularités**

3.3.3 Stratégies d'apprentissage

Les concepteurs et développeurs souhaitent disposer de différents langages dans la description du modèle de processus (besoin numéro 5). Cependant, il est extrêmement rare qu'un auteur fournisse un exemple simple pour aider à la compréhension des concepts qu'il manipule. Ainsi, W. Royce explique que pour un projet avec un budget de 5 millions de dollars, il s'attend à trouver un dossier de spécifications de 30 pages (Royce 1970), sans expliquer ce qu'il entend par « dossier de spécification ». La méthode UsiXML (UsiXML Consortium 2011) détaille des phases de construction de modèles, mais omet de définir ce qu'est un « *modèle de tâches* » ou un « *modèle de domaine* » : il faut se référer à un document spécifique pour comprendre le sens de ces termes (UsiXML Consortium 2012). Ces omissions sont légitimes, dans la mesure où il serait complexe de redéfinir chaque terme dans chaque document. Elles peuvent cependant mettre en difficulté un utilisateur novice, qui ne maîtriserait pas ce vocabulaire. C'est pourquoi nous questionnons les moyens qu'offre un modèle de processus pour son apprentissage :

> Les stratégies d'apprentissage sont les différentes modalités d'apprentissage permises par un modèle de processus.

La pédagogie différenciée (Przesmycki 2008) constate en effet que les capacités d'apprentissage diffèrent d'un apprenant à l'autre en fonction de ses caractéristiques

cognitives, socioculturelles et psychologiques et de son type d'apprentissage dominant. Elle préconise de mettre en œuvre plusieurs stratégies d'enseignement pour que chaque apprenant puisse travailler selon son propre itinéraire d'appropriation. D'après le modèle de Kolb en théorie de l'apprentissage, il y a quatre types d'apprenants (Kolb 1976), qui mobilisent plus ou moins des capacités d'observation, d'expérimentation, de conceptualisation ou de ressenti. Ces styles d'apprentissages sont représentés sur la figure 4, et définis comme[9] :

- le **divergent**, qui s'intéresse aux gens et aux émotions et qui favorise la réflexion basée sur une expérience ; il est habile à percevoir un objet ou un problème sous différents angles. Il apprécie les activités novatrices, il a une imagination fertile et des intérêts variés ;
- l'**accommodateur**, qui aborde les problèmes par la mise en application de l'idée/action fondée sur l'expérience ; il apprend par manipulation, exécutant les tâches. Il aime être impliqué dans la planification et la réalisation d'activités, il fonctionne par essais / erreurs plutôt que par la logique ;
- le **convergent**, qui préfère les phases de conceptualisation abstraite et théorique de l'expérience accompagnées de la mise en application de l'idée/action. Les convergents aiment être pratiques et ont tendance à être peu émotifs; ils préfèrent composer avec des choses plutôt qu'avec des gens ;

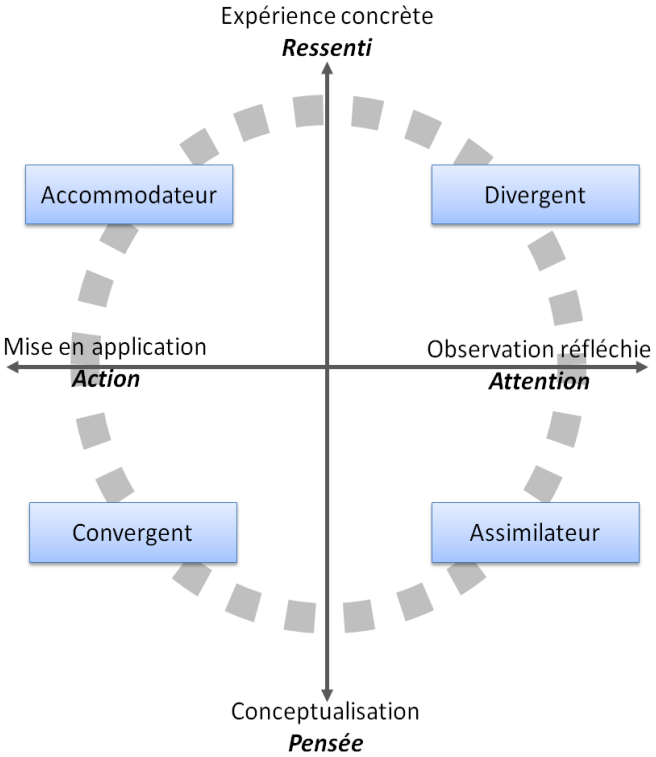

Figure 4 - Les quatre styles d'apprentissage de Kolb

[9] Les définitions présentées ici sont issues de la présentation des approches pédagogiques de l'INSA : http://enseignants.insa-toulouse.fr

- l'**assimilateur**, qui s'intéresse plus à la conceptualisation abstraite et théorique et moins aux applications pratiques, aime créer des modèles théoriques et jongle avec les idées, s'intéressant s'intéressent moins que les autres aux gens et aux applications pratiques des connaissances.

On peut donc imaginer qu'un modèle de processus soit présenté de plusieurs façons, chacune convenant mieux à l'un ou l'autre style d'apprentissage. Par exemple, à côté de la classique présentation des activités à mener, les auteurs pourraient proposer un cas d'étude avec le détail de la démarche menée, des témoignages de personnes ayant mis en œuvre le modèle de processus ou un « jeu sérieux » permettant de s'approprier la démarche.

Plutôt que de présenter quatre documents différents, on peut aussi concevoir un modèle de processus qui intègre plusieurs styles d'apprentissage dans sa structure, par exemple en proposant plusieurs formulations et documentations pour une même activité. Ainsi, une activité indiquant « créer le modèle de tâches », qui serait suffisante pour un expert du domaine, pourrait être accompagnée de tutoriels, d'exemples concrets et de reformulations avec un vocabulaire simple pour les moins experts.

La graduation utilisée pour les stratégies d'apprentissage correspond au nombre de styles d'apprentissages inclus dans le descriptif :

- **Une** stratégie d'apprentissage
- **Deux** stratégies d'apprentissage
- **Trois** stratégies d'apprentissage
- **Quatre** stratégies d'apprentissage

Comme pour la granularabilité, les différentes stratégies d'apprentissage prises en compte sont celles proposées les auteurs du modèle de processus. Nous considérons par ailleurs que le modèle de processus doit être « autoportant », c'est-à-dire que les différentes stratégies d'apprentissage doivent être incluses dans le modèle de processus (liens hypertexte vers des sites Internet, documentation directement associée aux activités,...). Ainsi, une formation payante proposée par une entreprise sans lien avec les auteurs de la méthode n'est pas considérée ici comme une partie de la méthode. Un exemple d'éléments que nous pouvons prendre en compte sont les différents documents destinés à l'apprentissage de la méthode Domain-Driven Design (Evans 2004), qui sont proposés par le site Internet dddcommunity.org, dans lequel E. Evans, l'auteur de la méthode, est impliqué.

3.3.4 Distensibilité[10]

Les concepteurs et développeurs attendent que le modèle de processus soit modifiable durant son exécution (besoin numéro 2). Le but est maintenant d'évaluer si le modèle de processus propose des moyens d'ajouter, modifier ou supprimer des éléments qu'il

[10] Ce terme provient du monde médical, où il désigne la capacité d'un organe à se contracter et à se détendre, comme le cœur ou les poumons.

comprend initialement. Par exemple, les équipes aimeraient pouvoir ajouter des activités relatives à la gestion du projet, comme la planification des activités (Garzotto & Perrone 2007) ou supprimer des niveaux de détails inutiles (Barry & Lang 2001). Ces besoins nous conduisent à la définition suivante :

> La distensibilité est la capacité d'un modèle de processus à être étendu ou réduit pendant son exécution.

La plupart des modèles de processus que nous avons étudiés ne proposent pas cette possibilité. Par exemple, le modèle en Spirale (Boehm 1986) et Rapid Application Development (Martin 1991) ne mentionnent aucune capacité d'extension. Extreme Programming (Beck 1999a) suggère d'ajouter des activités ou des outils lorsqu'il en est besoin, mais ne définit pas comment procéder.

Pourtant, de nombreux modèles de processus ont été étendus. Par exemple, Kolodner, lorsqu'il propose le *Case-Based Reasoning Process* (Kolodner et al. 1985), explique qu' « *il s'agit d'une extension de ce que Shank (1982) appelle la mémoire intentionnelle, et est aussi un composant du Projecteur de Wilensky (1983)* ». Dans le même ordre d'idées, des travaux comme (Laanti 2008) ont proposé d'adapter Scrum (Schwaber 1995) de façon à gérer plusieurs équipes collaborant sur un même projet, ce qui constitue une extension du modèle de processus de Scrum. Mais au final, ces extensions ne relèvent pas d'une caractéristique particulière du modèle de processus : celui-ci ne comportait rien de particulier qui ait encadré ou facilité ce travail. Nous ne parlerons de distensibilité que si le modèle de processus comporte de tels éléments. Il peut s'agir d'activités spécifiques (que l'on pourrait appeler des méta-activités), de points de contrôles à vérifier avant de faire la modification, ou encore d'outillage du modèle de processus prenant en charge ces changements et vérifiant la cohérence résultante.

A l'inverse de (Mulyar et al. 2007), la définition que nous proposons ne fait pas de différence entre la modification temporaire (le modèle de processus est amendé pour permettre la modification du processus, mais ces modifications ne sont pas capitalisées) et la modification permanente (la modification persiste dans le modèle de processus au-delà du processus). L'objectif premier est en effet que l'utilisateur puisse ajouter ou retrancher dynamiquement ce qui lui convient, conformément au besoin numéro 2.

La graduation associée à la distensibilité prend en compte les différentes possibilités proposées par les approches que nous avons étudiées. Nous l'avons dit, la plupart des modèles de processus ne parlent pas de leur propre extension, et quand ils le font, c'est d'une façon informelle et peu guidante, comme nous l'avons vu pour Extreme Programming (Beck 1999a) en section 3.1.3. Nous avons identifié deux paliers supplémentaires : la procédure permettant la distensibilité peut être plus ou moins bien définie. Nous avons vu dans la section 2.5.4 que Mulyar propose des patrons pour flexibiliser différentes phases du modèle de processus. Ces patrons peuvent être

considérés comme des procédures bien définies. Nous traduisons ces différentes possibilités dans la graduation :

- Le modèle de processus ne propose **pas de procédure** d'extension ou de réduction
- Il propose **des moyens informels** d'extension et de réduction sans procédure explicite décrivant comment faire
- Il propose des **procédures** d'extension et de réduction **bien définies** mais sans expliciter les limites ni les modes de validation
- Il propose des procédures d'extension et de réduction **bien définies détaillant les limites et/ou les modes de validation**

3.3.5 Réutilisation

Parmi les besoins des concepteurs et développeurs, nous avons vu le besoin de capitalisation-réutilisation (besoin numéro 3), dont l'objectif est d'augmenter les possibilités d'exploration par la possible réutilisation de solutions ou d'embryons de solutions existantes.

La littérature montre que plusieurs voies s'offrent pour répondre à ce besoin. Par exemple, les patrons de conception permettent de réutiliser des solutions testées et prouvées afin de résoudre des problèmes récurrents (Gamma et al. 1995). Une autre façon de répondre à ce besoin est d'offrir des composants prédéfinis, que Mulyar nomme des « *valeurs par défaut* » en entrée d'une activité (Mulyar et al. 2007). Ainsi, l'atelier de conception d'applications Web Django (Django Software Foundation 2005) propose un modèle d'utilisateur par défaut ainsi que le système d'identification correspondant. Les équipes de réalisation peuvent alors utiliser directement cette proposition, ou s'en servir comme point de départ pour définir une solution mieux adaptée à leurs besoins, et peuvent s'inspirer des procédures offertes par l'atelier pour mener à bien cette adaptation. Django n'est certes pas un modèle de processus, mais un atelier de développement. Cependant, les modèles de processus pourraient tout à fait offrir des approches similaires. Au niveau procédural, la flexibilité par changement permanent (Mulyar et al. 2007) permet de réutiliser des modifications du modèle de processus.

A partir de ces exemples, nous définissons :

> La réutilisation mesure la proposition d'éléments de solutions dans un modèle de processus.

La graduation de cette forme de flexibilité doit évaluer à quel point un modèle de processus supporte la réutilisation. Cependant, l'inclusion d'éléments réutilisables est encore relativement limitée dans les modèles de processus existants. De plus, les activités et les produits pour lesquels il est possible de fournir un élément réutilisable sont en nombre restreint. Il est par exemple difficilement imaginable de prédéfinir un modèle de

tâches, même sous une forme embryonnaire, car il représente un problème spécifique pour un groupe d'utilisateur donné. Cette rareté explique que la graduation ci-dessous considère comme flexibles des modèles de processus qui ne comportent que peu d'éléments de solution. Les valeurs limites indiquées en exemple sont empiriques et proviennent de notre analyse d'une soixantaine de modèles de processus.

La graduation proposée est :

- Le modèle de processus ne propose **aucun élément de solution**
- Le modèle de processus propose **quelques rares éléments de solution** (par exemple, il propose une solution prédéfinie, plus ou moins complète, pour 5% des artefacts à produire)
- Le modèle de processus **propose régulièrement des éléments de solution** (par exemple, il suggère une solution pour 5 à 20% des artefacts à produire)
- Le modèle de processus propose **de nombreux éléments de solution** (par exemple, il suggère une solution pour plus de 20% des artefacts à produire)

3.3.6 Polymorphisme

Le polymorphisme évalue comment le modèle de processus offre une ou plusieurs vues sur le processus, conformément au besoin numéro 6. Nous le définissons comme :

> Un modèle de processus est dit polymorphique s'il permet à ses utilisateurs de l'aborder selon ses différents constituants, grâce à des présentations multiples.

Il ne s'agit pas ici de dire qu'un modèle de processus est polymorphique à partir du moment où il contient par exemple une représentation graphique de son cycle de vie et une description de ce dernier en langage naturel : ceci relève plutôt de deux niveaux de détails (une vue d'ensemble et une vue détaillée) ou de différents langages (une syntaxe graphique et une syntaxe textuelle). Il s'agit ici de mesurer comment le modèle de processus permet que l'utilisateur puisse consulter le travail qu'il a à mener sous plusieurs formes : une liste d'activités à réaliser ou une liste de produits à élaborer, par exemple. Ainsi, le *Work Product Pool*, proposé par Gonzalez-Perez et Henderson-Sellers (Gonzalez-Perez & Henderson-Sellers 2008) est centré sur les produits[11]. Il n'est pas prévu pour permettre simplement à l'utilisateur de voir la liste des activités qu'il devra réaliser dans le futur : il faut consulter l'un des artefacts à réaliser pour ensuite, et ensuite seulement, avoir accès aux activités qui permettent de construire cet artefact. A l'inverse, le modèle du Waterfall (Royce 1970) est centré sur les activités, il faut consulter chaque activité pour voir les produits qu'elle propose d'élaborer.

Les modèles de processus peuvent être centrés sur les activités, les produits, les décisions, les buts ou le contexte (Rolland 2005). Classiquement, au-delà de l'élément

[11] Pour plus de détails sur le *Work Product Pool*, voir la section 4.4.2.

central (les buts, par exemple), un modèle de processus propose des activités et des produits, c'est le sens même de la définition de méthode telle que nous l'avons présentée en introduction : une méthode comporte un modèle de processus (des activités) et un modèle de produits (des artefacts) (Booch 1993) (Harmsen 1997). Il est donc classique d'accéder par exemple aux artefacts en navigant à travers les autres éléments.

L'objectif du polymorphisme est d'éviter cette navigation pour l'utilisateur final : est-il possible, par exemple, de voir les artefacts sans naviguer par les activités ? Si le modèle de processus propose différentes présentations, centrées sur différents éléments, alors nous le considérons comme polymorphique. Ces présentations distinctes peuvent être imaginées sous plusieurs formes, comme différentes sections d'un livre ou différents modes d'accès offerts par l'outillage du modèle de processus.

Dans la littérature, la plupart des modèles de processus ne comportent qu'une présentation. Dans l'état actuel, il n'apparaît donc pas nécessaire de proposer une graduation avec de multiples échelons. C'est pourquoi la graduation du polymorphisme est la suivante :

- Le modèle de processus (ou l'outillage afférent) ne permet qu'**une présentation du processus**
- Le modèle de processus (ou l'outillage afférent) offre **deux présentations du processus**
- Le modèle de processus (ou l'outillage afférent) offre **plus de deux présentations du processus**.

Le polymorphisme est la sixième et dernière forme de flexibilité définie dans Promote. Il conclut l'axe de la flexibilité, qui apporte une réponse à notre seconde question de recherche, l'évaluation de la réponse apportée par un modèle de processus aux besoins des concepteurs et des développeurs. Cette définition constitue aussi notre première contribution. Cependant, nous l'avons expliqué en introduction de ce chapitre, une taxonomie qui n'aurait comporté que la dimension de la flexibilité n'aurait pas été utilisable pour identifier les caractéristiques d'un modèle de processus. Nous avons donc défini d'autres critères, que nous présentons rapidement dans la section suivante.

3.4 Les cinq autres dimensions de Promote

Promote comporte cinq autres axes principaux (Céret, Dupuy-Chessa, et al. 2013), que l'on peut voir sur la figure 5 :

- **Le cycle de vie**, qui décrit l'organisation interne du cycle proposé par le modèle de processus, et comporte sept critères. le cycle peut être plus ou moins ***incrémental***, par exemple XP (Beck 1999a), qui propose de livrer une version du produit toutes les une à quatre semaines, est considéré comme très incrémental. Le cycle peut aussi être plus ou moins ***itératif***, ce qui est mesuré par la portée des itérations. Ainsi, RAD (Martin 1991), qui propose une itération sur l'ensemble des étapes de développement pour la création de différentes versions et une itération au sein de ces étapes, est considéré comme offrant des

itérations globales et régionales. La ***durée*** du cycle peut être plus ou moins longue. Le modèle du Waterfall (Royce 1970) ne donne aucune indication de durée, tandis que RAD (Martin 1991) propose un cycle de 60 jours. Le cycle peut comporter des étapes à réaliser en ***parallèle***. Ce n'est le cas ni du modèle du Waterfall (Royce 1970), ni de XP (Beck 1999a) ni de RAD (Martin 1991). Cependant, des approches comme Symphony (Hassine et al. 2002) ou Two Track Unified Process (Roques & Vallée 2002) recommandent un cycle en Y, afin de réaliser en parallèle l'étude des besoins et l'étude des aspects techniques ; Le cycle peut permettre ou non des ***retours en arrière***, comme XP (Beck 1999a), qui définit précisément comment les demandes de modification des utilisateurs finaux doivent être intégrés dans les tâches à réaliser dans l'itération suivante. Le cycle peut être basé sur différentes ***approches***. Ainsi, RAD (Martin 1991) propose de décomposer un problème complexe en problèmes plus simples, ce qui correspond à une approche descendante. Le cycle peut être centré sur différents éléments, comme les activités, les produits, les buts, les décisions, que nous nommons des ***focales*** du modèle de processus. Le modèle du Waterfall (Royce 1970), RAD (Martin 1991) et XP (Beck 1999a) proposent un processus présenté à travers ses activités : ils sont tous les trois centrés sur les activités.

- **La collaboration**, qui analyse comment le modèle de processus propose aux différentes parties prenantes de travailler ensemble. Au niveau interne, c'est-à-dire entre les membres de l'équipe de développement (les « informaticiens »), Promote propose de mesurer le ***nombre de rôles*** (l'idéal défini dans la littérature étant moins de 16 rôles clairement définis), ainsi que la ou les ***formes de collaboration*** recommandées (qui peuvent être de l'échange, de la

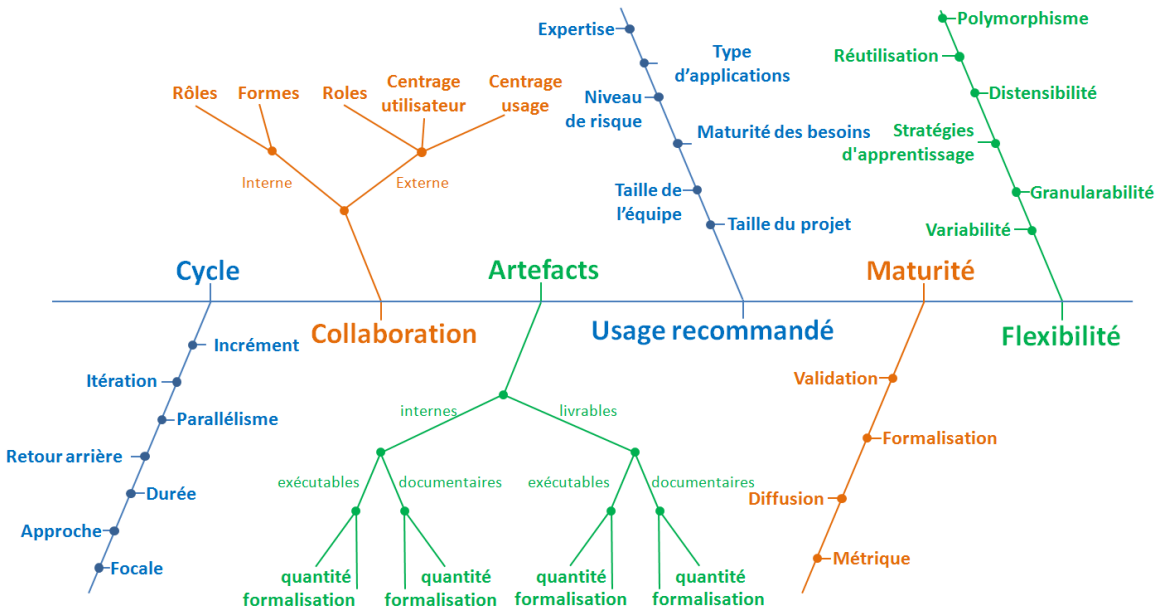

Figure 5 - Les axes de classification des modèles de processus de Promote

coordination ou de la coopération). Pour les autres parties prenantes (niveau « externe »), Promote propose aussi de quantifier les rôles, et de mesurer la quantité d'***activités centrées sur l'utilisateur*.** Ainsi, RAD (Martin 1991), qui spécifie de nombreuses activités réalisées conjointement par l'équipe de développement et les utilisateurs finaux, est fortement centrée sur l'utilisateur. Promote propose aussi d'évaluer la quantité d'***activités centrées sur l'usage***, c'est-à-dire focalisées sur la modélisation de la représentation visuelle et de l'interaction (Constantine et al. 2003). RAD (Martin 1991) propose l'utilisation de nombreux modèles, mais seuls quelques-uns concernent l'interaction, comme le diagramme d'actions. RAD est considéré comme légèrement centré sur l'usage.

- **Les artefacts** (ou produits) sont catégorisés comme ***internes*** (non destinés à être livrés au client) ou, à l'inverse, ***externes***, et subdivisés en produits ***documentaires*** ou ***exécutables***. Pour chacune des quatre catégories ainsi définies, Promote propose de mesurer le niveau de formalisation et la quantité préconisés par le modèle de processus. Ainsi, XP (Beck 1999a) définit peu d'artefacts et ne donne aucune indication sur leur formalisation.
- **L'usage recommandé** identifie les contextes pour lesquels le modèle de processus indique être adapté et repose sur des critères comme la ***taille du projet*** (et les indications pour évaluer cette taille), la ***taille de l'équipe***, la ***maturité des besoins*** (les besoins peuvent ou non changer pendant le processus), le ***niveau de risque*** traité, le ***type d'application*** (le modèle de processus est-il présenté comme universel ou spécialisé dans un domaine donné ?) et le ***niveau d'expertise*** de l'équipe. XP (Beck 1999a) se définit comme adapté pour des projets de taille moyenne, avec des équipes de taille moyenne, il propose quelques principes généraux pour la gestion des risques, et accepte le changement à tout moment du processus. Il préconise par ailleurs la présence d'un expert dans l'équipe.
- **La maturité** du modèle de processus, qui est évaluée à travers les ***validations*** que ses auteurs indiquent. Par exemple, (Martin 1991) affirme que RAD s'est avéré supérieur aux modèles de processus « *traditionnels* » dans de nombreux projets, mais ne précise pas comment cette information a été obtenue. Promote propose de mesurer le ***degré de formalisation*** du modèle de processus, c'est-à-dire la quantité de parties décrites avec des langages formels ou semi-formels. La ***diffusion*** du modèle de processus est mesurée à travers le nombre de publications, de sites Internet, de livres qui lui sont consacrés. Enfin, Promote évalue si le modèle de processus comporte une définition de métriques permettant de mesurer sa bonne application, conformément à la proposition de J. Cook et A. Wolf (Cook & Wolf 1999).

Nous ne détaillerons pas davantage ces dimensions, qui n'éclairent pas les travaux que nous présentons ici. La section suivante présente une expérimentation que nous avons menée en vue d'évaluer Promote.

3.5 Evaluation de Promote

Nous avons mené plusieurs expériences pour évaluer différentes versions de Promote au cours des cycles d'induction et de déduction. La figure 6 retrace l'évolution de la définition de la flexibilité dans la taxonomie. La bande verticale au centre de la figure représente le temps, d'Octobre 2010 à Mai 2014. La définition de la flexibilité s'est déroulée au cours de quatre cycles majeurs d'induction et de déduction, dont les fins sont indiquées par les numéros ❶, ❷, ❸ et ❹ sur la figure. La partie de droite présente les différentes versions des critères de la flexibilité :

- la première version, en novembre 2010, comportait deux critères, l'ingénierie (qui reflétait la façon dont le modèle de processus était structuré : monolithique, situationnel ou dynamique) et la prédéfinition (qui mesurait les capacités d'adaptation du modèle de processus) ;

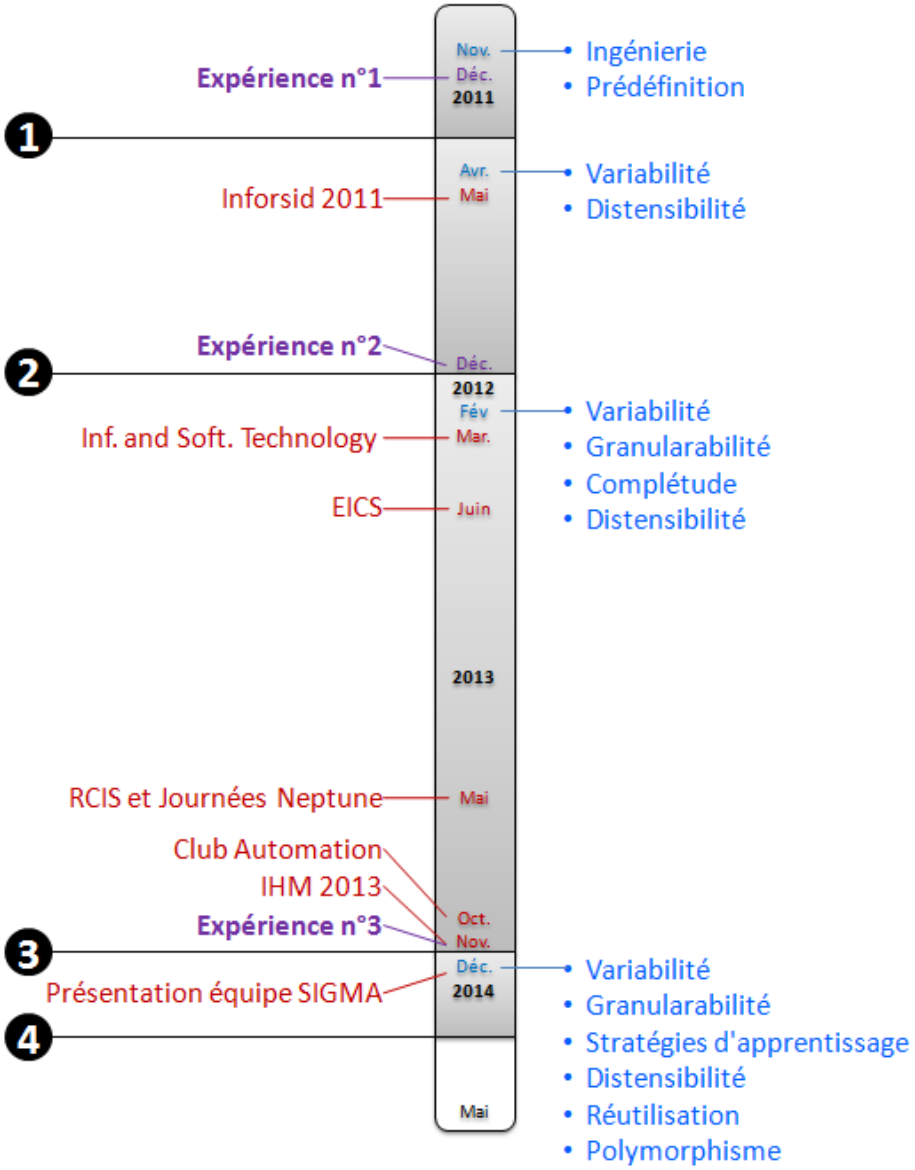

Figure 6 - Ligne de temps de la conception de Promote

- la seconde version, en avril 2011, reformulait la prédéfinition sous le nom de variabilité (définie comme nous l'avons présenté ci-dessus) et transformait l'ingénierie en distensibilité, elle aussi similaire à la définition actuelle ;
- la troisième version, en février 2012, ajoutait la complétude (définie comme la mesure de l'obligation de réaliser l'ensemble du processus et la possibilité d'utiliser des éléments de solutions) et la granularabilité, avec sa définition actuelle ;
- enfin, la dernière version, qui date de fin 2013, transformait la complétude dans sa seule dimension de réutilisation (l'obligation de réaliser tout le processus ayant été analysée comme un sous-ensemble de la variabilité), et ajoutait les stratégies d'apprentissage et le polymorphisme.

La partie gauche de la figure présente les éléments qui ont contribué à ces évolutions : les expériences, dont nous présentons les résultats ci-dessous, ainsi que plusieurs rencontres académiques et industrielles.

La section qui suit présente le protocole expérimental que nous avons mis en œuvre dans les expériences 1 et 2. La troisième expérience sera détaillée ultérieurement, car son principal objet n'était pas d'évaluer Promote, mais d'évaluer les outils que nous avons développés par la suite.

3.5.1 Protocole expérimental

Les hypothèses que nous souhaitions évaluer étaient les suivantes :

H1 : Promote est facile à comprendre, y compris pour des concepteurs peu familiers avec les notions de la taxonomie ;

H2 : Promote est évaluative, c'est-à-dire qu'elle définit les axes et graduations nécessaires pour (a) évaluer un modèle de processus et (b) les comparer entre eux ;

H3 : Promote est pertinente, elle définit des critères utiles et complémentaires.

Nous avons conduit des études de cas sur différentes versions de Promote, en 2010-2011 et 2011-2012, avec 11 étudiants répartis en deux groupes : cinq étudiants dans le premier et six dans le second. Il s'agissait d'étudiants de Master 2 Recherche, qui pouvaient être considérés comme des concepteurs novices, ce qui correspondait bien avec notre objectif d'évaluation, en particulier pour l'hypothèse H1.

Pour les deux premiers groupes, le protocole, élaboré avec N. Mandran qui est la spécialiste en études psychosociologiques du Laboratoire d'Informatique de Grenoble, a été quasiment le même :

- Nous leur avons présenté la taxonomie, en expliquant chaque axe et chaque graduation, et en illustrant nos propos par la classification d'un modèle de processus, en l'occurrence Scrum (Schwaber 1995).
- Nous leur avons ensuite demandé de répondre à un premier questionnaire, qui leur demandait une évaluation de la clarté des définitions des axes et des graduations (hypothèse H1).

- Chaque étudiant a ensuite dû mettre en œuvre la taxonomie en évaluant un modèle ou un métamodèle de processus, choisi parmi Extreme Programming (Beck & Fowler 2000), la MAP (Rolland et al. 1999), Feature Driven Development (Coad et al. 1999), Rational Unified Process (Kruchten 2003), Symphony (Hassine et al. 2002) et le modèle en Spirale (Boehm 1986). Chaque étudiant devait faire une présentation devant le groupe sur un modèle de processus, dans lequel il devait présenter ce modèle, l'évaluer et justifier de son évaluation. Dans le second groupe, nous avons rajouté une tâche : les étudiants devaient s'associer en binôme pour comparer les modèles de processus qu'ils avaient étudiés chacun de leur côté. Ils devaient aussi faire une présentation conjointe de cette comparaison.
- Un débat suivait chaque présentation pour valider si le groupe avait pu comprendre le modèle de processus et la classification proposée et s'ils avaient perçu les avantages et les inconvénients du modèle de processus.
- Enfin, nous leur avons demandé de répondre (anonymement sur Internet) à un questionnaire, dont un extrait est présenté en figure 7, qui portait sur la facilité de mise en œuvre de chaque axe (hypothèse H2a). Les réponses possibles étaient « Tout à fait d'accord », « Plutôt d'accord », « Plutôt pas d'accord » ou « Pas du tout d'accord ». Pour le second groupe, le questionnaire avait été modifié pour ajouter des questions spécifiques sur leur travail de comparaison

Évaluation de la classification

Pour chacune des questions ci-dessous, nous vous demandons d'indiquer votre opinion en sélectionnant la réponse qui vous correspond le mieux.

Pour chacune des affirmations ci-dessous, merci d'indiquer à quel point vous êtes d'accord

	Tout à fait d'accord	Plutôt d'accord	Plutôt pas d'accord	Pas du tout d'accord
La classification vous a conduit à vous poser des questions auxquelles vous n'auriez pas pensé	○	○	○	○
La classification est trop complexe pour une utilisation "en réel"	○	○	○	○
La classification vous a incité à aller chercher plus de documents (articles, revues,…)	○	○	○	○
La classification vous a aidé à identifier les points forts d'une méthode	○	○	○	○
Consulter la classification de plusieurs méthodes vous donnerait des arguments pour en choisir une	○	○	○	○
chacune permet de dire quelle méthodes est "moins forte" ou "plus forte"	○	○	○	○
il y a trop de définitions, trop de critères	○	○	○	○
La classification serait vraiment plus facile à exploiter si on pouvait utiliser un logiciel pour consulter des méthodes déjà classées	○	○	○	○
La classification vous a aidé à identifier les points non traités par une méthode	○	○	○	○
La classification facilite la comparaison de méthodes	○	○	○	○
Classer une méthode est un travail long et subtil	○	○	○	○
La classification vous permet de vérifier si la méthode est adaptée pour votre projet	○	○	○	○
La représentation graphique d'une classification est facile à interpréter	○	○	○	○

Figure 7 - Extrait du questionnaire d'évaluation de Promote

de modèles de processus (hypothèse H2b). Dans ce cas, la comparaison était aussi évaluée axe par axe, de « Très simple » à « Très complexe ». Une seconde partie du questionnaire était destinée à recueillir les opinions sur la pertinence et l'utilité des différents axes et de la taxonomie en général (hypothèse H3), ainsi que sur leur intention de réutiliser Promote ultérieurement. Dans ces questionnaires anonymes, les étudiants étaient aussi invités à exprimer des remarques, à signaler des manques ou des imprécisions, des difficultés éventuelles.

3.5.2 Résultats pour l'hypothèse H1

Après avoir travaillé sur la classification d'un modèle de processus, les étudiants ont estimé qu'ils avaient très bien compris les définitions et, comme le montre le tableau 4, la grande majorité des définitions ont été évaluées comme « Plutôt Claire » ou « Tout à fait Claire », par 282 appréciations sur 305.

	Pas du tout clair	Plutôt pas clair	Plutôt clair	Tout à fait clair	Total
Groupe 1	0	2	48	66	116
Groupe 2	3	18	57	111	189
Total	3	20	105	177	305

Tableau 4. Evaluation de la clarté des définitions, après utilisation de Promote

3.5.3 Résultats pour l'hypothèse H2

Le tableau 5 montre que les étudiants ont estimé que le fait d'appliquer la taxonomie à un modèle de processus les amène à étudier ce dernier en profondeur, à analyser et interpréter les descriptions et à comprendre ce qui est défini et, éventuellement, ce qui manque. Les étudiants ont tous exprimé que l'usage de la taxonomie a fait émerger des questions auxquelles ils n'auraient pas pensé sans cela. Six sont « *tout à fait d'accord* » et trois « *plutôt d'accord* » pour dire qu'après avoir classé un modèle de processus, ils ont pu se faire une meilleure idée de ses points clés. Huit répondants sur neuf pensent que classer un modèle les amène à mieux percevoir ses faiblesses, le dernier estimant que les « *logiciels critiques* » ne sont pas suffisamment pris en compte dans la taxonomie. Les étudiants concluent que l'utilisation de Promote peut les aider à vérifier si un modèle de processus est adapté pour un projet : cinq d'entre eux sont « *tout à fait d'accord* », trois sont « *plutôt d'accord* », et une personne est « *plutôt pas d'accord* ». Les trois personnes « *plutôt d'accord* » expriment une restriction pour expliquer leur pleine adhésion : tous les axes ne leur semblent pas d'une égale importance, et ils aimeraient que la taxonomie leur fournisse une indication sur les poids relatifs des différents axes. Le répondant « *plutôt pas d'accord* » explique ici aussi sa réticence par le manque de prise en compte des logiciels critiques.

Ces éléments confirment donc notre hypothèse H2a : Promote permet d'évaluer un modèle de processus et aide à le comprendre. Elle permet ainsi de mesurer l'adéquation d'un modèle de processus avec un projet.

	Tout à fait d'accord	Plutôt d'accord	Plutôt pas d'accord	Pas du tout d'accord
La classification vous a conduit à vous poser des questions auxquelles vous n'auriez pas pensé				
Groupe 1	2	1	0	0
Groupe 2	4	2	0	0
Total	*6*	*3*	*0*	*0*
La classification vous a aidé à identifier les points forts d'une méthode				
Groupe 1	2	0	1	0
Groupe 2	4	2	0	0
Total	*6*	*2*	*1*	*0*
La classification vous a aidé à identifier les points non traités par une méthode				
Groupe 1	3	0	0	0
Groupe 2	5	1	0	0
Total	*8*	*1*	*0*	*0*
La classification facilite la comparaison de méthodes				
Groupe 1	3	0	1	0
Groupe 2	3	1	0	1
Total	*6*	*1*	*1*	*1*
La classification vous permet de vérifier si la méthode est adaptée pour votre projet				
Groupe 1	1	1	1	0
Groupe 2	4	2	0	0
Total	*5*	*3*	*1*	*0*

Tableau 5. Evaluation de quelques bénéfices de l'usage de Promote

Sur l'hypothèse H2b, les étudiants ont majoritairement estimé que la taxonomie peut être utile pour comparer des méthodes (6 « *tout à fait d'accord* », 1 « *plutôt d'accord* » et 1 « *plutôt pas d'accord* », 1 « *pas du tout d'accord* »), parce qu'elle « *fournit des critères objectifs pour cette comparaison* ». L'un des répondants estime ainsi que Promote « *est utile pour comparer les méthodes et choisir la méthode qui nous convient parce qu'elle entre dans tous les éléments d'une méthode de développement* ». Les personnes ayant répondu « *pas du tout d'accord* » ont exprimé que, selon elles, le travail de classification de plusieurs modèles de processus serait trop long pour être effectué dans « *le monde réel* » et que la classification « *mériterait d'être automatisée afin de ne sélectionner que les axes qui nous intéressent* [dans] *une comparaison* ». Nous verrons dans la section 3.6 que ces remarques nous ont conduits à proposer un outillage de la taxonomie, afin de capitaliser les résultats des évaluations des modèles de processus.

3.5.4 Résultats pour l'hypothèse H3

L'hypothèse H3 propose d'évaluer si Promote est pertinente, c'est-à-dire si elle définit des critères utiles et complémentaires. Dans les questionnaires, nous avons aussi demandé aux étudiants s'ils pensaient que la liste des axes était pertinente, et si certains étaient superflus ou redondants, en leur demandant, le cas échéant, de mentionner quels axes manquaient ou étaient de trop. Le tableau 6 présente les réponses à cette partie du questionnaire.

Cinq étudiants évaluent les critères « plutôt pertinents », deux les estiment pertinents et deux n'ont pas d'opinion. Aucun d'entre eux n'a cependant motivé sa réponse. Nous

La liste des critères vous semble…				
	Incomplète	Plutôt incomplète	Plutôt complète	Complète
Groupe 1	0	0	2	1
Groupe 2	0	0	4	2
Total	*0*	*0*	*6*	*3*

Les critères proposés pour classer les modèles de processus sont d'après vous…					
	Pas du tout pertinents	Plutôt pas pertinents	Plutôt pertinents	Pertinents	Sans opinion
Groupe 1	0	0	2	1	0
Groupe 2	0	0	3	1	2
Total	*0*	*0*	*5*	*2*	*2*

Est-ce que des critères de vous paraissent superflus (si oui, préciser lesquels) ?				
	Oui, plusieurs	Oui, quelques-uns	Non, aucun	Sans opinion
Groupe 1	3	0	0	0
Groupe 2	5	1	0	0
Total	*8*	*1*	*0*	*0*

Est-ce que certains critères vous semblent redondants (si oui, préciser lesquels) ?				
	Oui, plusieurs	Oui, quelques-uns	Non, aucun	Sans opinion
Groupe 1	0	1	2	0
Groupe 2	0	1	5	0
Total	*0*	*2*	*7*	*0*

Tableau 6. Evaluations de la pertinence et de la complétude des axes de Promote

pouvons cependant trouver quelques arguments dans leurs commentaires généraux et dans l'analyse des débats. L'un des étudiants estime ainsi que « *pour une méthode riche en littérature, la classification requiert beaucoup d'efforts. Mais au moment où la classification est terminée, nous avons une étude bien détaillée pour chaque méthode. Donc elle peut être utilisée comme une référence* ». Cinq des six étudiants du second groupe ont aussi estimé que la classification d'un modèle de processus est un travail long et subtil, en particulier parce qu'il y a beaucoup de définitions à étudier. Huit étudiants (sur les neuf qui ont répondu au questionnaire) ont cependant estimé que classer un modèle de processus les conduisait à une découverte plus profonde que lorsqu'ils avaient étudié des modèles sans ce guide, six d'entre étant « *tout à fait d'accord* » et deux étant « *plutôt d'accord* » avec cette idée. D'après eux, chaque axe posait une (bonne) question à laquelle apporter une réponse induisait une compréhension plus fine du modèle de processus. La mise en œuvre de la taxonomie les avait ainsi conduits à être plus exhaustifs dans leur analyse et dans leur recherche de documents que ce qu'ils auraient fait sans cela. Ce retour n'était pas attendu avant l'expérimentation : au cours des années précédentes, dans le même module, les étudiants devaient analyser et décrire des modèles de processus et les appliquer à un petit projet. Le manque de document et le besoin de rechercher plus d'information n'étaient jamais apparus jusqu'à ce que nous leur proposions d'utiliser Promote. Il semble donc bien que l'étude d'un modèle de processus avec la taxonomie induise une investigation approfondie.

En plus de l'effort requis, il y a cependant une autre limite à leur désir de mettre en pratique la classification : quatre étudiants expriment, d'une façon ou d'une autre, que « *réaliser toute la classification est complexe, je pense que nous n'avons besoin de classer que selon les critères requis pour le projet* ». Cependant, lors des débats, ils ont tous acquiescé lorsque l'un d'entre eux a conclu : « *s'il est difficile de faire toute la*

classification, il est très important de s'en servir avant de choisir une méthode pour un projet. Elle nous permet d'analyser clairement les détails de chaque méthode. Ainsi, nous pouvons identifier quelle méthode est la plus adaptée pour le projet ». En d'autres termes, les étudiants ne semblaient pas particulièrement motivés pour classer des modèles de processus, mais intéressés comme utilisateurs de classifications déjà effectuées. Ces remarques confirment l'importance d'outiller la méthode pour capitaliser les efforts de classification et permettre une comparaison selon des listes de critères considérés comme pertinents et sélectionnés librement par l'équipe de développement.

Les étudiants ont été plus critiques à propos de la complétude de la taxonomie. Nous étions intéressés par leurs réponses, principalement pour obtenir des retours sur d'éventuels manques, même si le peu de pratique des étudiants rendait leurs réponses moins pertinentes. Les neufs répondants estiment que la taxonomie est « *plutôt complète* » (6 réponses) ou « *complète* » (3 réponses). Cependant, quatre d'entre eux estiment que des critères sont superflus. Ils mentionnent, avec un répondant pour chacun de ces critères, la formalisation des artefacts, les quantités d'artefacts, la distensibilité du modèle de processus et la proposition de variants. La formalisation des artefacts ne semble pas importante aux yeux de la personne qui estime ce critère superflu : selon lui, l'équipe de développement doit s'adapter à ce que propose le modèle de processus choisi, et le fait que les artefacts soient plus ou moins formalisés relève seulement de la formation à l'approche. Un autre étudiant estime que l'analyse des artefacts est trop détaillée. Comme nous l'avons expliqué dans la présentation des axes autres que la flexibilité (voir section 3.4), les artefacts sont classés comme livrables ou internes, chaque catégorie étant subdivisée en deux : documents ou exécutables. La taxonomie propose d'évaluer la quantité d'artefacts à produire dans chacune de ces quatre catégories, ce qui apparaissait aux yeux du répondant comme trop complexe. Nous détaillerons les retours sur la distensibilité et la variabilité dans la section suivante, consacrée à la flexibilité.

Les étudiants ont conclu leur évaluation de Promote en exprimant qu'ils étaient convaincus de son utilité et qu'ils pensaient « *sûrement* » (7 réponses) ou « *peut-être* » (2 réponses) la réutiliser quand ils en auraient besoin, ce qui, selon nous, témoigne à la fois de la pertinence de la classification et du besoin de la personnaliser selon le contexte du projet dans lequel elle serait mise en œuvre.

3.5.5 Résultats spécifiques pour la flexibilité

Deux versions successives de la flexibilité ont été utilisées dans les expériences. Nous présentons ci-dessous uniquement les résultats du second groupe (de six étudiants), qui a utilisé les définitions actuelles de la variabilité et de la distensibilité.

La définition de la variabilité a été estimée « *tout à fait claire* » par cinq étudiants et « *plutôt claire* » par le dernier. Deux étudiants ont cependant indiqué, lors des débats et dans les questionnaires, qu'ils ne voyaient pas l'intérêt de ce critère. Ils pensaient qu'il y a « *une seule bonne façon de faire* », et qu'il est donc inutile de prévoir des variants. Lors des débats, une étudiante a exprimé que « *même si le modèle de processus ne propose pas*

de variants, il peut quand même être adapté » : elle ressentait une distorsion entre la classification qu'elle avait faite de Feature Driven Development (FDD) (Coad et al. 1999) et la « réalité » du modèle de processus. D'après la classification, FDD ne proposait que quelques variants informels et aucune procédure explicite pour son extension et était donc peu flexible. D'après les forums sur Internet qu'elle avait consultés, l'étudiante s'était forgé l'impression que l'une des grandes forces de FDD était au contraire sa flexibilité. Cette étudiante se basait donc sur ce qu'elle avait lu des expériences d'application du modèle de processus, et non sur les prescriptions de ce dernier, qui sont seules prises en compte dans la taxonomie. Cette intervention nous a amenés à préciser, à chaque fois que nous parlons de cette taxonomie, que nous évaluons ce qui est dit dans le modèle de processus et non ce que les gens ont l'habitude d'en faire.

La définition de la distensibilité est estimée « *tout à fait claire* » par cinq étudiants et « *pas du tout claire* » par le sixième. Cet étudiant indique que ce critère est superflu, car une méthode « *doit être complète* » et il n'y a donc aucune raison de vouloir l'étendre.

Par ailleurs, les étudiants, qui étaient pendant l'expérimentation considérés comme des concepteurs novices, ont insisté sur la difficulté de « *rentrer dans un modèle de processus* », le plus souvent en raison d'un vocabulaire mal maîtrisé, d'un « *jargon* » difficilement accessible, d'activités trop complexes pour être perçues dans leur ensemble. Ces échanges nous ont amené à approfondir l'étude des besoins des concepteurs et développeurs, car nous n'avions pas identifié ces besoins dans un premier temps. Ils étaient masqués par la place importante que les études attribuent à la variabilité. Nous avons ainsi pu identifier les besoins de disposer de différents niveaux d'information (besoin numéro 4) et de différents niveaux de complexité (besoin numéro 5), ce qui nous a permis de définir les critères de granularabilité et de stratégies d'apprentissage.

La remarque d'un étudiant sur le fait qu'une méthode « *doit être complète* » nous a amenés à réfléchir sur la complétude des modèles de processus. L'axe de la complétude a été ajouté dans la troisième version de la flexibilité. Cet axe était défini comme (1) la possibilité de ne pas traiter une partie du modèle de processus et (2) la possibilité de réutiliser des éléments existants. Après avoir mûri notre réflexion, il nous est apparu que la première forme de complétude était en fait une forme de variabilité : la partie potentiellement non traitée pouvait être considérée comme une option, un variant dans les chemins possibles. La seconde partie de la complétude a été conservée sous la dénomination de réutilisation, un autre besoin des concepteurs et développeurs que nous n'avions pas initialement identifié.

Le polymorphisme est le dernier ajout à la dimension de la flexibilité. Il n'a cependant par émergé de cette expérimentation, mais des retours des relecteurs et des autres chercheurs, notamment grâce à l'un des relecteurs qui nous a chaudement encouragés à étudier les limites des approches centrées uniquement sur les activités, ce qui nous a amenés à ajouter le critère sur les différentes perspectives sur le processus.

Ces expériences nous ont donc été extrêmement utiles. Les analyses quantitatives des questionnaires et les analyses qualitatives des présentations et débat (qui étaient, comme

nous l'avons dit, enregistrés), ont mis en lumière les réactions des étudiants lors d'une utilisation réelle de la taxonomie, faisant ainsi ressortir les axes qui avaient besoin d'être raffinés ou retravaillés. Les résultats sont limités par le petit nombre de participants, qui ne permet pas de tirer des conclusions définitives sur la taxonomie, et par le fait que la dernière version de la flexibilité n'a pas pu être évaluée. Par ailleurs, un biais potentiel se situait dans le fait de mener une expérience avec des étudiants que nous devions noter sur leur bonne application de la taxonomie. Les questionnaires anonymes, qui permettaient aux étudiants de s'exprimer librement et sans crainte, limitent cependant ce biais. De même, le pouvoir génératif de la taxonomie n'a pas été évalué ici. Mais comme nous le verrons plus tard, nous avons pu obtenir quelques retours lors de l'évaluation de nos outils de conception de modèles de processus (voir section 6.4).

La section suivante présente une réponse à l'un des besoins exprimés par les étudiants : économiser les efforts de classification à l'aide d'un outil permettant de les capitaliser.

3.6 Outillage de Promote

La classification d'un modèle de processus représente un travail relativement long : il est nécessaire d'étudier précisément ce qu'il propose et de confronter cette étude aux définitions et graduations des 36 critères de Promote. Il est donc difficile pour les équipes de développement désireuses de sélectionner une approche de trouver le temps d'en comprendre et d'en analyser plusieurs. Il semblait donc intéressant de proposer un outil où ces classifications pourraient être centralisées et accessibles librement. Nous avons développé un site Internet dédié à la présentation de Promote et à la classification des modèles de processus, www.design-methods.net.

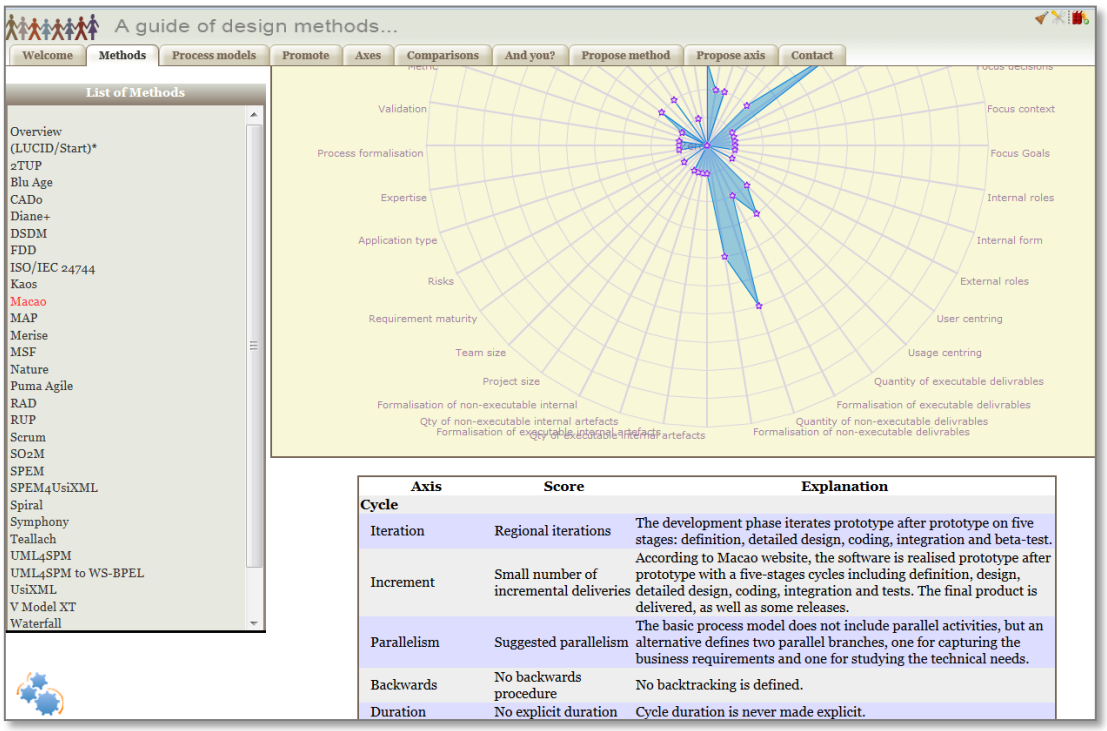

Figure 8 - La page de la méthode Macao (Crampes 2002) dans design-method.net

Ce site est constitué de deux modules : l'un, avec un accès restreint (administrateur), pour définir les axes, les graduations et les classifications des méthodes et l'autre, dont on peut voir une copie d'écran sur la figure 8, avec un accès public, pour consulter ces mêmes éléments. L'utilisateur peut consulter différents onglets :

- Le premier onglet, Methods (Méthodes), permet d'avoir un descriptif général des méthodes et d'accéder à la description de chacune des méthodes classées dans le site. Chaque description comporte (a) la description du cycle de vie de la méthode, (b) la classification du modèle de processus sous forme textuelle avec l'explication de la classification pour chaque axe et (c) une représentation graphique de cette même classification. La figure 8 montre une partie de la page dédiée à la méthode Macao (Crampes 2002), on aperçoit le bas du diagramme de Kiviat représentant graphiquement la classification du modèle de processus, ainsi que le début du tableau présentant les explications sur cette classification ;
- Le second onglet (« process models », c'est-à-dire modèles de processus) donne des explications sur ce qu'est un modèle de processus ;
- L'onglet « Promote » donne une vision globale de la taxonomie et de ses objectifs ;
- L'onglet « Axes » donne la définition de chacun des axes de Promote et permet une visualisation de la classification de tous les modèles de processus inclus dans le site pour chaque axe. L'utilisateur peut ainsi comparer facilement les différents modèles selon les critères qui l'intéressent.
- L'onglet « Comparisons » (comparaisons) permet d'obtenir différentes représentations, graphiques ou textuelles, d'un sous-ensemble de modèles de

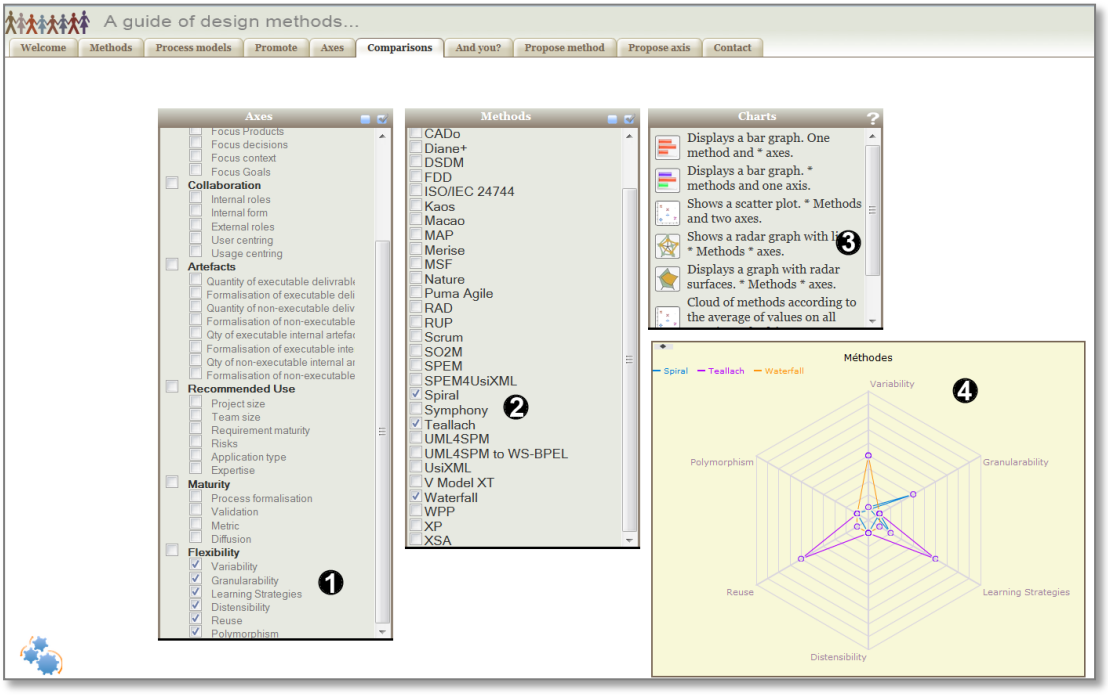

Figure 9 - Exemple de comparaison de modèles de processus selon différents axes

processus en fonction d'un sous-ensemble d'axes. La figure 9 montre un exemple d'une telle comparaison : l'utilisateur a demandé une comparaison des modèles de processus du Waterfall, du modèle en Spirale et de Teallach ❶[12] selon les six dimensions de la flexibilité ❷. Il a demandé à avoir une représentation sous forme d'un diagramme de Kiviat ❸. Ce diagramme est généré puis affiché ❹. Design-methods offre plusieurs formes de comparaisons : graphique ou textuelle. La représentation graphique présente l'avantage d'une comparaison visuelle rapide, mais n'offre pas d'argument expliquant la classification. Ainsi, lorsque l'utilisateur compare les approches des modèles de processus, la graduation (approche ascendante, approche descendante ou approche mixte) n'est pas ordonnée, alors que le graphique peut donner l'impression qu'un modèle de processus est « supérieur », du fait qu'il couvre une aire plus importante. Pour contrebalancer ces défauts des représentations graphiques, Design-methods comporte aussi la possibilité de générer un tableau textuel avec les évaluations des modèles de processus comparés ainsi que les justifications de ces évaluations (voir figure 10).

- Les onglets suivants permettent au visiteur de faire des propositions concernant l'évolution du site ou de la taxonomie. Il peut ainsi proposer une méthode qui manquerait, un axe qui lui paraîtrait intéressant ou envoyer un message pour toute autre remarque ou demande. Ces suggestions, si elles s'avèrent pertinentes, peuvent être intégrées dans le site. Il peut aussi saisir des commentaires sur chaque page, permettant ainsi de compléter les informations présentées.

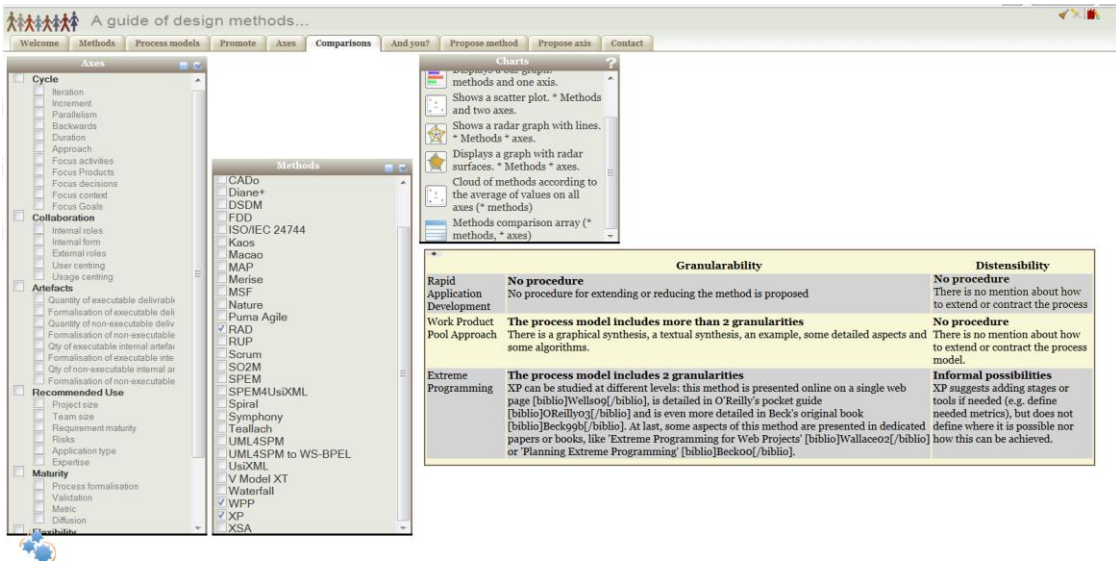

Figure 10 - Comparatif textuel de modèles de processus

[12] Les numéros dans le texte renvoient aux mêmes numéros sur la figure.

Résumé du chapitre 3

Dans ce chapitre, nous avons présenté Promote (**Pro**cess **Mo**dels **T**axonomy for **E**nlightening choices), une taxonomie des modèles de processus comportant six dimensions principales : le **cycle de vie**, la **collaboration**, les **produits**, l'**usage recommandé**, la **maturité**, la **flexibilité**.

Nous avons détaillé la dernière dimension, celle de la flexibilité, qui est au cœur des travaux que nous présentons ici, et qui se décline en six critères : la **variabilité**, la **granularabilité**, les **stratégies d'apprentissage**, la **distensibilité**, la **réutilisation**, le **polymorphisme**.

Nous avons présenté les expériences que nous avons menées en vue d'évaluer les pouvoirs descriptifs et évaluatifs de la taxonomie avec un groupe de cinq étudiants et un groupe de six étudiants, tous inscrits en Master 2 Recherche. Nous avons montré que leurs réactions sont favorables, aussi bien en termes de facilité de compréhension de la taxonomie et de ses axes, qu'en terme d'utilisation : toujours d'après les conclusions de cette expérience, Promote permet d'évaluer un modèle de processus et de les comparer entre eux, elle est aussi pertinente, au sens où elle définit des critères utiles. Nous avons indiqué que les limites de cette expérimentation proviennent du petit nombre de participants et du fait que nous n'avons pas été en mesure d'évaluer la dernière version de la flexibilité.

Enfin, nous avons présenté le site internet qui opérationnalise Promote et permet ainsi de capitaliser les efforts de classification des modèles de processus et de comparer les propositions lors du choix d'une approche pour un projet donné.

4 ANALYSE CRITIQUE DE LA FLEXIBILITE DE QUELQUES MODELES DE PROCESSUS

Dans cette section, nous évaluons les formes de flexibilité de plusieurs modèles de processus, issus des domaines du Génie Logiciel et des Interfaces Homme-Machine. Nous avons sélectionné ceux qui ont illustré nos propos dans les chapitres précédents, le modèle du Waterfall (Royce 1970), Rapid Application Development (Martin 1991) et Extreme Programming (Beck 1999a)[13], auxquels nous avons ajouté les modèles qui apportent des éclairages variés sur les différentes formes de flexibilité, ainsi que ceux dont les auteurs mentionnaient une attention particulière à la flexibilité.

Pour clarifier le positionnement de cette étude, nous rappelons que nous nous focalisons ici sur les méthodes de développement comportant un modèle de processus au sens où nous l'avons défini. Ceci exclut un certain nombre de suggestions qui nous ont été faites à plusieurs reprises comme, par exemple, la conception centrée utilisateur (International Organization for Standardization 1999), qui ne comporte pas de modèle de processus en tant que tel mais des principes pour intégrer les utilisateurs dans un modèle de processus, et les méthodes d'inspection de l'utilisabilité (Nielsen 1994), qui sont des méthodes d'évaluation et non des méthodes de développement. A l'inverse, nous avons fait le choix d'évaluer des métamodèles de processus, qui ne sont pas des modèles de processus, mais qui, en permettant la conception de ces derniers, définissent la flexibilité potentielle qu'ils pourront comporter. Ainsi, un métamodèle qui ne comporterait pas, sous une forme ou sous une autre, de décomposition arborescente des activités en sous-activités, ne permettrait pas aux modèles de processus qui l'instancieraient de comporter différents niveaux de détails.

Ce chapitre commence par une présentation de notre méthode de classification des modèles de processus, puis présente sa mise en œuvre sur les (1) approches prescriptives, qui définissent complètement un modèle de processus, (2) les approches agiles, qui sont intermédiaires et sous-spécifient le modèle de processus et enfin (3) les approches constructivistes, celles qui proposent de construire un modèle de processus. Les approches situationnelles et les métamodèles de processus entrent en particulier dans cette dernière catégorie.

4.1 Classification de modèles de processus

Un des éléments importants pour appliquer Promote, notre taxonomie des modèles de processus, est la méthode de conduite de l'analyse des modèles de processus. L'indicateur premier pour catégoriser un modèle de processus est ce que le modèle de processus lui-même exprime. Ainsi, Extreme Programming (Beck 1999a) est considéré comme ne proposant pas une approche ascendante ni descendante car l'auteur revendique clairement une « *autre approche* », même s'il ne la définit par concrètement.

[13] Ces modèles de processus ont été présentés dans la section 3.1

Ce n'est qu'à défaut d'indication que la classification se fait sur une interprétation. Par exemple, Royce ne spécifie pas quelle est l'approche recommandée dans le Modèle du Waterfall (Royce 1970), mais l'analyse de la démarche proposée montre qu'il suggère de décomposer les problèmes, ce qui correspond à une approche descendante.

4.2 Les approches prescriptives

Nous nommons cette catégorie d'après le terme utilisé dans (Gonzalez-Perez & Henderson-Sellers 2008). Nous trouvons ici des approches « classiques », développées depuis les années 1960, qui définissent un modèle de processus complet, détaillé. Nous présentons ci-dessous les évaluations du modèle du Waterfall (Royce 1970), qui a été mentionné comme particulièrement rigide par de nombreux auteurs comme (Glass 1969), (Boehm 1979), (Mills 1971) ou (Prakash 1997), du modèle en Spirale (Boehm 1986), qui a été proposé pour dépasser les limites des modèles précédents, en particulier celui du Waterfall. Nous terminerons cette catégorie par l'évaluation de Rapid Application Development (Martin 1991), car cette approche revendique une forte capacité à s'adapter au contexte dans lequel elle est mise en œuvre, ce qui a amené des auteurs comme (Anderson 2004) ou (Abbas et al. 2008) à la considérer comme l'origine des méthodes agiles. Enfin, comme nous nous intéressons particulièrement aux modèles de processus permettant la conception d'IHM, nous présenterons aussi l'évaluation des modèles de processus de Macao (Crampes 2002) et de (Lucid/Star)* (Helms 2001), dont les auteurs mentionnent une préoccupation particulière pour la flexibilité de leur proposition.

4.2.1 Le Modèle du Waterfall

Ce modèle ayant été présenté dans la section 3.1.1, nous ne reprenons pas ici l'explication de cette proposition.

Le modèle du Waterfall (Royce 1970) propose un variant, la séparation du projet en deux temps (une phase de prototypage pour les parties critiques du projet, suivie du développement du projet complet) lorsque « *le programme est développé pour la première fois* », ce qui est une condition bien définie. Le modèle du Waterfall ne propose qu'un niveau de granularité. En effet, même s'il comporte de nombreux schémas et des descriptions qui les expliquent, tous les éléments sont au même niveau de détail, celui des grandes étapes du processus. Il ne présente pas d'informations particulières que nous pourrions considérer comme des stratégies d'apprentissage qui complèteraient les explications données, et ne mentionne pas de possibilité d'extension ou de contraction. Il n'offre pas non plus d'éléments de solutions réutilisables ou enrichissables, et propose une seule vision du processus, centrée sur les activités à réaliser.

L'évaluation de sa flexibilité est la suivante :

Variabilité :	Rares variants bien définis
Granularabilité :	Une granularité
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Pas de procédure
Réutilisation :	Pas d'élément de solution

Polymorphisme : Une seule vision

Cette évaluation est représentée sur la figure 11. La figure représente les six critères de la flexibilité, l'origine étant sur l'hexagone rouge au centre (pour éviter d' « écraser » les aires lorsque de nombreuses valeurs sont minimales), l'hexagone vert représentant les valeurs maximales. Les graduations des six critères composant la flexibilité sont ordonnées, ce qui permet de lire directement sur le graphique à quel point le modèle de processus répond à chacun des critères considérés. On peut ainsi lire directement que le modèle du Waterfall présenté ici n'offre pas de flexibilité en termes de granularabilité, de stratégies d'apprentissage, de distensibilité, de réutilisation et de polymorphisme, mais qu'il propose une flexibilité « moyenne » en termes de variabilité. Dans la suite de ce chapitre, nous présenterons les représentations graphiques de l'évaluation de chaque approche, afin, dans la conclusion, de montrer comment ces représentations, que l'on trouve aussi dans l'outillage de la taxonomie, peuvent permettre une comparaison rapide de différents modèles de processus.

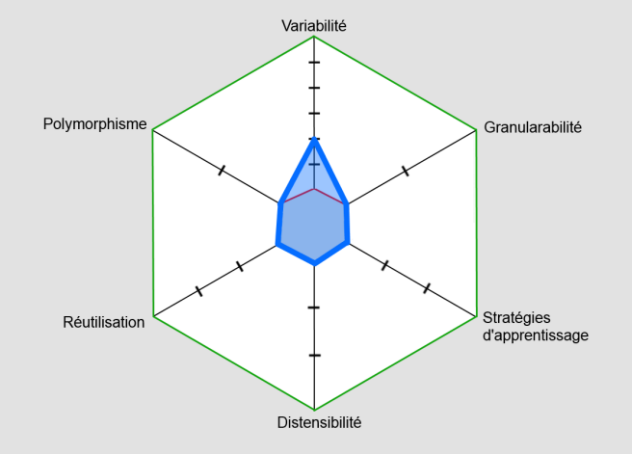

Figure 11 - Représentation graphique de la classification du modèle du Waterfall

4.2.2 Le Modèle en Spirale

Le modèle en Spirale (Boehm 1986), présenté sur la figure 12, repose sur un cycle itératif. Chaque itération repose sur des étapes d'évaluation des risques, de prototypage et simulations, de spécification ou de conception, de validation et se termine par la définition des itérations suivantes. Les besoins ne sont plus exprimés en une fois au début du projet, mais pris en compte peu à peu au fil des itérations. L'étude des risques et la phase de validation permettent d'identifier les problèmes au fur et à mesure de l'avancement du projet et de les corriger rapidement si besoin.

Le modèle en Spirale ne comporte pas de variants dans sa version initiale. Cependant, son auteur a rédigé de nombreuses publications, précisant les concepts de base (Boehm 1989), rajoutant des variants (Boehm & Hansen 2001) ou proposant des extensions du modèle de processus initial (Boehm et al. 1998). En prenant en compte ces différents compléments, le modèle en Spirale propose quelques variants portant par exemple sur le niveau d'utilisation des prototypes selon le degré de technicité du projet ou sur le choix de la

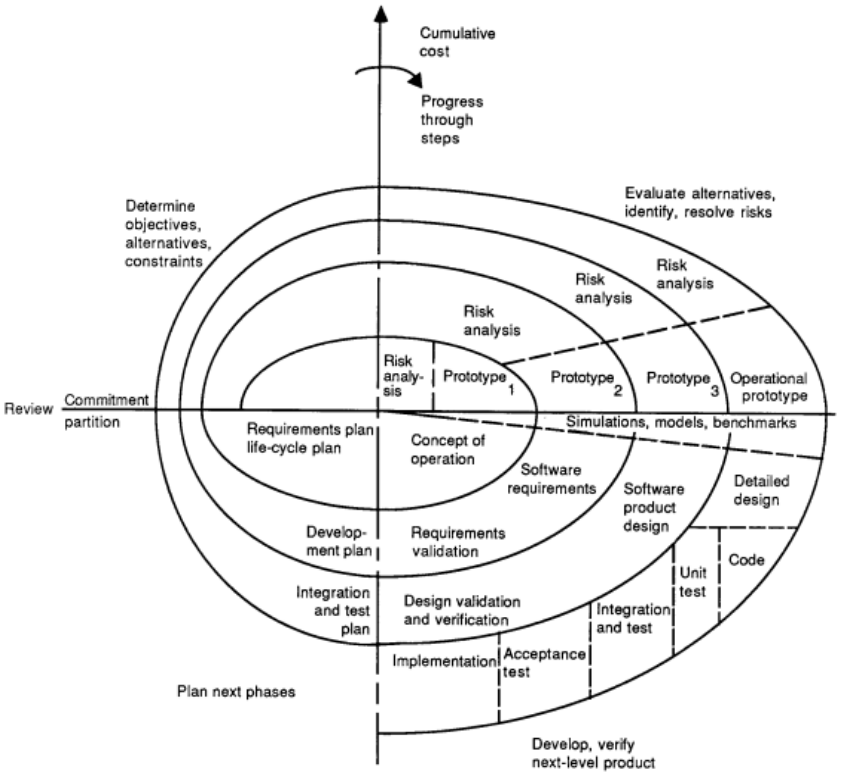

Figure 12 - Le modèle en Spirale

méthode d'évaluation des risques. Les critères de choix d'un variant sont informels : la quantité de prototypes dépend par exemple d'une technicité qui peut être « *basse* » ou « *haute* ». L'ensemble des publications permet d'avoir des informations détaillées sur certaines activités, apportant une seconde granularité au modèle de processus initial. L'auteur argumente la validité et la pertinence de son modèle de processus au niveau conceptuel, et s'il fournit quelques exemples (Boehm & Hansen 2001), c'est principalement pour illustrer son propos, mais de façon insuffisante pour permettre un véritable apprentissage de l'approche. Il ne propose pas non plus de procédure concrète pour étendre ou réduire le modèle de processus, suggérant de façon très informelle d'évaluer « *le juste équilibre entre des choix trop rapides et des investigations trop longues et parfois peu utiles* » (Boehm & Hansen 2001). Ceci suggère d'écourter ou d'allonger les études mais sans modifier la structure du modèle de processus. Enfin, dans toutes les publications que nous avons consultées, l'auteur reste dans une vision centrée sur les activités et ne propose jamais de représentation avec une autre focale.

L'évaluation de sa flexibilité est la suivante :

Variabilité :	Quelques variants informels
Granularabilité :	Deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Pas de procédure
Réutilisation :	Pas d'élément de solution
Polymorphisme :	Une seule présentation

La représentation graphique de cette évaluation est présentée sur la figure 13.

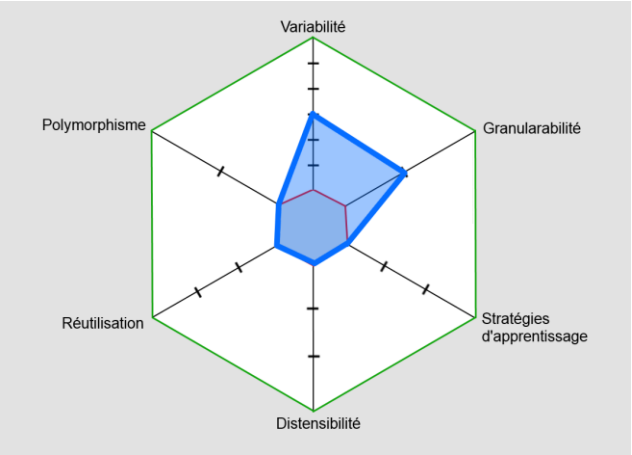

Figure 13 - Représentation graphique de la classification du modèle en Spirale

4.2.3 Rapid Application Development

Rapid Application Development (Martin 1991) est considérée, comme nous l'avons dit plus haut, comme l'une des méthodes fondatrices des concepts agiles (Anderson 2004, Abbas et al. 2008). RAD revendique en effet une forte capacité à s'adapter à la situation dans laquelle elle est mise en œuvre. Nous avons détaillé ce modèle de processus dans la section 3.1.2, nous ne reprendrons pas cette présentation ici.

RAD propose plusieurs variants. Par exemple, selon la taille et la complexité du projet, le cycle est plus moins long (entre 60 et 120 jours), l'équipe a aussi la possibilité d'utiliser une « *timebox* » (une structuration temporelle imposant des livraisons à des dates fixées), et les étapes de planification et de conception peuvent être fusionnées si le domaine est bien maîtrisé. RAD comporte de plus une première phase de préparation, destinée à faire des choix parmi les variantes proposées et à sélectionner des outils. Ces variants sont présentés de façon explicite, dans des cartes méthodologiques (sous forme d'algorithmes), avec des critères bien définis. Cette approche est donc évaluée comme proposant quelques variants bien définis.

Dans le livre de référence sur RAD, le modèle de processus est tout d'abord présenté de façon globale au chapitre 1, puis détaillé au fil des chapitres suivants et enfin précisé de façon très fine dans des cartes méthodologiques (« *methodology charts* ») qui présentent le travail à effectuer et les éléments à prendre en compte dans les différents choix, sous une forme quasi-algorithmique. RAD offre ainsi plus de deux granularités.

Ces deux descriptions des activités, en langage naturel et sous la forme d'algorithmes, sont bien exprimées dans des langages différents. Cependant, les cartes méthodologiques n'ont pas pour objectif de faciliter l'apprentissage du modèle de processus, elles sont destinées à présenter un niveau de détail supplémentaire dans un langage adapté. Si nous nous référons aux quatre types d'apprenants de Kolb (Kolb 1976), ces deux descriptions se focalisent sur la conceptualisation du modèle. Nous ne considérons donc pas ces deux langages comme deux stratégies d'apprentissage distinctes. A l'inverse, les annexes offrent 250 pages d'explications et d'exemples sur certaines activités du modèle de

processus, comme le calcul des points de fonction (dans l'estimation des charges de travail) ou la modélisation des données. Ces annexes sont donc des éclairages sur des expérimentations, des mises en œuvre du modèle de processus. Elles permettent un processus d'observation et constituent une stratégie d'apprentissage distincte par rapport aux informations sur le modèle de processus proprement dit.

Ce modèle de processus n'offre aucune procédure au sujet de son éventuelle extension ou réduction, il n'est donc pas distensible.

RAD promeut la réutilisation et propose de construire des composants de façon à les rendre réutilisables. Cependant, il ne propose pas d'élément de solution dont les concepteurs et développeurs pourraient s'inspirer. Il ne propose pas non plus de présentation centrée sur une autre focale que les activités.

L'évaluation de sa flexibilité est la suivante :

Variabilité :	Quelques variants bien définis
Granularabilité :	Plus de deux granularités
Stratégies d'apprentissage :	Deux stratégies d'apprentissage
Distensibilité :	Aucune procédure
Réutilisation :	Pas d'élément de solution
Polymorphisme :	Une seule présentation

La représentation graphique de cette évaluation est présentée sur la figure 14.

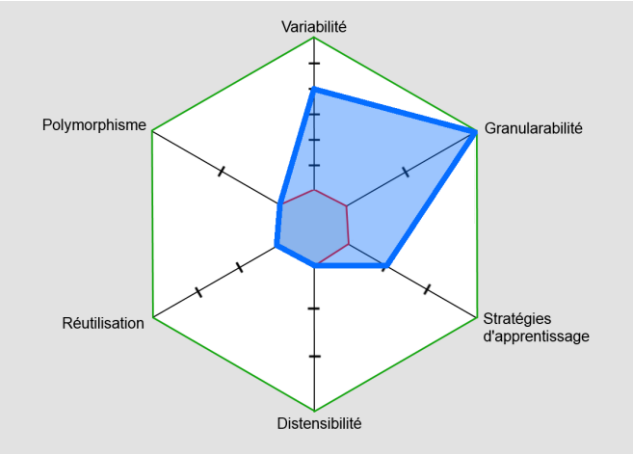

Figure 14 - Représentation graphique de la classification de Rapid Application Development

4.2.4 La méthode MACAO

La méthode MACAO (Crampes 2002) présente l'originalité de traiter à la fois de la conception du noyau fonctionnel et de la conception des IHM, en intégrant une approche centrée sur l'utilisateur. Cette volonté d'universalité (selon le terme de l'auteur) est suffisamment rare dans les modèles de processus pour qu'il soit intéressant d'en mesurer la flexibilité.

Cette méthode a été présentée initialement dans le livre de J.-B. Crampes. Il s'agit d'une méthode orientée objets basée sur les modèles, qui supporte en particulier les

modèles UML (Object Management Group 2011) et définit des modèles pour la conception d'IHM. En 2008, une thèse (Ferry 2008) dirigée par l'auteur a permis de créer un métamodèle pour les IHM, afin de faciliter leur définition et leur maniement, de créer un éditeur graphique pour les concevoir, et de proposer un outil de génération d'IHM par transformation de modèles. Un certain nombre d'éléments complémentaires sont détaillés sur le site Internet de Crampes (Crampes 2008). Considérant que ce site est daté de 2008, soit bien après la publication initiale de la méthode, et qu'il n'y a pas eu de seconde publication de la méthode, nous considérons que les éléments qu'il apporte constituent un raffinement de la méthode initiale et font partie de notre évaluation.

MACAO repose sur une démarche participative, elle est itérative et incrémentale et repose sur la réalisation successive de prototypes. Elle met en œuvre, comme nous l'avons dit, des modèles UML et propose des modèles pour concevoir l'IHM sur la base de patrons : le Schéma Navigationnel d'Interfaces (SNI), qui est raffiné en Schéma d'Enchaînement des Pages (SEP) pour les IHM de type web et en Schéma d'Enchaînement des Fenêtres (SEF) pour les « *IHM de type fenêtre* ». Elle intègre aussi des activités pour l'évaluation des charges (la quantité de travail requise pour accomplir les tâches) et des risques (Ferry 2008). Le cycle de vie du modèle de processus de MACAO est représenté sur la figure 15. Il se décompose en quatre phases principales : (1) l'analyse globale, destinée à cerner les besoins des utilisateurs, (b) la conception globale, qui permet la conception de l'architecture fonctionnelle et interactive du logiciel, (3) le développement, qui est découpé en itérations, chacune d'entre elles conduisant à la création d'un prototype et à son évaluation par le client, et (4) la finalisation, dont le résultat est la documentation finale et la livraison du logiciel complet.

MACAO propose plusieurs formes de variants. Selon les contraintes et la taille du

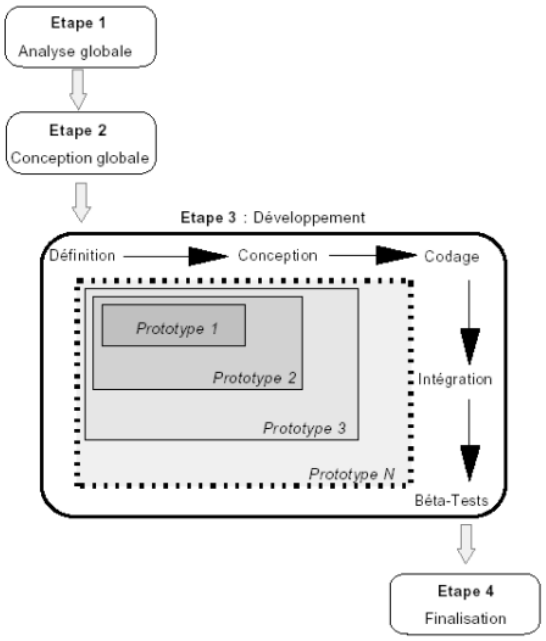

Figure 15 - Cycle de vie de MACAO (Ferry 2008)

projet, il est possible de mettre en œuvre un cycle en Y avec deux branches parallèles pour l'étude des besoins fonctionnels et l'étude de l'architecture technique. L'auteur précise que, dans le cas de petits projets (moins de 100 fonctions), le découpage en prototypes est inutile, et une dizaine de tâches sont exclues du modèle de processus ou confondues avec d'autres. En d'autres termes, la méthode MACAO prévoit que certaines tâches soient optionnelles selon la taille du projet. De même, l'étude d'opportunité peut être réduite à une étude de coût si le projet est indispensable. En termes de granularités, le modèle de processus est présenté sur la forme de diagrammes et de descriptions en langage naturel, c'est-à-dire avec deux niveaux de détails. Les différents modèles et activités sont largement expliqués, mais il n'y a pas d'autre élément que nous puissions considérer comme une seconde stratégie d'apprentissage. Les auteurs ne mentionnent pas de possibilité de modifier dynamiquement le modèle de processus. En termes de réutilisation, si la méthode insiste fortement sur la création de composants réutilisables, elle ne propose pas d'éléments de solutions, à l'exception du niveau procédural, puisque quelques patrons de conception sont présentés, comme le « *patron racine* » qui permet de définir les différents niveaux de l'arborescence des interfaces, ou le « *patron détails* » qui précise que lorsqu'un élément n'est pas affiché complètement (par exemple, une liste qui comporte uniquement le nom et le prénom d'une personne, mais pas ses autres attributs), il est nécessaire d'ajouter un élément de navigation permettant d'accéder à la totalité de l'information (la fiche personne dans notre exemple). MACAO est classée comme proposant de rares éléments de solutions. Enfin, le modèle de processus est articulé exclusivement sur les activités à mener et ne propose pas d'autre point de vue.

L'évaluation de la flexibilité de l'approche MACAO est donc :

Variabilité :	Quelques variants bien définis
Granularabilité :	Deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Aucune procédure
Réutilisation :	Quelques rares éléments de solution
Polymorphisme:	Une seule vision sur le processus

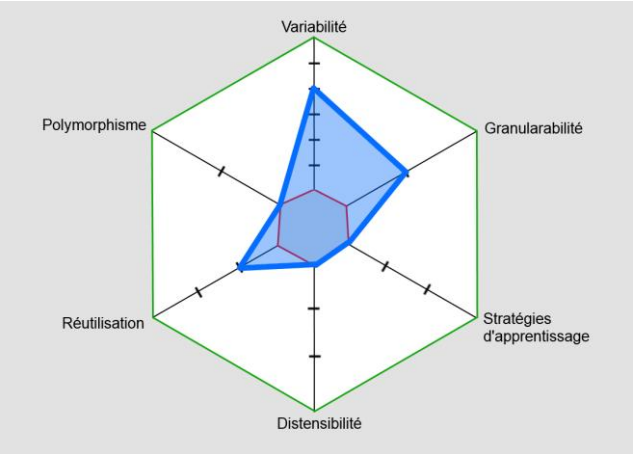

Figure 16 - Représentation graphique de la classification de MACAO.

La figure 16 montre la représentation graphique de cette classification.

4.2.5 (Lucid/Star)*

La méthode (Lucid/Star)* nous a intéressés car l'auteur, J. Helms, part du constat que, du fait de la prise en compte croissante des besoins en utilisabilité dans la conception des systèmes interactionnels, de nombreux modèles de processus ont émergé, mais que ces modèles de processus manquent (entre autres) de flexibilité et de possibilités de personnalisation (Helms 2001), ce qui est le cœur de nos préoccupations. Pour traiter ces manques, l'auteur propose un nouveau modèle de processus, qu'il nomme (Lucid/Star)*, qui s'inspire de l'approche de conception d'interaction Lucid (Smith & Dunckley 1998), du modèle d'ingénierie de l'utilisabilité Star Life Cycle (Hix & Hartson 1993) et des modèles de processus du Waterfall (Royce 1970) et de la Spirale (Boehm 1986).

Cette approche repose sur la volonté d'offrir « *une structure flexible, non-linéaire et itérative* », ce qui conduit J. Helms à proposer des itérations imbriquées, à la fois dans le cycle complet et à l'intérieur cycle, sur un sous-ensemble d'activité, comme le représente la figure 17 : un cycle global est divisé en cycles régionaux, qui se succèdent selon une rotation horaire. Chaque cycle régional est lié au cycle suivant et à une évaluation globale au centre du cycle global, qui permet si besoin de revenir à une étape précédente ou de sauter à une étape ultérieure. Ainsi, après avoir réalisé la conception des écrans ❶, l'équipe peut passer à l'étape de prototypage de basse fidélité ❷ ou évaluer son travail ❸ et peut par exemple revenir à l'étape précédente de conception du produit ❹, aller à

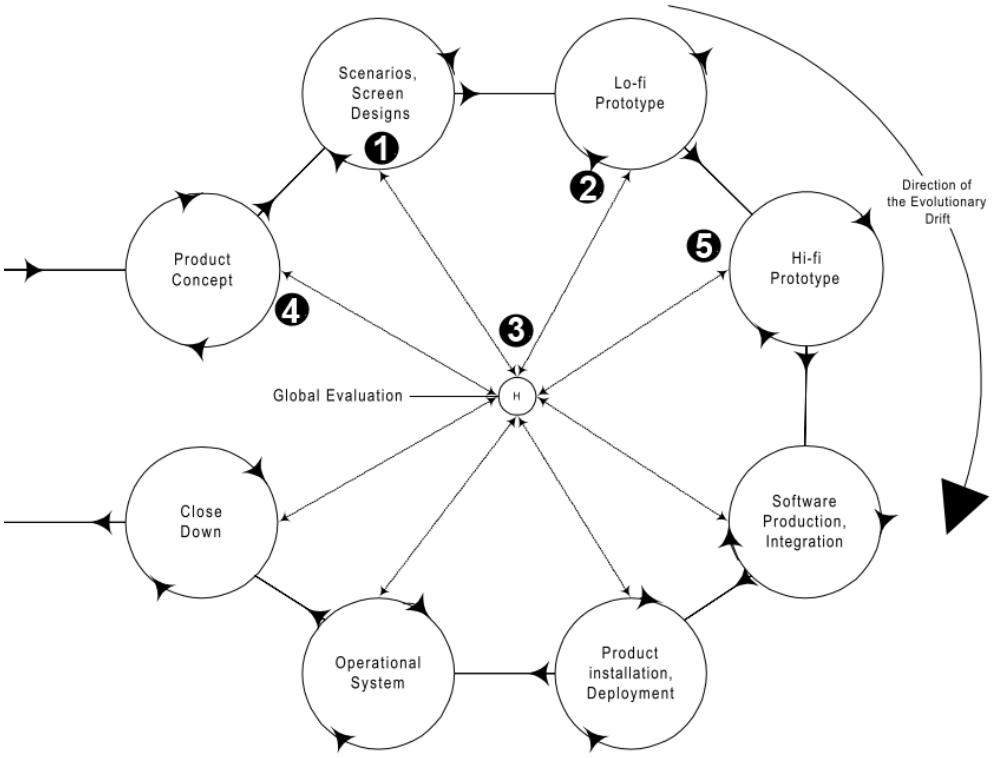

Figure 17 - Cycle de vie de (Lucid/Star)*.

l'étape suivante de prototypage à basse fidélité ❷ ou passer directement au prototypage de haute fidélité ❺.

Les cycles régionaux sont raffinés en « *types d'activités* » : chaque cycle est structuré autour de quatre types d'activités : l'analyse, la conception, l'implantation et l'évaluation. Les activités concrètes, c'est-à-dire les instances des types d'activités, ne sont pas prédéfinies, elles sont choisies par l'équipe de développement, soit parmi les propositions incluses dans (Lucid/Start)*, soit à partir des activités déjà connues par l'équipe de développement. Un certain nombre de variants sont donc bien définis : ceux proposés dans l'approche. Les autres sont inconnus. Par ailleurs, la condition pour choisir parmi les variants (« *Les développeurs peuvent et devraient inclure ou éviter toute activité ou tout cycle en fonction de leur objectif, de la composition de l'équipe, du style de gestion de projet, du budget ou du calendrier attendu* ») n'est pas clairement définie, au sens où nous l'avons exprimé dans la section 3.3.1. Cette approche offre donc de nombreux variants informels.

On trouve donc dans cette proposition trois granularités : le cycle global, les cycles régionaux, les activités. L'approche présente une explication globale du fonctionnement des cycles et de leur personnalisation. Quelques activités concrètes sont présentées à titre d'illustration. Il n'y a donc pas de stratégie d'apprentissage du modèle de processus en dehors des explications sur sa structure.

J. Helms estime que la structure du cycle permet de « *greffer des méthodes ou des techniques déjà familières dans les types d'activités* », puisque la définition du processus permet d'instancier les types d'activités avec des activités librement choisies, ce qui correspond à la flexibilité par sous-spécification de (Mulyar et al. 2007), que nous avons vue en section 2.5.4. Cependant, la procédure pour réaliser ces « greffes » ne peut se résumer à une instanciation de type d'activité, il est nécessaire de vérifier la cohérence du modèle de processus. Les procédures pour effectuer cette vérification ne sont pas indiquées : il s'agit de moyens informels d'extension et de réduction.

Aucune des structures présentées ne permet de définir comment réutiliser des éléments de solution. La mise en œuvre d'activités déjà connues ne repose pas sur la modélisation de ces activités : il n'y a pas d'entité concrète réutilisée, mais plutôt la création d'une nouvelle entité (une activité) à partir des connaissances de l'équipe, sans capitalisation possible.

Enfin, l'approche étant centrée sur un cycle global, des cycles régionaux, des types d'activités et des activités, il est exclusivement orienté sur les activités. Les types d'activités sont en effet des activités à différents niveaux de granularité ou d'abstraction, il ne s'agit ni de produits, ni de décisions, ni de buts. L'approche n'est pas polymorphique.

L'évaluation de la flexibilité de (Lucid/Start)* est donc :

Variabilité :	Nombreux variants informels
Granularabilité :	Plus de deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage

Distensibilité : Moyens informels
Réutilisation : Aucun élément de solution
Polymorphisme: Une seule vision sur le processus

La figure 18 représente cette classification.

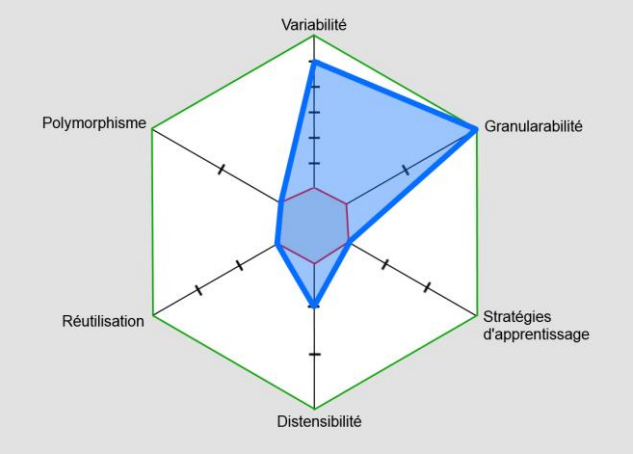

Figure 18 - Représentation graphique de la classification de (Lucid/Start)*

4.3 Les approches agiles

Les méthodes agiles reposent sur un ensemble de valeurs communes, formalisées dans le Manifeste Agile (Fowler & Highsmith 2001), comme l'adaptation au changement ou la place importante réservée à l'utilisateur final au cours du cycle de vie. Nous les avons classées en dehors des approches que nous nommons prescriptives car, selon (Gonzalez-Perez & Henderson-Sellers 2008), elles ont justement été imaginées pour diminuer « *l'excès de prescription dans les approches conventionnelles* ». Par ailleurs, à l'inverse des méthodes précédentes, (Sharp et al. 2006) considère que les méthodes agiles offrent un guidage « *léger* », c'est-à-dire qu'elles définissent les grandes étapes du modèle de processus sans entrer réellement dans le détail des activités. C'est cette même légèreté dans le guidage qui fait dire à C. Gonzalez-Perez et B. Henderson-Sellers que « *la liberté perçue dans les méthodes agiles* […] *tient au fait qu'elles proposent ce que les développeurs auraient fait de toute façon* », ce qui confirme la réputation de grande flexibilité de ces approches et les rend intéressantes dans notre étude.

Il existe de nombreuses méthodes dites agiles, comme Scrum (Schwaber 1995), Extreme Programming (Beck 1999a), Feature Driven Development (Palmer & Felsing 2002), Dynamic Systems Development Method (Stapleton 1999), Lean Software Development (Poppendieck & Poppendieck 2003). Il ne serait pas possible présenter ici toutes les évaluations. Nous avons choisi de présenter Extreme Programming (XP) (Beck 1999a), qui est mentionné sur le site de l'Alliance Agile comme l'exemple de méthode agile le plus connu[14].

[14] voir http://c2.com/cgi/wiki?AgileAlliance.

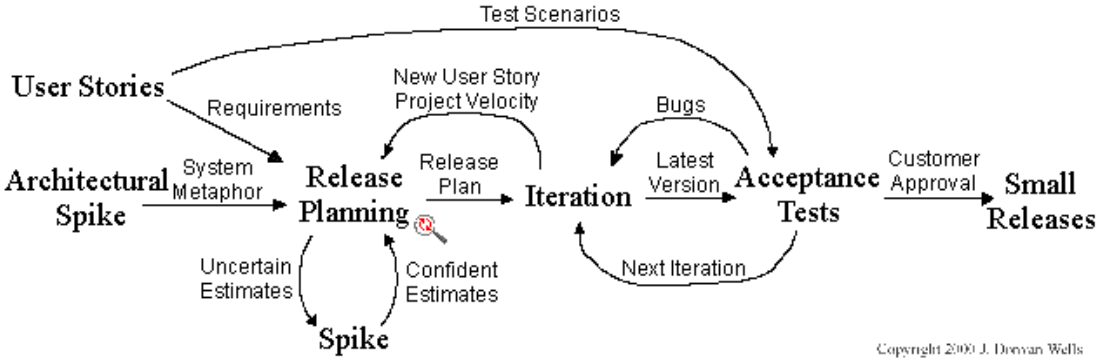

Figure 19 - Le cycle de vie de Extreme Programming (Wells 2012).

XP a été élaboré en 1999 et se focalise sur les besoins des utilisateurs finaux en proposant un cycle de développement itératif court, qui peut être adapté aux changements survenant dans les besoins. Les valeurs revendiquées par ce modèle de processus sont la simplicité, la communication, les retours d'information, le respect et le courage. Le cycle de ce modèle de processus repose sur six grandes étapes, comme le montre la figure 19 : (1) collecter les scénarios utilisateurs, (2) étudier les solutions pour les problèmes de conception ou les difficultés techniques les plus aigus, (3) préparer un agenda de développement, (4) développer itérativement les fonctionnalités au cours de cycles brefs, (5) soumettre les développements à des tests d'acceptabilité, pour partie automatisés (tests de non-régression, par exemple) et pour partie réalisés par les utilisateurs finaux, et avec l'accord de ces derniers, (6) livrer des versions comportant de faibles incréments.

La présentation du cycle de vie montre qu'il ne formule que quelques rares variants, par exemple lorsque l'évaluation d'un scénario utilisateur est incertaine (ce qui n'est pas une condition très précise), le modèle de processus propose d'étudier les solutions en réalisant des développements dédiés au problème. Ce modèle de processus est donc considéré comme offrant quelques variants informels.

En termes de niveaux de détails, XP est décrit par K. Beck dans une publication (Beck 1999a) et dans plusieurs ouvrages comme (Beck 1999b), (Beck & Fowler 2000), (Beck 2002). Le livre initial comporte un premier chapitre présentant une vue générale de la méthode, suivi d'autres chapitres qui détaillent chacune des prescriptions de l'approche. Le second ouvrage repositionne l'approche dans les difficultés générales du développement de logiciels et apporte des compléments sur des points considérés comme primordiaux, comme la prise en compte des coûts, de la qualité attendue, du temps disponible et de la portée du logiciel dans la planification, ou sur la façon de hiérarchiser les scénarios utilisateurs et les fonctionnalités à développer. Son objectif est clairement de compléter le premier ouvrage en détaillant certains éléments, ce qui constitue une troisième granularité. XP est donc considéré ici comme offrant plus de deux granularités.

Les différents ouvrages donnent des niveaux de détails différents, mais restent dans le même type de discours : K. Beck présente des arguments, qu'il explique clairement, montrant comment il pense que son approche permettra de dépasser les limites qu'il a

identifiées. Il ne propose cependant pas d'exemples de mise de œuvre ni d'autres façons d'aborder le modèle de processus. Ce dernier offre donc une seule stratégie d'apprentissage.

K. Beck explicite aussi qu'il convient d'ajouter des activités ou des outils lorsqu'il y en a besoin, comme, par exemple, de définir des métriques pour le suivi de projet (Beck 1999b). Cependant, l'auteur ne définit ni où ces ajouts sont possibles, ni comment il convient de procéder pour les réaliser concrètement. Le modèle de processus est donc évalué comme offrant des moyens informels d'extension et de réduction.

Il n'y a pas, dans le modèle de processus de XP, de mention d'élément de solution réutilisable par l'équipe de développement. Par ailleurs, ce modèle est centré sur les activités. Les artefacts et les autres focales comme les décisions sont mentionnés mais restent très secondaires. Il n'est pas possible, à travers les ouvrages mentionnés, d'aborder le modèle de processus autrement que par les activités. XP ne peut donc être considéré comme polymorphique.

L'évaluation de la flexibilité de Extreme Programming est donc :

Variabilité :	Quelques variants bien définis
Granularabilité :	Plus de deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Des moyens informels
Réutilisation :	Aucun élément de solution
Polymorphisme:	Une seule vision sur le processus

La figure 20 montre la représentation graphique de cette classification.

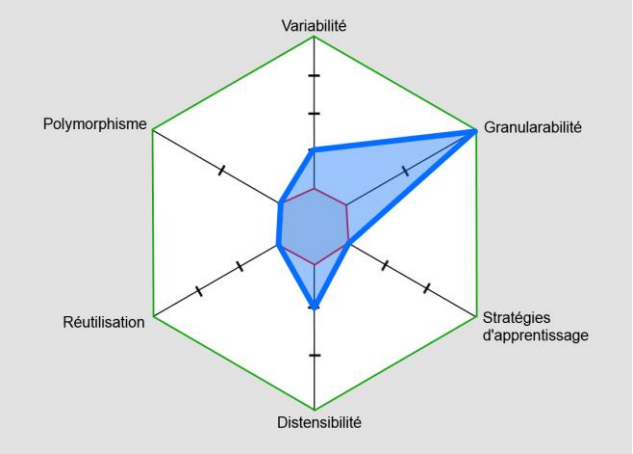

Figure 20 - Représentation graphique de la classification de Extreme Programming.

4.4 Les approches constructivistes

Nous plaçons dans la catégorie des approches constructivistes toutes les (méta-)méthodes qui proposent la construction d'un modèle de processus. Il s'agit de métamodèles de processus comme SPEM (Object Management Group 2008), d'approches

situationnelles comme (Hug 2009). Le modèle de processus n'étant pas directement défini ici, l'évaluation concerne donc le potentiel de flexibilité apporté par le métamodèle ou la méta-méthode. L'évaluation est donc sensiblement différente de l'évaluation d'un modèle de processus. Il s'agit de prendre en compte les structures offertes par le métamodèle qui permettront ou non de construire ensuite des modèles plus ou moins flexibles. Par exemple, une approche constructiviste peut définir de nombreuses formes d'aide pour la création d'un modèle de processus. Mais ces aides ne concernent pas le travail de développement de logiciel : ce ne sont pas des stratégies d'apprentissage destinées aux concepteurs et aux développeurs. A l'inverse, une approche constructiviste pourrait spécifier comment inclure dans le modèle de processus des stratégies d'apprentissage destinées aux concepteurs et aux développeurs. Dans le premier cas, nous évaluerions l'approche comme n'offrant pas de stratégies d'apprentissage (pour les concepteurs et les développeurs), alors que dans le second cas, nous considérerions que l'approche prend bien cette forme de flexibilité en compte.

4.4.1 SPEM

SPEM (pour Software and Systems Process Engineering Meta-Model) est un métamodèle de processus standardisé et proposé par l'Object Management Group (Object Management Group 2008). SPEM comporte sept paquetages (voir figure 21), qui définissent les structures élémentaires utilisées dans la définition de modèles de processus, eux-mêmes intégrés dans la définition de méthodes.

Une activité (modélisée comme un *BreakdownElement*) comporte un attribut *isOptional* qui définit si cette activité est optionnelle. SPEM permet donc de définir différents chemins : l'un passant par une activité optionnelle et l'autre non. Cependant, cette forme de variabilité est limitée, elle ne permet pas d'exprimer toutes les possibilités. Il n'est par exemple pas possible de définir des alternatives, comme deux façons de réaliser une même tâche. La variabilité reste donc limitée, ce qui nous amène à classer SPEM comme offrant quelques variants informels, en considérant que le seul attribut *isOptional* ne suffit pas à spécifier une condition bien définie.

En termes de granularités, un *BreakdownElement* (sur lequel repose la définition des activités et des artefacts) est une composition d'autres *BreakdownElements*, ce qui permet de définir autant de niveaux de détails que souhaité dans ces deux éléments. De plus, le paquetage *MethodPlugin* permet de composer des structures de processus venus du paquetage *ProcessStructure*, assurant une granularité à un niveau plus élevé que celui des activités. Grâce à ces deux structures, le nombre de niveaux de détails peut être modulé à volonté.

SPEM permet de définir des patrons de processus (nommés *MethodContents*), qui peuvent être réutilisés. La réutilisation ne porte cependant pas sur les éléments de solution, il n'y a pas de structure permettant de modéliser un artefact prédéfini ou un patron de conception.

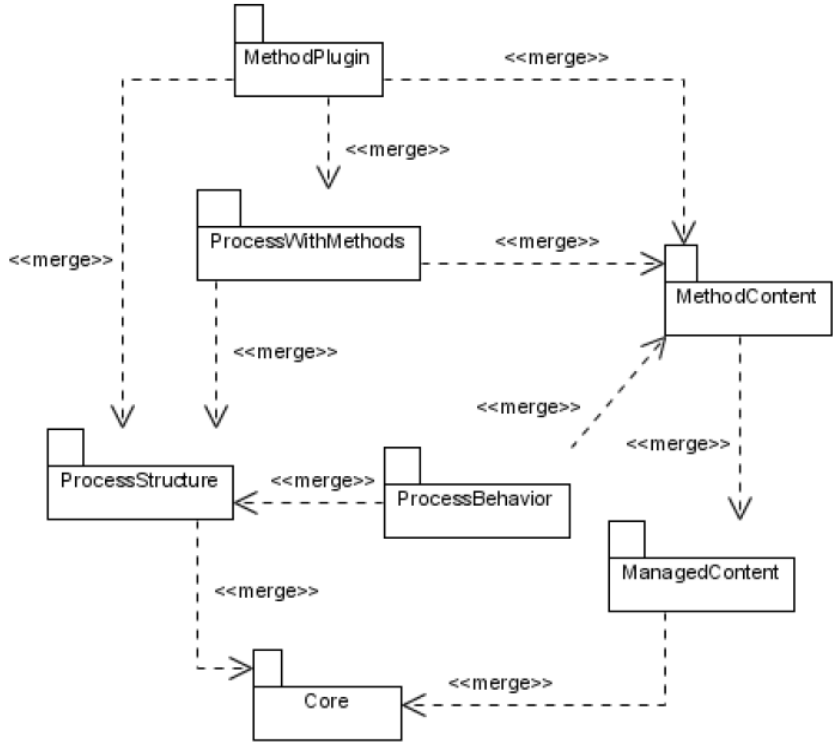

Figure 21 - Structure générale du métamodèle de SPEM 2.0

La description de SPEM est un document de 236 pages avec une définition complète de chacun des éléments. Cette documentation n'est cependant pas transmissible au modèle de processus : elle est destinée à comprendre comment construire un modèle de processus et non à apprendre comment appliquer ce dernier. SPEM n'offre donc pas de possibilité d'intégrer de stratégie d'apprentissage dans le modèle de processus, en dehors de la structure de ce dernier.

Dans le paradigme de SPEM, un modèle de processus est conçu de manière flexible, soit en créant les structures nécessaires, soit en réutilisant des structures déjà existantes. Puis il configuré selon les contraintes du projet et enfin mis en œuvre. Il n'existe pas de possibilité de modifier le modèle de processus une fois qu'il est mis en œuvre : il faut revenir à la phase de conception ou à la phase de configuration. SPEM n'offre donc pas de possibilité d'étendre ou de contracter dynamiquement le modèle de processus qui l'instancie, il n'est pas distensible.

SPEM étant défini comme un profil UML, il permet la conception des modèles de processus centrés sur les artefacts (Hug 2009). Cependant, pour que ce métamodèle soit considéré comme polymorphique, il faudrait que SPEM propose une structure permettant que les modèles de processus qui l'instancient puissent eux-mêmes être polymorphiques. Or, il n'y a rien dans le métamodèle qui recommande ou permette de basculer d'une présentation centrée sur les activités à une présentation centrée sur les produits, dynamiquement et selon le choix de l'utilisateur. Ce métamodèle est donc considéré comme offrant une vision unique du processus.

L'évaluation de la flexibilité de SPEM, représentée graphiquement sur la figure 22, est donc :

Variabilité :	Quelques variants informels
Granularabilité :	Autant de granularités que souhaité
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Pas de procédure
Réutilisation :	Pas d'éléments de solution
Polymorphisme :	Une présentation du processus

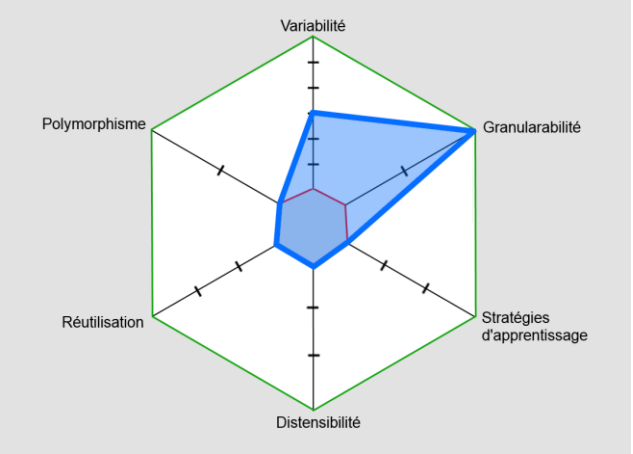

Figure 22 - Représentation graphique de la classification de SPEM

4.4.2 Le Work Product Pool

Le Work Product Pool (Gonzalez-Perez & Henderson-Sellers 2008) nous semble intéressant à présenter ici à deux titres. D'une part, il s'agit d'une approche centrée sur les produits, ce qui signifie que l'élément central du modèle de processus est l'ensemble des artefacts qui conduisent à la construction système, contrairement aux approches que nous avons vues ci-dessus et qui sont centrées sur les activités. D'autre part, les auteurs estiment, comme nous l'avons vu dans l'étude des besoins des concepteurs et des développeurs, que les modèles de processus centrés sur les activités sont trop prescriptifs et rebutent leurs utilisateurs, qui se trouvent limités dans leurs capacités de décision et dans leurs choix.

L'approche proposée consiste à partir du produit final que l'on souhaite obtenir, et à analyser les artefacts requis pour le construire « *simplement et directement* », puis à répéter cette analyse sur ces artefacts, jusqu'à l'obtention d'artefacts élémentaires, fournis par le client ou disponibles en interne. L'équipe de réalisation construit ainsi son propre modèle de processus, adapté par construction à son projet et à ses compétences. Pour permettre une certaine variabilité dans les artefacts, les auteurs réutilisent la notion de *Work Product Kind,* qui permet de modéliser un type d'artefact plutôt qu'un artefact concret. Ce concept est originellement défini dans (International Organization for Standardization 2007).

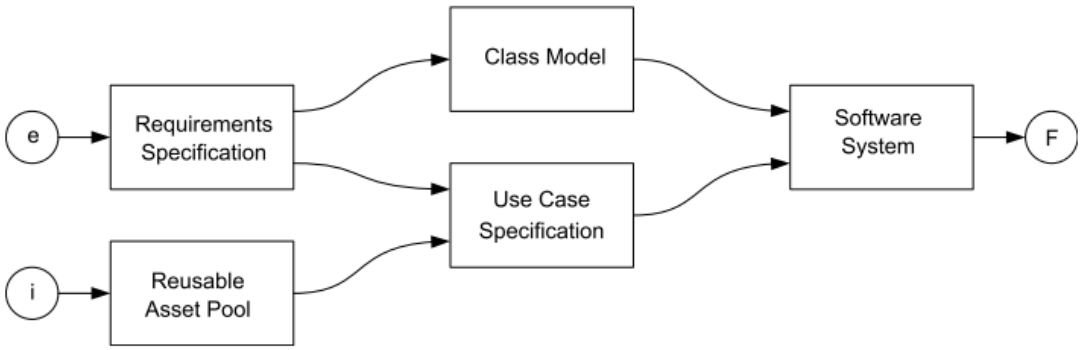

Figure 23- Un exemple simple de modèle de processus créé avec le Work Product Pool (Gonzalez-Perez & Henderson-Sellers 2008). La lettre (e) signifie que "Requirement Specifications" est fourni par l'extérieur (le client), la lettre (i) indique que "Reusable Asset Pool" est disponible en interne. (F) indique que "Software System" est le produit final.

La figure 23 présente un exemple de modèle de processus simple construit à l'aide de cette proposition : le système final (« *Software System* ») requiert un diagramme de classes (« *Class Model* ») et des spécifications (« *Use Case Specification* »). La construction du diagramme de classes repose sur les besoins (« *Requirement Specifications* »), fournis par le client. Les spécifications reposent elles aussi sur ces besoins, ainsi que sur un ensemble de cas d'utilisation récurrents et réutilisables (« *Reusable Asset Pool* »), qui est préexistant en interne.

Le modèle de processus final consiste donc en réseau d'artefacts construits les uns à partir des autres. Les auteurs explicitent que le « processus », c'est-à-dire les activités, sera défini concrètement au fur et à mesure de la construction des artefacts. Pour permettre ce choix des activités, le Work Product Pool repose sur l' « *assistance d'outils logiciels* » permettant d'identifier les paires <processus, artefact>, c'est-à-dire les activités. Les auteurs indiquent que, comme « *le processus n'est pas inclus dans la structure de la méthodologie, il peut être librement changé tant que le réseau d'artefacts reste identique* ». De ce fait, si les variants potentiels sont nombreux, aussi bien dans les artefacts que dans les activités, ils restent informels, puisqu'ils ne sont pas définis dans le modèle de processus et devront être identifiés lors de sa mise en œuvre par l'équipe de développement.

L'étude des niveaux de détails offerts par cette approche repose sur la structure des artefacts, puisqu'ils sont l'élément central du Work Product Pool. Les auteurs explicitent que la notion de *WorkProduct* dans leur approche est similaire à celle définie dans la norme ISO/IEC 24744 (International Organization for Standardization 2007). La figure 24 présente la modélisation des artefacts dans cette norme : la classe *WorkProduct* est spécialisée en *CompositeWorkProduct*, qui est une agrégation de *WorkProducts*. Cette structure permet de définir autant de niveaux de détails que souhaité.

L'apprentissage du modèle de processus est difficile à anticiper : celui-ci étant bâti dynamiquement, il n'est pas possible de prévoir les besoins d'apprentissage, sauf à imaginer que les outils logiciels permettant la définition des activités comportent eux-

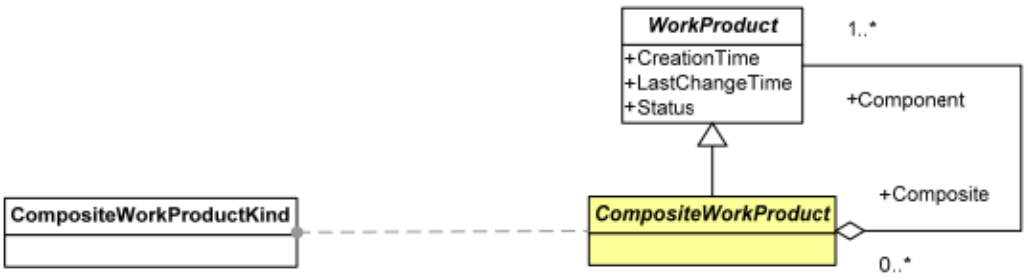

Figure 24 - Structure des Artefacts dans (International Organization for Standardization 2007)

mêmes différentes stratégies d'apprentissage, ce qui n'est pas mentionné par les auteurs. A défaut de précision sur ce sujet, le Work Product Pool est classé comme offrant une seule stratégie d'apprentissage : sa propre structure. La méta-méthode propose de modifier dynamiquement le modèle de processus en fonction de la situation propre au projet et permet, comme le disent les auteurs, d'ajouter, de modifier ou de supprimer des artefacts intermédiaires en fonction des changements dans les besoins de l'équipe. L'approche offre donc la distensibilité. Cependant, si la procédure pour définir et insérer les artefacts dans le réseau de produits est clairement définie, les procédures de vérification de la cohérence du réseau ne le sont pas.

Les auteurs expliquent aussi qu'un artefact intermédiaire (utilisé dans la construction d'un artefact plus avancé) peut être fourni par l'« *extérieur* » (le client, par exemple), construit à partir d'autres artefacts intermédiaires ou trouvé dans un « *vivier de composants réutilisables* » (*reusable asset pool*). Nous avons vu que la figure 23 montre l'exemple de réutilisation de cas d'utilisation récurrents. L'approche offre donc des possibilités de réutilisation. Cependant, comme le modèle de processus n'est pas prédéfini, les propositions d'éléments de solution ne sont pas spécifiées : tout dépend du vivier de composants disponibles localement et de la façon dont l'équipe de développement va mener la décomposition du produit en final en produits intermédiaires. L'approche permet donc la réutilisation mais ne propose aucun élément de solution.

Enfin, cette approche étant centrée exclusivement sur les artefacts et les auteurs recommandant explicitement de mettre les activités au second plan et de ne pas présenter le modèle de processus selon une autre perspective, le Work Product Pool offre une seule perspective.

La classification de ce modèle est donc la suivante :

Variabilité :	Nombreux variants informels
Granularabilité :	Autant de granularités que souhaité
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Procédures bien définies, sans validation
Réutilisation :	Pas d'éléments de solution
Polymorphisme :	Une présentation du processus

La représentation graphique de cette classification est présentée sur la figure 25.

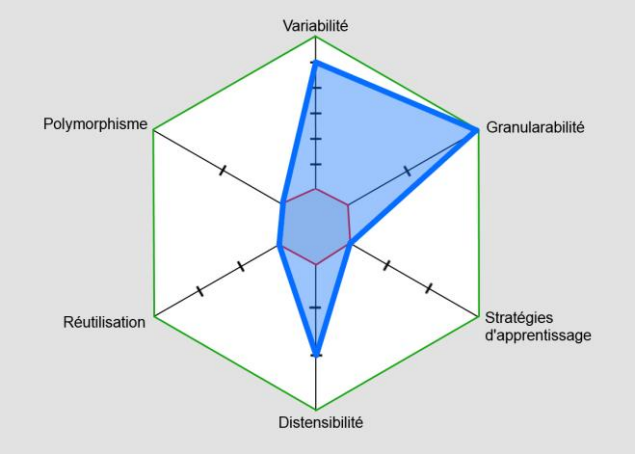

Figure 25 - Représentation graphique de la classification du Work Product Pool.

4.4.3 L'approche situationnelle de C. Hug

Les approches situationnelles ne sont pas des modèles de processus : il s'agit plutôt de méta-méthodes permettant de construire un modèle de processus, de façon adaptée à la situation, par assemblage d'éléments méthodologiques. Ce principe de construction ad-hoc de méthode permet aux équipes de développement de « *suivre des prescriptions ajustées à leurs besoins et de s'en emparer avec un certain sentiment de possession* » (Henderson-Sellers & Ralyté 2010). Plusieurs visions de ce qu'est un élément de méthode existent : le **fragment**, qui est défini comme un élément atomique cohérent de méthode (Harmsen et al. 1994), la **partie** (ou, plus classiquement, *chunk* en anglais), qui combine un fragment orienté sur le processus avec un fragment orienté sur les produits (Henderson-Sellers et al. 2008), le **composant**, qui est « *un élément autoporteur d'une méthode de génie logiciel exprimant le processus pour transformer un ou plusieurs artefacts en un artefact cible et la motivation de cette transformation* » (Wistrand & Karlsson 2004), et le **service « méthode »**, qui est un « *service métier proposé aux concepteurs/développeurs en mettant en œuvre un certain processus de développement* » (Guzelian 2007), et qui peut être assemblé avec d'autres services pour définir des processus complexes. Le terme de fragment étant le plus générique, c'est celui que nous retenons ci-dessous pour englober tous les autres. Plusieurs techniques d'assemblage existent, reposant sur la modélisation des fragments, en exprimant différentes caractéristiques, comme leurs buts, leurs interfaces, les caractéristiques des projets adéquats, leurs finalités, leurs arguments (Deneckère et al. 2009).

L'approche de C. Hug nous intéresse particulièrement ici parce qu'il s'agit d'une approche multi-points de vue, c'est-à-dire permettant de composer des fragments issus de modèles de processus avec différentes focales (Hug 2009), ce qui porte le potentiel d'offrir du polymorphisme dans le processus issu de la construction. Pour permettre une telle composition, l'approche repose sur un alignement des concepts issus de métamodèles de processus orientés sur différentes focales. Un métamodèle de domaine résultant permet de représenter les fragments, quelle que soit leur orientation originelle. Ce métamodèle de

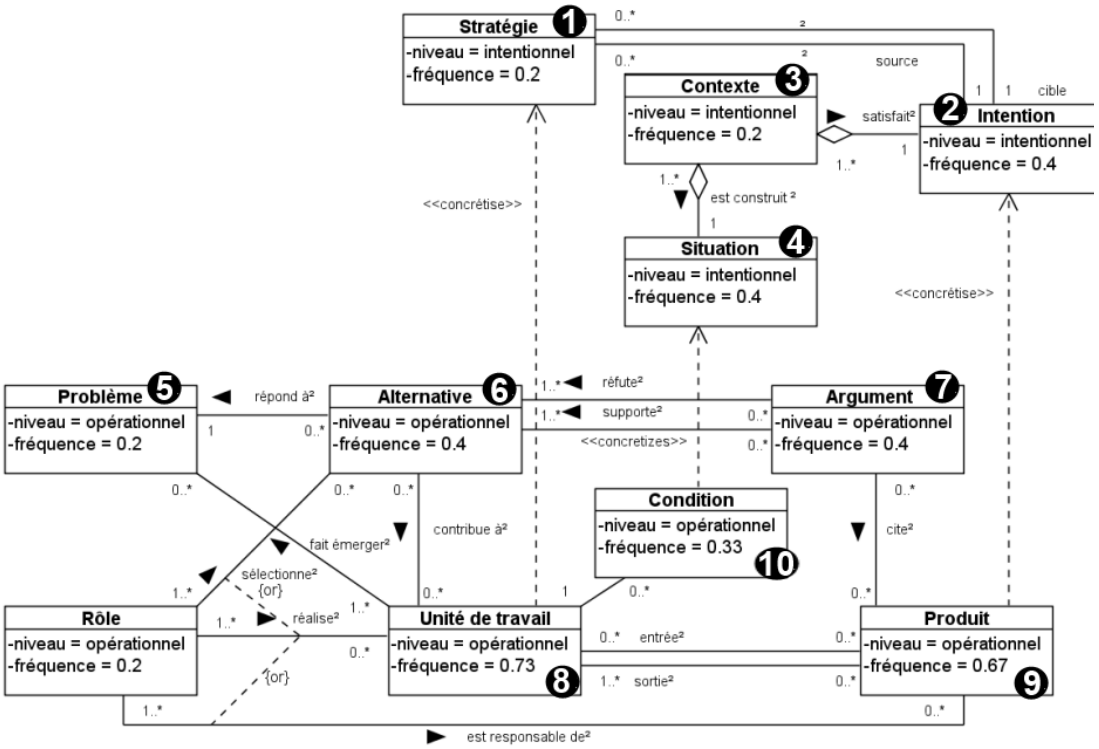

Figure 26 - Métamodèle de l'approche de C. Hug.

domaine offre deux niveaux d'abstraction (voir figure 26) : un niveau intentionnel qui représente les buts des fragments et un niveau opérationnel qui incarne ces buts. Au niveau intentionnel, le métamodèle contient des *stratégies* ❶, des *intentions* ❷, des *contextes* ❸ et des *situations* ❹ : ces concepts permettent de représenter les approches orientées sur le contexte ou les buts. Au niveau opérationnel, le métamodèle propose les concepts de *problème* ❺, *alternative* ❻ et *arguments* ❼, qui permettent de modéliser les processus orientés sur les décisions. Ce niveau comporte aussi les concepts *d'unité de travail* ❽, utilisé pour la représentation des activités, et de *produit* ❾, utilisé pour représenter les artefacts. L'approche définit, à destination d'un *ingénieur des méthodes*, une procédure de création de métamodèle de processus par sélection itérative et incrémentale de fragments. Ce métamodèle de processus est ensuite instancié en modèle de processus, puis confié à un *chef de projet* qui le met en œuvre.

En termes de variabilité, la classe *Alternative* ❻ joue un rôle clé : elle comporte un attribut en instanciation profonde nommé *sélectionnée*, dont la valeur est décidée lors de l'exécution du processus construit à partir du métamodèle de domaine. Les choix sont donc faits durant l'exécution du processus, c'est-à-dire par l'équipe de développement. Cette approche permet donc d'inclure autant de variants que souhaité. De plus, une alternative comporte un attribut *description*, elle répond à un *problème* ❺, et elle est associée à autant d'*arguments* ❼ que nécessaire, pour exprimer les éléments qui militent en faveur de son choix, aussi bien que ceux qui réfutent ce choix. Une *unité de travail* peut de plus être soumise à des *conditions* ❿, ce qui implique aussi des choix (contraints) à ce niveau. Cette approche est donc évaluée comme offrant de nombreux variants bien définis.

Le métamodèle de domaine définit deux granularités : le niveau intentionnel et le niveau opérationnel. Ainsi, les *stratégies* sont concrétisées en *unités de travail*. Ces dernières ne peuvent pas être raffinées : elles ne sont pas définies avec une structuration arborescente. Cependant, l'approche définit un patron permettant d'ajouter une composition réflexive ou une agrégation à une classe. Il est dès lors possible d'ajouter une personnalisation qui ajoute une structure composite aux unités de travail. En prenant en compte ce patron, l'approche propose donc autant de granularités que souhaité.

Si l'approche permet ainsi de raffiner une unité de travail en unités de travail plus simples, elle ne comporte pas de structure dans le métamodèle ni de patron permettant d'exprimer un élément avec différents vocabulaires, ni d'ajouter d'autres types d'éléments favorisant l'apprentissage du modèle de processus résultant par l'équipe de développement. L'approche ne permet de construire que des modèles de processus avec une seule stratégie d'apprentissage, le modèle de processus lui-même.

Par ailleurs, les différents patrons permettent d'ajouter librement des éléments dans le modèle de processus. Le modèle de processus peut donc être étendu pendant l'instanciation du métamodèle de domaine, c'est-à-dire lors de la conception du modèle de processus. Cette opération n'est cependant pas possible pendant l'exécution du modèle de processus résultant : les extensions doivent être incorporées pendant la conception du modèle de processus. Les équipes de développement, qui interviennent pendant l'exécution du modèle, ne peuvent donc pas étendre dynamiquement ce dernier. L'approche ne propose pas de procédure d'extension ou de réduction utilisables par les concepteurs et les développeurs.

Il en va de même pour les éléments de solution : les fragments méthodologiques peuvent être réutilisés librement, mais uniquement lors de la conception du modèle de processus, ils ne sont pas réutilisables par l'équipe de développement lors de l'exécution du modèle de processus. Le métamodèle ne comporte par ailleurs pas de possibilité d'exprimer qu'un artefact est fourni ou prédéfini pour être directement réutilisé ou enrichi si besoin : il n'est pas possible, d'après la structure du métamodèle, de réutiliser des éléments de solutions obtenus lors d'un précédent projet. L'approche n'offre donc aucun élément de solution à l'équipe de développement.

L'approche de (Hug 2009) permet de modéliser et d'assembler des fragments avec différentes focales. Ces différentes focales sont incluses dans le métamodèle de processus conçu par assemblage des fragments. On les retrouve dans le modèle de processus, qui instancie le métamodèle sans perdre la notion de la focale d'origine. Il semble donc que cette approche ait le potentiel de présenter différentes focales lors de la mise en œuvre du modèle de processus. Ce point n'est cependant pas traité dans (Hug 2009) : s'il est clairement mentionné que le modèle de processus est multi-points de vue (multi-focales, avec notre vocabulaire), rien n'est dit du processus lui-même. Au contraire, tous les exemples de modèle de processus montrent exclusivement le niveau opérationnel et sont centrés sur les activités. Rien n'indique donc que les concepteurs et développeurs puissent

disposer de plusieurs perspectives pendant l'exécution du modèle de processus. Cette approche n'est pas polymorphique.

En synthèse, l'évaluation de la flexibilité de (Hug 2009), représentée sur la figure 27, est donc :

Variabilité :	Nombreux variants bien définis
Granularabilité :	Plus de deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Aucune possibilité
Réutilisation :	Aucun élément de solution
Polymorphisme:	Une seule vision sur le processus

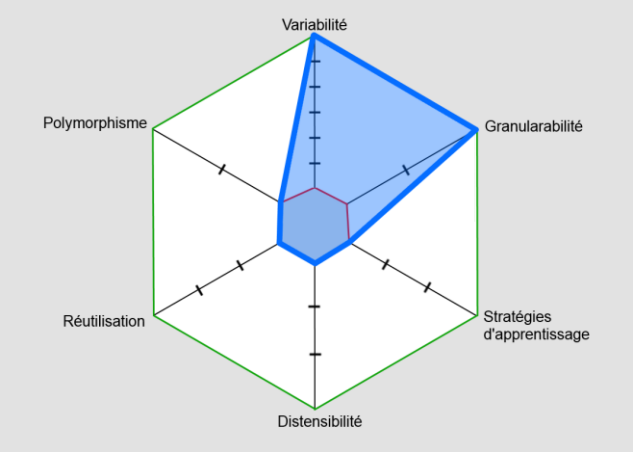

Figure 27 - Représentation graphique de la classification de l'approche de C. Hug

4.4.4 Le Component-oriented Process Modeling Framework

F. Golra propose une structure nommée Component-oriented Process Modeling Framework (CPMF), composée de plusieurs métamodèles complémentaires, correspondant aux phases du cycle de vie d'un modèle de processus (Golra 2014). D'après l'auteur, on trouve typiquement les phases de spécification, d'implantation et d'instanciation. L'approche propose donc trois métamodèles correspondant à ces trois phases. Le métamodèle de spécification (« *Process Specification Metamodel* ») « *définit les structures de base du modèle de processus* ». Il comporte par exemple les concepts de rôle, d'activité et de « contrat » (les dépendances) entre activités. Le métamodèle d'implantation (« *Process Implementation Metamodel* ») représente le modèle de processus à deux niveaux : abstrait et concret. Le niveau abstrait permet de définir des « types d'objets », par exemple des types d'activités, qui sont concrétisés au niveau en dessous, par exemple en activités. Cette structure est utilisée pour permettre la représentation de plusieurs solutions, par exemple plusieurs activités concrètes pour réaliser une même étape définie comme une activité abstraite. Enfin, le métamodèle d'instanciation (« *Process Instanciation Metamodel* ») représente la « *sémantique de l'exécution du modèle de processus* », c'est-à-dire les choix parmi les éléments du niveau concret du modèle d'implantation. Lors de la définition du modèle de processus, le

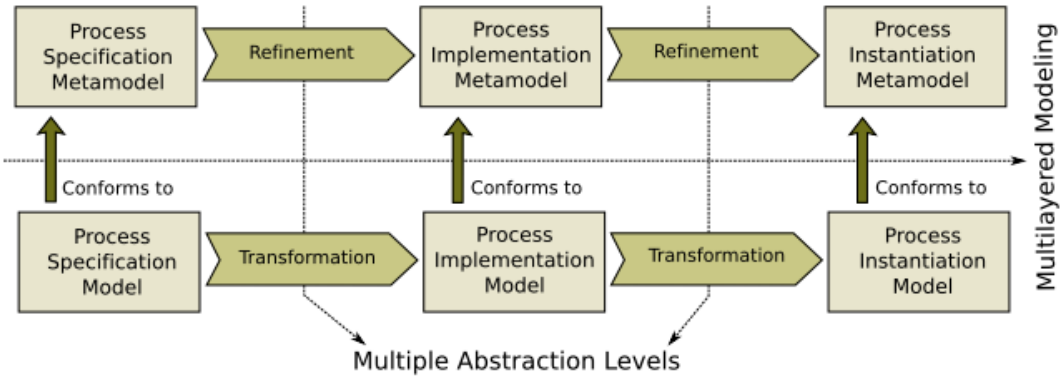

Figure 28 - Métamodèles de processus pour le "développement multi-métamodèles" (Golra 2014)

modèle de spécification est transformé en modèle d'implantation, qui est ensuite transformé en modèle d'instanciation, comme le montre la figure 28.

La double structure en niveaux abstrait et concret permet de définir des variants pour chaque activité du modèle de processus. Les activités abstraites sont soumises à un contrat, qui définit leurs besoins et leur production, ainsi à qu'à des pré- et post-conditions. Le contrat et les conditions sont définis au niveau abstrait, tandis que les variants sont définis au niveau concret et sont donc tous soumis aux mêmes contraintes, issues de l'activité abstraite qu'ils concrétisent. Rien ne permet donc d'exprimer les motivations pour choisir un variant plutôt qu'un autre : CPMF propose de nombreux variants informels.

Le métamodèle de spécification définit un processus comme une composition d'activités. Une activité peut aussi être composée d'un processus. Cette structure permet de définir autant de niveaux de détails que souhaité. On peut cependant noter qu'il n'existe pas de structuration hiérarchique des artefacts.

L'approche proposée ne signale aucune possibilité pour faciliter l'apprentissage du modèle de processus, en dehors des explications portées par les activités. Cette approche est donc évaluée comme offrant une stratégie d'apprentissage.

En termes de distensibilité, l'approche permet de modifier dynamiquement un processus, c'est-à-dire un modèle de processus pendant son exécution. Ces modifications consistent en l'ajout, la suppression ou la modification d'activités du niveau concret. Les conditions dans lesquelles ces modifications peuvent être effectuées sont bien définies. Nous évaluons donc cette approche comme offrant des procédures d'extension bien définies, dont les limites sont claires.

La réutilisation est un but explicite de l'approche. Celle-ci propose en effet deux dictionnaires, l'un pour artefacts et l'autre pour les composants des modèles de processus. La réutilisation est ici forte : CPMF propose de nombreux éléments de solution.

Un modèle de processus est une composition d'activités qui, à travers leur « *contrat* » utilisent des artefacts. Le modèle de processus lui-même n'est donc pas polymorphique, l'accès aux différents composants se faisant uniquement à travers les activités. L'outillage

associé à CPMF ne propose pas non plus de représentation centrée sur un autre élément, comme les artefacts. Cette approche propose donc une seule vision sur le processus.

L'évaluation de la flexibilité de CPMF, est donc :

Variabilité :	Nombreux variants informels
Granularabilité :	Plus de deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Procédures bien définies, avec limites
Réutilisation :	Nombreux éléments de solution
Polymorphisme:	Une seule vision sur le processus

La figure 29 représente graphiquement la classification de CMPF.

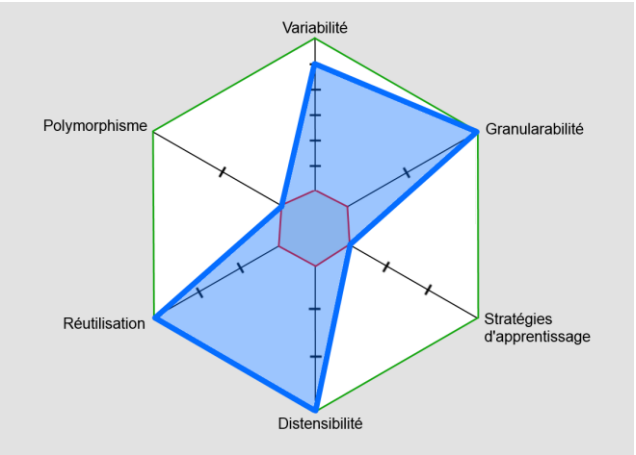

Figure 29 - Représentation graphique de la classification de CPMF.

4.4.5 UsiXML

UsiXML est à la fois un projet ITEA2 et un langage de description d'interfaces utilisateur supportant différentes modalités d'interaction et dynamiquement adaptables à leur contexte d'usage (Limbourg & Vanderdonckt 2004). Le langage UsiXML repose sur une approche dirigée par les modèles, dont la structure en plusieurs niveaux d'abstraction est qualifiée de « *extraordinairement flexible* » (Jerónimo et al. 2009). Nous ne pouvions manquer de regarder si la flexibilité du langage se retrouve dans le modèle de processus. Une seconde raison de présenter cette approche est que UsiXML comporte un métamodèle de processus et un modèle de processus dont les flexibilités sont différentes. Cet écart apporte un éclairage supplémentaire sur la différence entre le potentiel de flexibilité offert par un métamodèle de processus et la flexibilité de son instanciation comme modèle de processus.

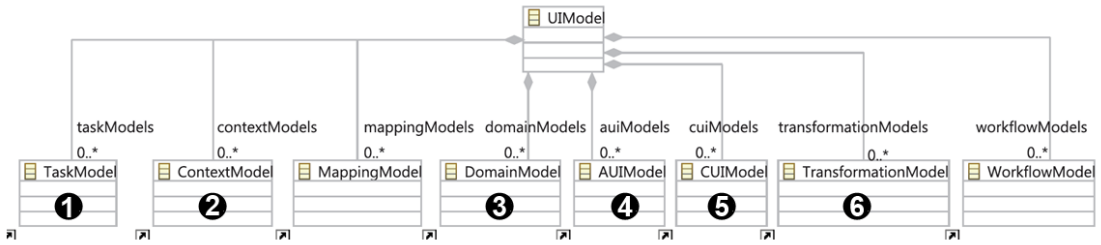

Figure 30 - Structure d'un modèle d'interaction dans l'approche UsiXML (UsiXML Consortium 2012)

L'approche UsiXML reprend, affine et étend les modèles définis dans le projet Caméléon (Calvary et al. 2001). La figure 30 présente la structure des modèles dans UsiXML. On trouve en particulier un modèle de tâches ❶, qui représente les tâches de l'utilisateur et qui est le niveau le plus abstrait de représentation de l'interaction ; un modèle de contexte ❷, qui représente l'utilisateur, la ou les plateformes et l'environnement ; un modèle de domaine ❸, qui représente les concepts manipulés dans l'interface ; un modèle d'interface abstraite (ou AUI, pour *Abstract User Interface*) ❹, qui représente les différents espaces de travail de l'interface et les possibilités de navigation entre eux ; un modèle d'interface concrète (ou CUI pour *Concrete User Interface*) ❺, qui représente les interacteurs mis en œuvre.

Ces modèles sont des instances de métamodèles définis dans l'approche et sont utilisés dans des transformations (voir figure 31), qui permettent de générer l'interface finale (ou FUI), c'est-à-dire le code exécuté par l'utilisateur. Les transformations peuvent permettre de réifier un modèle abstrait en un modèle plus concret : le modèle de tâches est considéré comme le plus abstrait et peut se concrétiser successivement en interface abstraite, interface concrète et interface finale. Il est aussi possible d'abstraire par transformation un code en modèle ou un modèle en un modèle plus abstrait, ce qui permet de faire de la rétro-ingénierie. Enfin, une opération de translation permet de

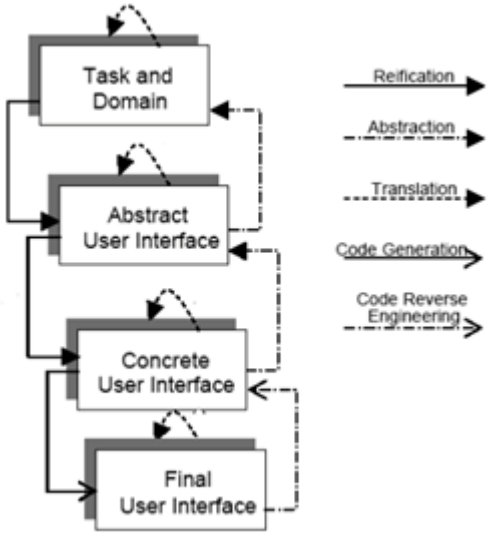

Figure 31 - Les différentes types de transformations dans l'approche UsiXML (UsiXML Consortium 2012)

transformer un modèle en un modèle de même niveau, par exemple pour prendre en compte une modification dans le contexte. Le modèle de transformations (figure 30, n°❻) représente non seulement les transformations mises en œuvre dans les processus de réification, abstraction et translation mais aussi la structuration temporelle de ces transformations, c'est-à-dire le ou les processus de génération de l'IHM finale.

La conception d'une interaction à l'aide de l'approche UsiXML nécessite la définition des modèles et des transformations. L'approche méthodologique de UsiXML (UsiXML Consortium 2011) est définie à deux niveaux : un métamodèle de processus nommé SPEM4UsiXML et une instanciation de ce métamodèle. Nous considérons ci-dessous les flexibilités de ces deux propositions.

4.4.5.1 Le métamodèle de processus SPM4UsiXML

Le métamodèle SPEM4UsiXML (UsiXML Consortium 2011) reprend les sept paquetages de SPEM 2.0 (Object Management Group 2008). La flexibilité offerte par le métamodèle de processus de UsiXML repose donc sur celle offerte par SPEM 2.0, que nous avons analysée dans la section 4.4.1. Cependant, dans l'approche UsiXML, deux paquetages ont été étendus par rapport à SPEM :

- *MethodContent*, qui décrit des rôles qui exécutent des tâches pour réaliser des produits, sans préciser l'enchainement des tâches et leur place précise dans un processus. Ce paquetage a été étendu dans UsiXML pour permettre la modélisation des différents types de transformation (réification, abstraction, translation,…), à l'aide de définitions d'étapes et de sous-étapes de développement (« *DevelopmentStepDefinition* » et « *DevelopmentSub-StepDefinition* ») ;
- *ProcessStructure*, qui définit la structure des processus, c'est-à-dire l'ordonnancement des activités et la composition des structures des processus, et qui a été étendu pour permettre de spécifier le flux des étapes et sous-étapes de développement.

Nous avons vu en section 4.4.1 que SPEM définit une structure permettant de décrire une activité ou un artefact comme optionnel, ce qui nous avait conduits à classer l'approche comme offrant quelques variants informels, cette « optionalité » ne permettant pas d'exprimer tous les variants, comme différentes formes de réalisation de la même tâche. L'extension du métamodèle pour UsiXML ne modifie pas cette analyse. En effet, si les extensions apportées dans le paquetage *ProcessStructure*, permettent d'exprimer que les étapes de développement (« *DevelopmentStepUse* ») peuvent utiliser différents modèles de transformation (une réification, une abstraction, une translation, une génération de code ou une rétro-ingénierie de code), cela ne constitue pas une différence suffisamment importante pour qu'on puisse considérer que la notion de variants a été généralisée dans le métamodèle UsiXML.

La structuration des éléments étant elle aussi héritée de SPEM, elle propose, comme dans ce métamodèle, autant de niveaux de détails que souhaité.

Le Consortium UsiXML a présenté ses travaux et a participé à de nombreux ateliers dans différentes conférences[15]. Ces éléments réunis permettent l'observation d'expériences. Cependant, la structure du modèle de processus, basée sur celle de SPEM (Object Management Group 2008), ne permet pas d'inclure la documentation dans le modèle de processus : ces éléments ne peuvent être considérés comme une stratégie d'apprentissage alternative pour la mise en œuvre de l'approche.

Dans les autres dimensions de la flexibilité, les modifications apportées au métamodèle de processus de SPEM ne permettent pas d'amender la classification : aucune procédure n'est définie pour permettre l'extension ou la contraction du modèle de processus pendant son exécution, la mise à disposition d'éléments de solution, comme des modèles embryonnaires à enrichir, n'est pas prévue, et le métamodèle est fortement orienté sur les activités et ne permet pas d'offrir une autre représentation du processus.

La classification de SPEM4UsiXML, qui est représentée sur la figure 33 avec celle du modèle de processus de UsiXML, est donc la suivante :

Variabilité :	Quelques variants informels
Granularabilité :	Plus de deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Aucune procédure
Réutilisation :	Pas d'éléments de solution
Polymorphisme :	Une seule vision sur le processus

4.4.5.2 Le modèle de processus

Le projet UsiXML propose un modèle de processus, représenté sur la figure 32 (UsiXML Consortium 2011). Ce modèle de processus ne présente qu'une seule des possibilités offertes par UsiXML : la conception d'interface par réification. Le processus est divisé en trois grandes étapes. Dans la première ❶, le concepteur réifie un modèle de domaine et un modèle de tâches (dont l'origine n'est pas précisée) en un modèle d'interface abstraite (AUI), en réalisant trois activités successives : l'identification des structures de l'IHM abstraite, la sélection d'un composant abstrait et l'identification de l'arrangement du composant abstrait. Dans la seconde étape ❷, le concepteur réifie ce modèle d'AUI en un modèle d'interface concrète (CUI), cette fois grâce à quatre activités successives (réification des conteneurs abstraits en conteneurs concrets, sélection des composants concrets, arrangement de chaque composant concret, définition de la navigation). Enfin, la troisième étape ❸, comportant une seule activité, permet la génération du code. Le document présentant le modèle de processus référence une publication (Limbourg & Vanderdonckt 2009) et indique que la démarché détaillée peut être trouvée dans cette publication. Ces informations doivent donc être prises en compte dans l'évaluation du modèle de processus.

[15] Voir http://www.usixml.eu/effective-ie-done/scientific-publications et http://www.usixml.eu/effective-ie-done/workshops

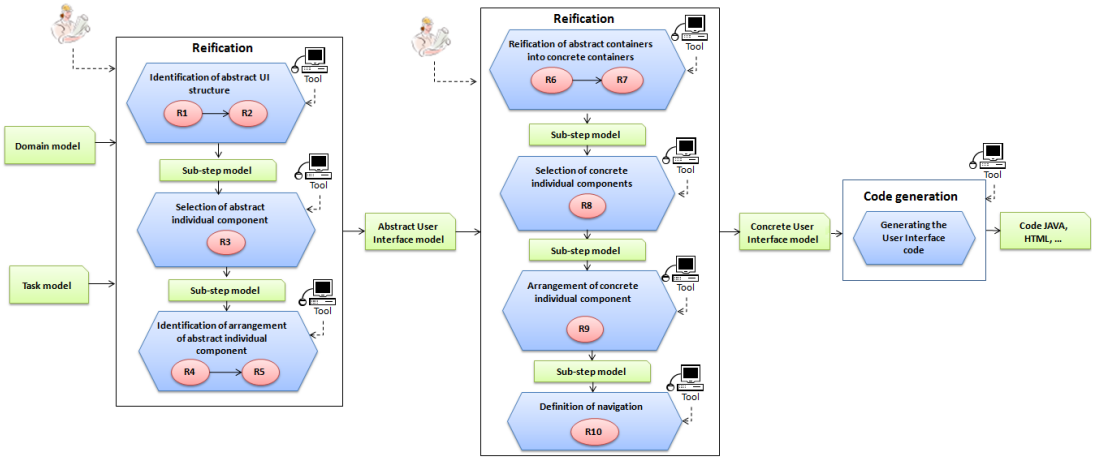

Figure 32 - Le modèle de processus de UsiXML (UsiXML Consortium 2011)

On peut noter que le modèle de processus tel qu'il est représenté sur la figure est incomplet : il ne représente pas toutes les stratégies offertes par le projet et le langage UsiXML (réification, abstraction, translation, rétro-ingénierie de code), et ne propose de aucun variant : l'enchaînement des activités est linéaire. La publication de Q. Limbourg et J. Vanderdonckt apporte cependant des variants : plusieurs chemins de développement sont définis :

- les chemins correspondant aux trois approches par réification, abstraction et translation ;
- le « reciblage » (« *retargeting* »), qui mixte les trois précédents : un code est abstrait en IHM concrète ou abstraite, qui est ensuite remaniée et finalement réifié de nouveau en code, afin de permettre une adaptation vers une nouvelle plateforme, par exemple ;
- le développement à partir d'un point intermédiaire (« *middle-out development* »), où, au lieu de partir du modèle de tâches, le développeur commence directement au niveau de l'IHM concrète ou de l'IHM abstraite ;
- le développement libre (« *Widespread development* »), dans lequel « *la conceptrice commence où elle veut* » et exécute les règles « *pertinentes à ce niveau, évalue les résultats* […] *et réalise les autres étapes de développement appropriées* ».
- le développement par aller-retour (« *Round-trip engineering* ») : après une génération de code, celui-ci est modifié manuellement, ces modifications étant perdues à défaut d'être réintégrées dans les niveaux d'abstraction supérieurs.

Ces quatre chemins sont cependant peu détaillés, la publication les évoque sans les définir avec le même niveau de détail que les trois premiers. En incluant cette publication, le modèle de processus de UsiXML offre donc trois variants bien définis, quatre variants informels, mais aucun variant dans les activités, ce qui nous amène à évaluer cette approche comme offrant de rares variants bien définis.

Le modèle de processus comporte deux niveaux de granularité : les étapes générales de réification et de génération de code, et un niveau de détail, qui décrit des activités. La publication de Q. Limbourg et J. Vanderdonckt précise ces activités sans les raffiner en sous-activités. Le modèle de processus n'apporte pas de complément facilitant son apprentissage, les stratégies d'apprentissage sont évaluées de la même façon que pour SPEM4UsiXML. Il en va de même pour la distensibilité, la réutilisation et le polymorphisme.

L'évaluation de la flexibilité du modèle de processus de UsiXML sous sa forme actuelle est donc :

Variabilité :	Rares variants bien définis
Granularabilité :	Deux granularités
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Aucune possibilité
Réutilisation :	Pas d'éléments de solution
Polymorphisme :	Une seule vision sur le processus

La figure 33 présente les classifications des flexibilités du métamodèle ❶ et du modèle ❷ de processus proposés par le Consortium UsiXML. Elle permet d'éclairer l'idée qu'un métamodèle apporte un potentiel de flexibilité et que le modèle réalise plus ou moins ce potentiel. On visualise en effet clairement que la flexibilité du modèle de processus est inférieure à celle du métamodèle qu'il instancie.

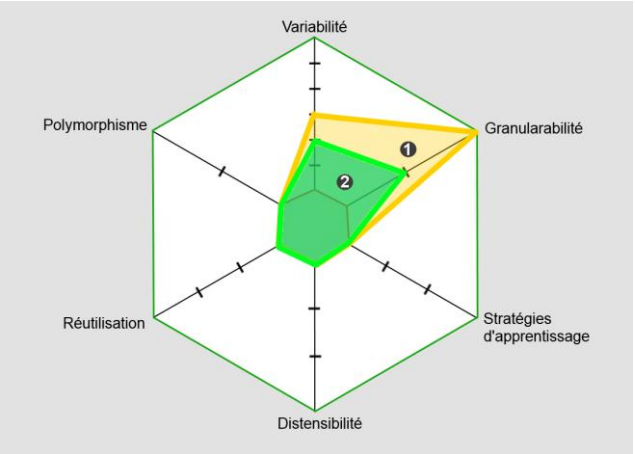

Figure 33 - Flexibilité de SPEM4UsiXML (en pointillés orange) et de UsiXML (en trait continu bleu)

4.5 Conclusion

Ce chapitre a été consacré à l'application de Promote à douze modèles et métamodèles de processus, représentant les approches prescriptives, agiles et constructivistes, ainsi que des approches dédiées à la conception d'Interfaces Homme-Machine.

La figure 34 présente la superposition des évaluations des différentes approches que nous avons présentées ci-dessus. Comme nous l'avons dit plus haut, les graduations des critères composant la flexibilité étant ordonnées, plus le trait représentant l'évaluation

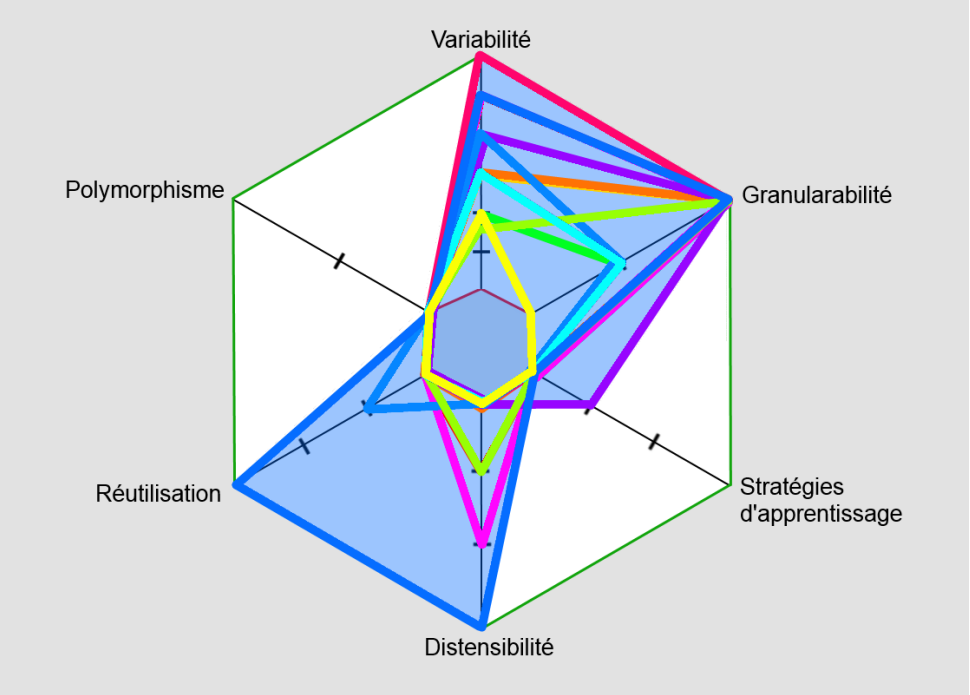

Figure 34 - Représentation cumulative des évaluations de flexibilité

d'un (méta)modèle de processus s'approche de la bordure extérieure du graphique et plus le (méta)modèle de processus est considéré comme offrant le critère considéré. Malgré la quasi-impossibilité de distinguer les évaluations des différents modèles, ce graphique nous semble permettre de visualiser rapidement deux éléments :

1. les dimensions de la flexibilité qui ont été traitées dans notre échantillon. On perçoit immédiatement que de nombreux efforts ont porté sur la variabilité et la granularabilité, deux travaux apportent une réelle distensibilité, deux ont apporté une seconde stratégie d'apprentissage, un seul propose de nombreux éléments de solution et aucun ne supporte le polymorphisme. Aucune des approches évaluées dans ce chapitre n'apporte de réponse à l'ensemble des besoins des concepteurs et des développeurs tels que nous les avons analysés dans le chapitre 2. Même lorsqu'on cumule leurs possibilités (en imaginant construire un modèle de processus qui agrège les flexibilités les plus fortes des différentes approches, si tant est que ce soit possible), ce qui est représenté par la zone grisée, le polymorphisme et les stratégies d'apprentissage resteraient à augmenter.
2. L'entrelacement des traits représentant chaque évaluation montre (empiriquement) que Promote permet de distinguer les différents modèles de processus : si tel n'était pas le cas, par exemple si les graduations étaient insuffisamment précises pour distinguer des approches, les traits seraient très superposés. Bien sûr, une interprétation aussi intuitive d'un diagramme ne constitue pas une preuve scientifiquement valide de cette capacité à distinguer les modèles de processus les uns des autres. Elle constitue simplement un indice favorable.

La comparaison fine des modèles n'est cependant pas possible avec un tel diagramme. C'est pourquoi nous présentons aussi la figure 35, qui montre les mêmes informations mais sous la forme d'un diagramme en barres. Cette figure permet plus facilement de lire l'évaluation de chaque approche et de les comparer entre elles.

Elle montre clairement qu'une granularabilité maximum est offerte par sept approches, tandis que la variabilité maximale n'est atteinte que par une approche, celle de C. Hug. Trois approches, (Lucid/Star)*, le Work Product Pool (WPP) et CMPF offrent cependant une variabilité assez forte. Seul Rapid Application Development offre deux stratégies d'apprentissage, les autres approches n'en proposant qu'une. La distensibilité est maximale dans CMPF, assez présente dans XPP, un peu présente dans (Lucid/Star)* et Extreme Programming, et totalement absente dans les autres approches. La réutilisation est possible dans MACAO mais de façon limitée, elle est forte dans CPMF et absente dans les autres approches. Le polymorphisme, enfin, n'est proposé par aucune approche.

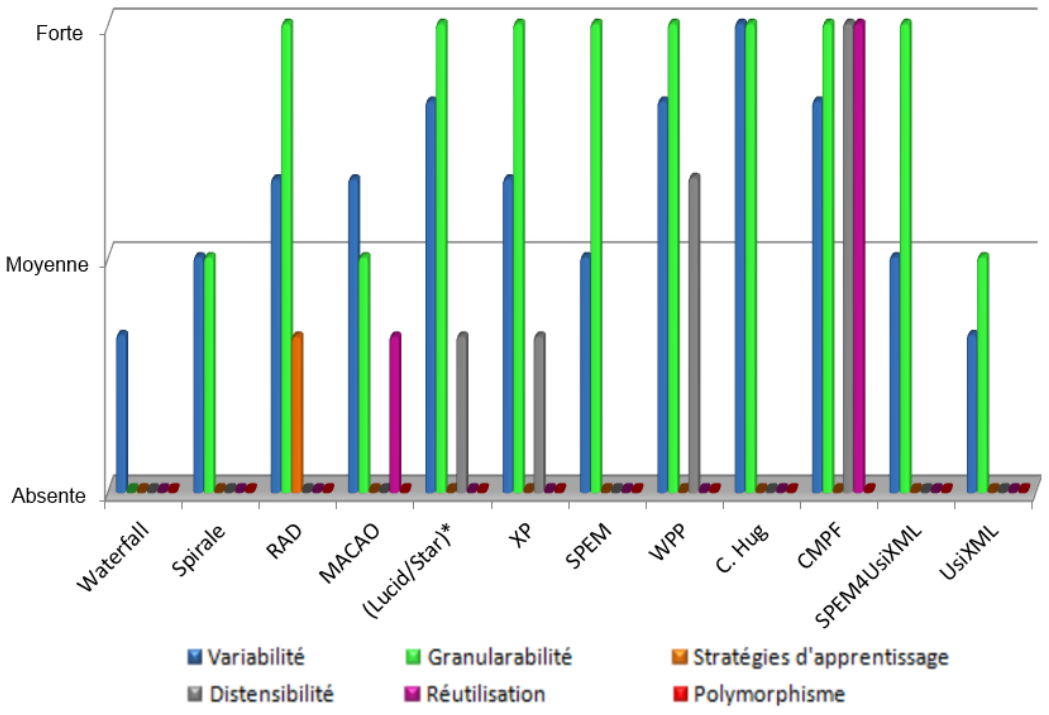

Figure 35 -Synthèse des évaluations des six formes de flexibilité dans les modèles de processus

Résumé du chapitre 4

Dans ce chapitre, nous avons présenté l'évaluation de la flexibilité de douze modèles et métamodèles de processus que nous avons répartis en trois catégories. Parmi les approches prescriptives, nous avons évalué la flexibilité du modèle du Waterfall (Royce 1970), du modèle en Spirale (Boehm 1986), de Rapid Application Development (RAD) (Martin 1991), de MACAO (Crampes 2002), et de (Lucid/Star)* (Helms 2001). Dans la catégorie des méthodes agiles, nous avons évalué la flexibilité Extreme Programming (Beck 1999a). La dernière catégorie est celle des approches constructivistes, c'est-à-dire des métamodèles ou des méthodes permettant de construire des modèles de processus. Dans cette catégorie, nous n'avons pas évalué directement la flexibilité de l'approche, qui n'intéresse pas les concepteurs et les développeurs. Nous nous sommes focalisés sur le potentiel de flexibilité des modèles de processus créés par ces approches. Nous avons ainsi évalué SPEM (Object Management Group 2008), le Work Product Pool (Gonzalez-Perez & Henderson-Sellers 2008), l'approche de construction situationnelle de C. Hug (Hug 2009), le CMPF (Golra 2014), ainsi que le métamodèle et le modèle de processus proposés par le Consortium UsiXML (UsiXML Consortium 2011).

Nous avons montré que l'évaluation de ces approches permet de mettre en évidence leurs similitudes et leurs différences en termes de flexibilité. Nous avons aussi montré qu'aucune approche ne couvre à elle seule toutes les formes de flexibilité. Nous avons aussi montré que les dimensions des stratégies d'apprentissage et du polymorphisme sont très peu prises en compte, et que seul le CMPF offre pleinement la distensibilité et la réutilisation.

Deuxième Partie

Deuxième Partie

Dans la première partie, nous avons montré qu'aucun modèle de processus n'offre toutes les formes de flexibilité attendues par les concepteurs et les développeurs. Ce constat nous amène, dans cette seconde partie, à proposer une solution pour la création de modèles de processus flexibles.

La seconde partie de cette thèse comporte les chapitres suivants :

- Le chapitre 5 présente M2Flex, un métamodèle de processus qui permet de créer des modèles de processus intégrant toutes les formes de flexibilité.
- Afin de faciliter la mise en œuvre de ce métamodèle, nous présentons dans le chapitre 6 D2Flex, un outil de conception de modèle de processus, et R2Flex, un outil qui permet l'exécution des modèles définis dans D2Flex

5 M2FLEX : METAMODELE DE PROCESSUS FLEXIBLE

Pour permettre la définition de modèles de processus flexibles comportant les six dimensions de la flexibilité à l'exécution telle qu'elle est spécifiée dans Promote, nous proposons dans cette section un métamodèle, M2Flex.

La première section de ce chapitre présente la structure de M2Flex. Nous détaillons ensuite comme cette structure permet de créer des modèles de processus avec toutes les formes de flexibilité définies dans Promote.

5.1 Vision globale

La figure 36 présente les paquetages de M2Flex. Comme cela a été proposé dans plusieurs modèles de processus orientés sur les buts, comme KAOS (Dardenne et al. 1993), I* (Yu 1995) ou la MAP (Rolland et al. 1999), nous considérons ici qu'un modèle de processus (*ProcessModel*) est considéré comme un « *ensemble d'étapes partiellement ordonnées,*

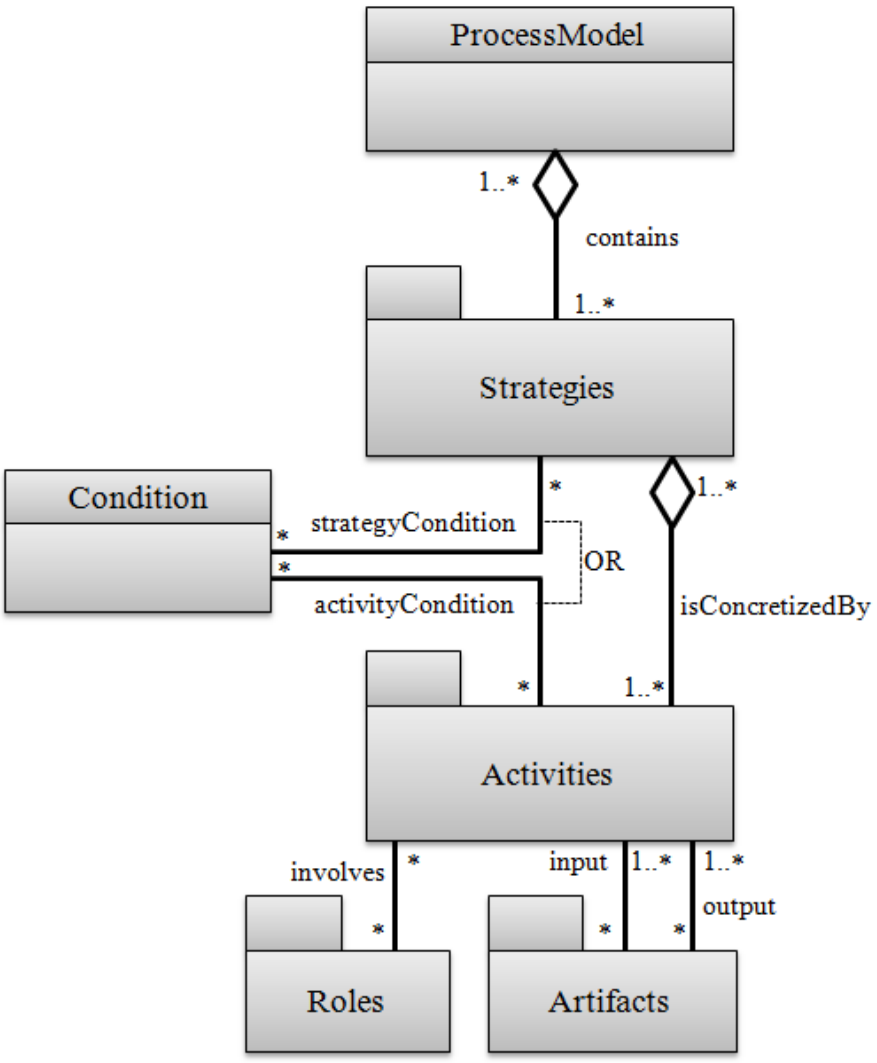

Figure 36 - Vue d'ensemble des paquetages de M2Flex

dont l'intention est d'atteindre un but » (Feiler & Humphrey 1993), ce qui permet de capturer les intentions à un niveau général (le « *pourquoi* ») avant d'être raffiné en activités (le « *comment* ») (Rolland 2007) et en produits (le « quoi »).

Comme il est possible d'atteindre les buts de différentes façons, ces possibilités sont représentées comme des *stratégies*. Une stratégie peut être associée à une *condition*, par exemple pour pouvoir exprimer qu'adopter une approche centrée sur l'utilisateur requiert que des utilisateurs soient disponibles.

Les stratégies sont concrétisées en *activités*, représentant les tâches opérationnelles à réaliser. Les activités produisent des artefacts à partir d'autres artefacts. Ainsi, dans le cycle en V (McDermid & Ripken 1984), la phase de conception suit la phase de spécification. Implicitement, cela signifie que le document des spécifications est produit par l'activité de spécifications et utilisé dans l'activité de conception.

Les activités sont aussi associées à des *rôles*, de façon à exprimer que des compétences ou des outils peuvent être requis pour réaliser une tâche. Par exemple, l'activité de conception du cycle en V (McDermid & Ripken 1984) ne peut être réalisée que par un acteur qui a des compétences en tant que concepteur. Les activités peuvent aussi être soumises à des *conditions*. Ceci a pour objectif de permettre l'expression de restrictions sur la faisabilité d'une activité. Par exemple, dans Rapid Application Development (Martin 1991), l'activité « *considérer la participation* [de personnes d'autres entreprises] » n'est réalisée que dans le cas où « *le système est utilisé* [par de telles personnes] ». Cette contrainte ne peut pas être exprimée par une dépendance à un artefact ou à un rôle. Les conditions associées aux activités permettent de l'exprimer.

Dans les sections suivantes, nous détaillons les paquetages que nous venons de présenter brièvement. Dans les diagrammes présentant ces paquetages, les classes décrites dans une section sont présentées avec un arrière-plan gris, les classes qui appartiennent à un autre paquetage sont représentées avec un arrière-plan blanc. Les nombres indiqués en blanc sur fond gris dans le descriptif font référence aux mêmes nombres dans la figure illustrant la section. Notre objectif étant ici de détailler comment M2Flex permet de créer des modèles de processus flexibles, nous ne détaillons pas exhaustivement tous ses attributs. Enfin, les attributs dont le nom est suivi du nombre 2 en exposant (par exemple, status^2) sont des « champs simples » incarnant une « instanciation profonde » (Atkinson & Kühne 2001) : lorsque le métamodèle est instancié, ces attributs sont réifiés en attributs identiques dans le modèle résultant (et, comme d'habitude, en valeurs au niveau de l'objet qui instancie le modèle). Nous utilisons ce mécanisme pour imposer, dans le métamodèle, des attributs que nous estimons nécessaires dans le modèle de processus.

5.2 Le paquetage Stratégies

La figure 37 montre la classe représentant un modèle de processus et les détails du paquetage Stratégies. Un modèle de processus, modélisé par la classe *ProcessModel* ❶, a un nom, un ou des auteurs et une date de publication. C'est une agrégation d'éléments du

paquetage Stratégies, une même stratégie peut être utilisée dans plusieurs modèles de processus.

Nous cherchons à exprimer que, au sein d'un modèle de processus, différentes démarches (au sens général du terme) peuvent mener d'une étape de haut niveau à une autre étape de haut niveau. En nous inspirant des travaux réalisés pour définir le métamodèle de la MAP (Rolland et al. 1999), nous modélisons ces éléments comme des buts et des stratégies.

Les pairs de buts (*GoalsPairs* ❷) représentent des couples de buts (*Goal* ❸), chaque couple étant associé à un but source et à un but cible. Un but représente un objectif important du modèle de processus. Par exemple, dans RAD (Martin 1991), le modèle de processus définit quatre étapes de haut niveau : (1) la définition des besoins, (2) la

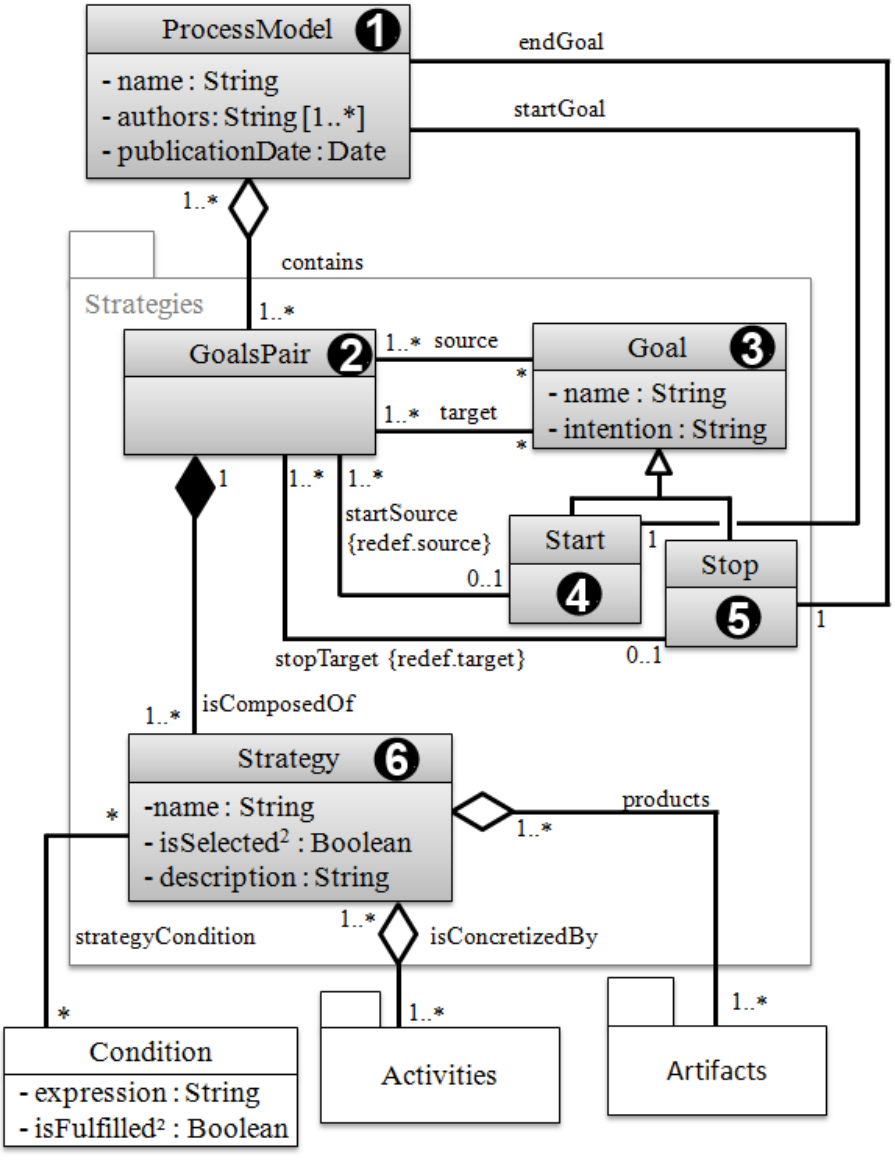

Figure 37 - Le paquetage Stratégies

« conception utilisateur » (« *user design* »), (3) la construction, qui regroupe la conception détaillée et l'implantation, et (4) la finalisation et la mise en place (« *cutover* »). Pour représenter RAD avec M2Flex, nous convertirions ces étapes en buts, en les nommant classiquement avec un verbe, par exemple « Définir les besoins », « Concevoir le système », etc. Un but, tel que nous le modélisons, a un nom et une intention, qui est une description de son dessein.

Le but Début (*Start* ❹) représente le point de départ du processus, le moment où aucune activité n'a encore été réalisée. Le but Fin (*Stop* ❺) représente la fin du processus, où toutes les stratégies choisies ont été terminées. Un modèle de processus comporte un unique but Début et un et un seul but Fin. Il y a donc au moins une paire de buts qui comporte le but Début comme source et au moins une paire de buts qui comporte le but Fin comme cible. Le modèle de processus peut en revanche comporter plusieurs paires avec l'un de ces deux buts, par exemple, il peut, en partant du but Début, proposer plusieurs objectifs premiers, comme « Définir les besoins » ou « Lister les fonctionnalités à réaliser ». Un modèle de processus ne peut contenir qu'un but Début et un but Fin : les associations *StartGoal* et *EndGoal* expriment cette contrainte grâce aux multiplicités valant 1.

Plusieurs stratégies (*Strategy* ❻), qui modélisent les différentes façons d'atteindre un but, peuvent relier deux buts, ce qui est représenté par la composition entre une paire de buts et la classe Stratégie. Par exemple, l'analyse des besoins dans le cycle en V (McDermid & Ripken 1984), qui serait représentée ici par un but « Décrire les besoins », pourrait être réalisée en utilisant une approche centrée sur l'utilisateur (Norman & Draper 1986), une approche d'étude des besoins comme celle définie dans la Map (Rolland et al. 1999) ou les sessions JRD proposées dans Rapid Application Development (Martin 1991). La figure 38 montre une représentation graphique de ces différentes stratégies : les buts sont représentés comme des « cibles » à atteindre et les stratégies comme des arcs entre ces cibles.

Une stratégie comporte un nom et une description. Lors de l'exécution du modèle de processus (instance de M2Flex), une stratégie peut ou non être sélectionnée par l'équipe de développement : ceci est modélisé par l'attribut estSelectionnée (*isSelected* dans la figure 37) en instanciation profonde. Une stratégie peut être associée à des conditions qui

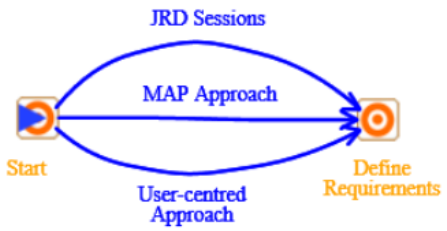

Figure 38 - Différentes stratégies entre deux buts.

représentent, comme nous l'avons dit plus haut, les contraintes qui doivent être remplies avant que la stratégie puisse être mise en œuvre. Par exemple, une stratégie centrée sur les utilisateurs (Norman & Draper 1986) nécessite l'agrément du client et la disponibilité des utilisateurs finaux.

Les différentes stratégies conduisant d'un but à un autre but offrent différents chemins parmi lesquels l'équipe de développement en choisira un, lors de l'exécution du modèle de processus. Elles apportent une première forme de variabilité. Leur concrétisation en activités offre un premier niveau de granularité.

5.3 Les conditions

La classe Condition exprime les contraintes auxquelles sont soumises les activités et les stratégies. Par exemple, comme le montre la figure 39, une activité d'évaluation de l'utilisabilité d'un logiciel pourrait suggérer de faire appel à un expert du domaine. L'implication de cet expert peut être modélisée grâce aux rôles requis par l'activité. Cependant, le budget nécessaire pour financer l'opération ne peut pas être représenté par un rôle. Cette activité devrait alors être soumise à une condition indiquant le budget nécessaire à sa réalisation.

La classe Condition comporte deux attributs principaux : une expression, qui peut être évaluée à vrai ou à faux, et un attribut estSatisfaite (*isFulfiled*) en instanciation profonde, et qui sera donc valorisé lors de l'exécution du modèle de processus, indique si la condition est satisfaite. Une condition peut contraindre (au moins) une stratégie ou (au moins) une activité. La même condition ne peut pas être associée à la fois à des stratégies et à des activités : une stratégie étant une agrégation d'activités, il semble difficilement imaginable que la même condition puisse s'appliquer à ces deux types d'éléments.

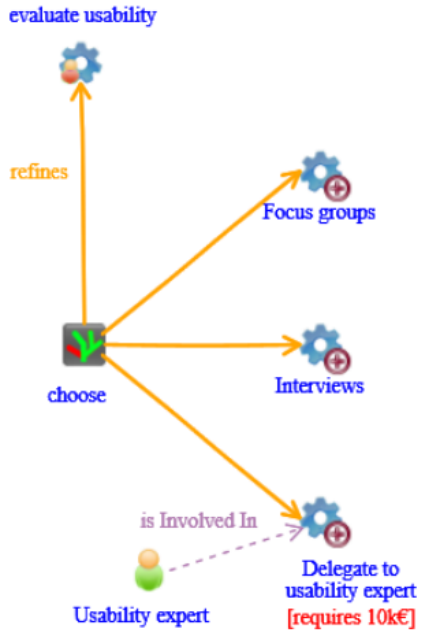

Figure 39 - Une condition sur une activité.

Cependant, une condition peut être associée à plusieurs stratégies ou à plusieurs activités car, comme le modèle de processus est susceptible de proposer plusieurs stratégies équivalentes, elles peuvent toutes dépendre de la même contrainte. De façon similaire, des activités concrétisant des stratégies équivalentes peuvent être soumises à des contraintes similaires.

5.4 Le paquetage Activités

Comme le montre la figure 40, une stratégie est concrétisée par une ou plusieurs Activités (*Activity* ❶), ce qui constitue une première possibilité de créer des niveaux de détails.

Les modèles de processus structurent les activités, par exemple en séquences d'activités successives. Ainsi, Scrum (Schwaber 1995) définit un Sprint, qui comporte une séquence itérative d'activités comme « Mise à jour la liste de fonctionnalités » (« *update backlog* ») ou « Réunion de planification du Sprint » (« *Sprint planning meeting* »). Pour représenter ces structures, les activités sont modélisées en utilisant un patron Composite : une Activité peut être soit une activité élémentaire (*ElementaryActivity* ❷) soit une activité complexe (*ComplexActivitiy* ❸), qui sera à son tour constituée d'Activités. Cette structure permet d'offrir plusieurs niveaux de détails dans les activités et constitue une seconde forme de granularabilité.

Une activité complexe permet la composition et l'ordonnancement d'activités élémentaires et complexes. L'ordonnancement est défini grâce aux artefacts et aux opérateurs associés aux activités complexes. Nous ne détaillerons pas ici ces opérateurs, qui font l'objet de la prochaine section.

Une activité a un nom, elle peut être itérative, comme par exemple le Sprint de Scrum (Schwaber 1995) que nous avons déjà mentionné. Elle peut aussi être incrémentale, indiquant que les artefacts qui en résultent sont construits incrémentalement. Elle peut aussi disposer de chemins vers un tutoriel et une documentation. La documentation est une explication détaillée de l'activité, tandis que le tutoriel est un exemple de mise en œuvre décrite pas à pas : ces deux éléments permettent de faciliter l'apprentissage du modèle de processus. Une activité peut enfin être reformulée par une ou plusieurs autres activités grâce à l'association « *rephrases* » ❹. Cette reformulation permet d'exprimer une activité avec plusieurs niveaux de langages ou plusieurs types de vocabulaire, ce qui est une stratégie d'apprentissage puisqu'on peut exprimer une activité dans un langage destiné à un expert et la reformuler dans un langage destiné à un concepteur ou un développeur novice. Une activité peut être associée à des rôles ❺. Ces rôles permettent d'indiquer les compétences requises par l'activité. Dans le cas des activités complexes, cela permet par exemple d'indiquer que seul le chef de projet peut exécuter certains opérateurs, comme par exemple les choix entre différentes propositions, qui peuvent être critiques dans certaines parties du modèle de processus.

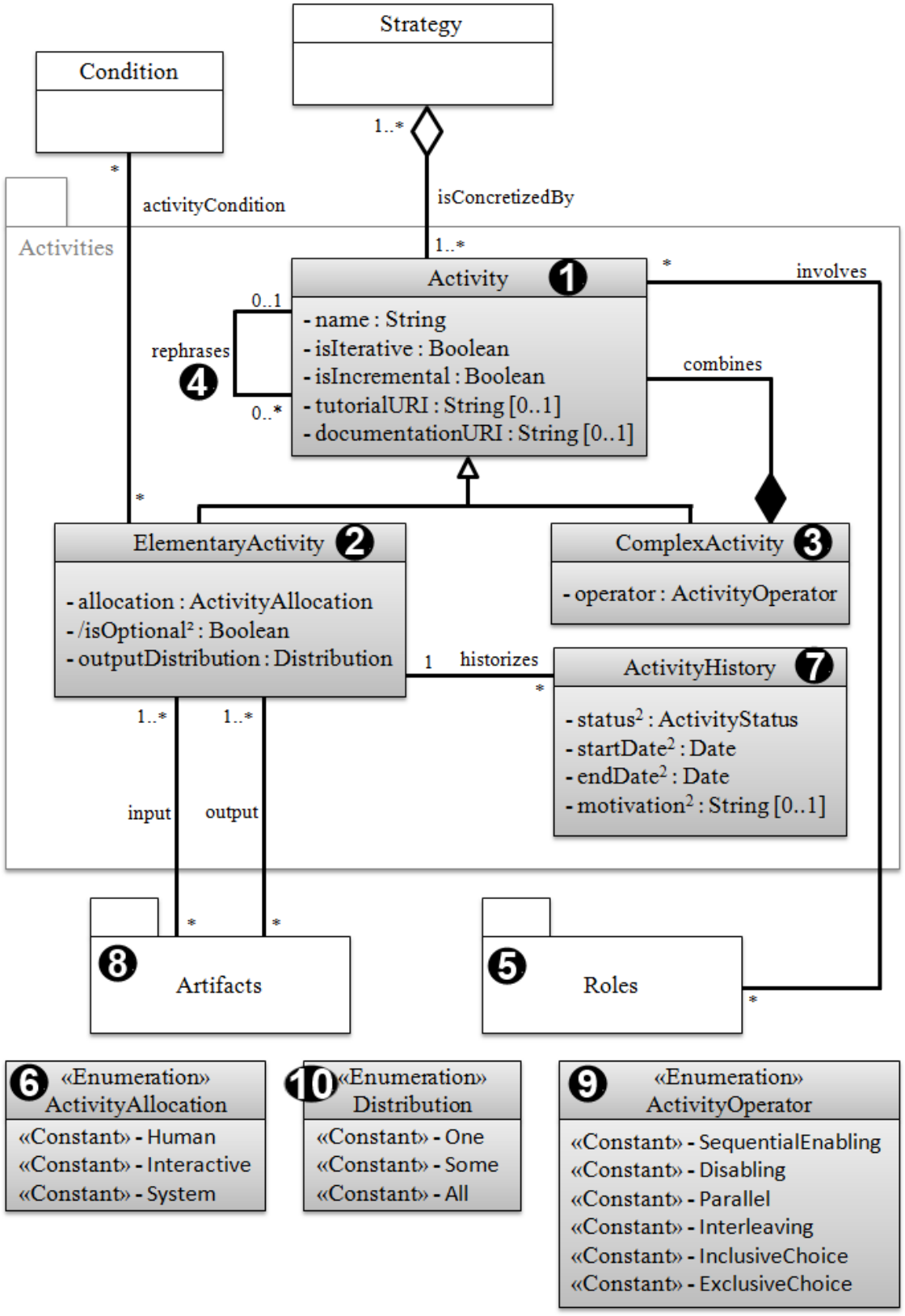

Figure 40 - Le paquetage Activités de M2Flex

Une activité élémentaire a une allocation, qui indique, comme le spécifie l'énumération correspondante ❻, si l'activité doit être réalisée par un être humain sans interaction avec le système (« *Human* »), par un être humain en interaction avec le système (« *Interactive* ») ou par le système seul (« *System* »). Par exemple, dans une phase de tests, une activité interactive pourrait être « Codage des tests unitaires », une activité du système pourrait être « Exécution des tests unitaires » et une activité humaine pourrait être « Réunion de l'équipe pour l'analyse des résultats des tests ». Une activité élémentaire a aussi un attribut calculé en instanciation profonde pour indiquer si elle est optionnelle (*isOptional*). Cet attribut indique si l'activité peut **ne pas** être exécutée, c'est-à-dire s'il existe un chemin dans le modèle de processus qui ne l'inclut pas. Nous verrons par la suite que cet attribut intervient dans la vérification de la validité du modèle de processus. Nous ne détaillons pas ici l'attribut distribution de la sortie (*outputDistribution*), sur lequel nous reviendrons dans la section sur les artefacts.

Une activité élémentaire est associée à la classe HistoriqueActivité (« *ActivityHistory* » ❼), de façon à mémoriser les informations sur son évolution. Dans cette classe, quatre attributs sont valorisés pendant l'exécution du modèle de processus : le statut de l'activité (sélectionnée, rejetée, en cours,…), les dates de début et de fin du statut, ainsi que, si besoin, la raison de l'évolution. Cette raison est particulièrement intéressante lorsque l'activité est choisie ou rejetée, afin d'intégrer une sémantique associée aux aspects évolutifs du processus, comme le recommandent les modèles de processus orientés décision (Jarke et al. 1993).

Une activité élémentaire peut aussi, comme nous l'avons déjà mentionné, nécessiter des artefacts ❽ en entrée (association « *input* ») et/ou en sortie (association « *output* »).

5.5 Les opérateurs

La classe des activités complexes contient un attribut opérateur, qui doit être l'un des opérateurs énumérés (« *ActivityOperator* », fig. 40, n° ❾). Ces opérateurs sont basés sur un sous-ensemble de ceux utilisés en modélisation de tâche utilisateur (Nóbrega et al. 2006), tous les opérateurs n'étant pas requis ici. Par exemple, la nuance entre « *Activation* » (*Une tâche en active une autre lorsqu'elle se termine*) et « *Activation avec passage d'information* » (*Quand elle se termine, une tâche en active une autre et lui fournit une information*) ne nous est pas utile, car elle est exprimée grâce aux artefacts en entrée et en sortie des activités. Après avoir analysé quels opérateurs étaient utiles dans notre cas, nous avons défini un ensemble de 6 opérateurs n-aires. Dans la description qui suit, un élément est une activité ou une activité élémentaire.

Activation : après qu'un élément source soit achevé, les éléments cibles peuvent commencer. Cet opérateur n'est pas requis lorsque la dépendance entre les éléments peut être exprimée à travers les artefacts, comme le montre la figure 41[16]. Si une activité B dépend d'un produit fabriqué par une activité A, la dépendance est déjà exprimée par les

[16] Dans les exemples, les rouages bleus représentent des activités, les icônes carrées représentent des opérateurs et les documents représentent des artefacts. Les flèches représentent les relations entre les éléments.

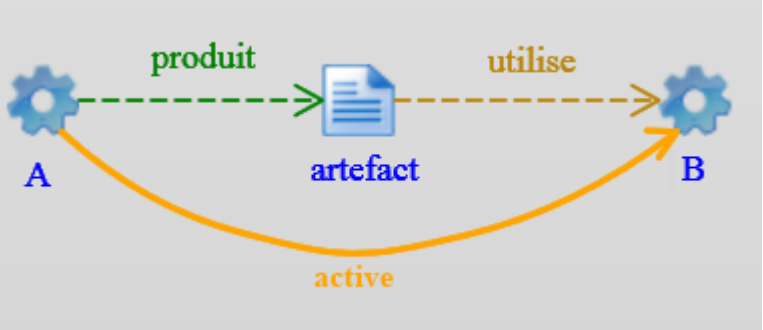

Figure 41 - La relation "active" entre A et B est exprimée par les dépendances avec "artefact"

associations de sortie de A et d'entrée en B et il est inutile de surcharger le modèle de processus avec une relation d'activation entre A et B. A l'inverse, cet opérateur est nécessaire lorsque B ne dépend pas d'un artefact produit par A mais qu'il est pertinent d'exprimer que B doit être réalisée après A. Par exemple, il n'est pas nécessaire d'attendre que la préparation des tests unitaires soit achevée avant de commencer le code, mais un modèle de processus pourrait vouloir imposer cette contrainte, ce qu'il pourrait faire grâce à cet opérateur.

Désactivation : les éléments ciblés par cet opérateur sont désactivés lorsque l'élément source est achevé. Par exemple, dans une approche agile, lorsque le client décide de ne plus ajouter de fonctions au produit, cela entraîne la désactivation des itérations de réalisation de fonctionnalités.

Parallèle : les éléments ciblés par l'activité contenant cet opérateur peuvent être réalisés en parallèle, c'est-à-dire dans le même moment, par des rôles différents. Par exemple, un modèle de processus peut spécifier que les tests unitaires doivent être écrits en parallèle avec la création du code (contrairement à l'exemple donné ci-dessus). Ces deux activités seraient alors combinées par un opérateur « parallèle », chacune d'entre elles étant associée à son ou ses rôles propres.

Entrelacement : avec cet opérateur, les éléments ciblés sont exécutés en parallèle (comme dans le cas précédent), mais cette fois, ils sont réalisés par le même rôle, qui peut donc librement passer d'une activité à l'autre quand il le souhaite. Un exemple d'activités entrelacées est la conduite de tests (non automatisés) : l'agent doit en même temps réaliser les cas de test et enregistrer leurs résultats, il passe sans cesse d'une tâche à l'autre.

Choix exclusif : l'équipe de développement (ou une personne jouant le rôle associé à l'opérateur, s'il est défini) peut choisir entre différentes activités. Ce choix est ici exclusif, les activités sont donc censées être soit équivalentes, soit incompatibles les unes avec les autres. Ainsi le générateur de code Oxygen (Tech-IS 2012) propose la création de composants parmi lesquels on trouve, comme représenté sur la figure 42, un module d'accès aux données ❶, une interface d'accès aux données (qui est optionnelle) ❷ et des objets de transfert de données ❸. Le développeur doit choisir d'utiliser ou non l'interface optionnelle. La modélisation dans M2Flex de ces possibilités se traduit par un choix exclusif ❹.

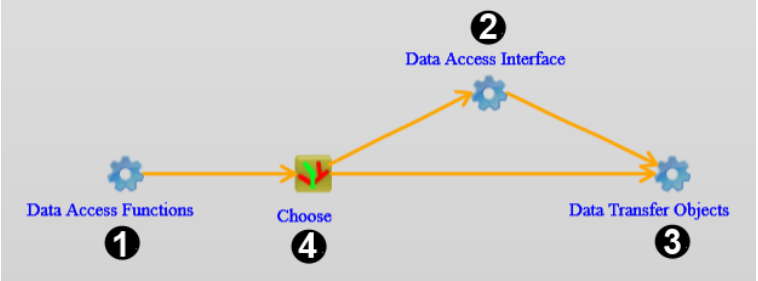

Figure 42 - La modélisation de la proposition de mise en œuvre de modules de Oxygen

Choix inclusif : à l'inverse du choix exclusif, l'équipe peut ici choisir plusieurs activités parmi celles proposées. C'est par exemple le cas si le modèle de processus laisse un libre choix parmi différentes possibilités (« focus groups », entretiens, questionnaires,…) pour mettre en œuvre une approche centrée sur l'utilisateur (Norman & Draper 1986) : ces activités ne sont pas mutuellement exclusives et peuvent être utilisées conjointement pour affiner l'analyse.

Les deux opérateurs de choix permettent de créer des variants, ce qui constitue une seconde forme de variabilité.

5.6 Le paquetage Artefacts

Les activités peuvent être associées à des artefacts, comme indiqué sur la figure 43 : elles peuvent les utiliser en entrée ou les produire en sortie, cette production pouvant prendre la forme d'une création ou d'une modification. Les artefacts associés à une activité ont un statut (« *Status* » ❶). Ainsi, l'activité « Validation des besoins avec les utilisateurs » prend en entrée une étude des besoins au statut « proposée » et la fait passer à un autre statut, par exemple « validée » ou « rejetée ». L'activité prend en entrée un statut particulier d'un artefact donné : la classe ActivitéElémentaire est donc associée à la classe Statut, qui est à son tour associée à la classe Artefact (*Artifact* ❷).

Un statut est composé d'un nom, d'une date, d'un attribut calculé indiquant le caractère optionnel du statut et d'un booléen qui indique si le statut (et donc l'artefact) est pré-existant ou s'il est produit par le processus lors de son exécution. Par exemple, le modèle de processus peut proposer de réutiliser un composant (un modèle, une classe,…) défini antérieurement. Le nom est une chaîne de caractères que le concepteur du modèle de processus définit librement. Nous avons choisi de laisser le nom du statut sous la forme d'un texte libre. Pour garantir la cohérence du modèle de processus, il aurait pu paraître souhaitable de préférer des énumérations. Cependant, ces énumérations auraient dû contenir un très grand nombre de propositions, ce qui aurait rendu leur utilisation difficile. De plus, il n'aurait pas été possible de garantir la complétude d'une telle énumération : de nouveaux statuts d'artefacts peuvent apparaître au fil des projets et des situations. Il aurait alors fallu compléter le métamodèle, ce qui ne nous paraît pas souhaitable. L'utilisation d'un champ en texte libre évite ces contraintes. Il est cependant nécessaire de s'assurer de la cohérence entre les statuts produits par des activités et les statuts utilisés par d'autres activités, nous verrons dans la section de validation du modèle

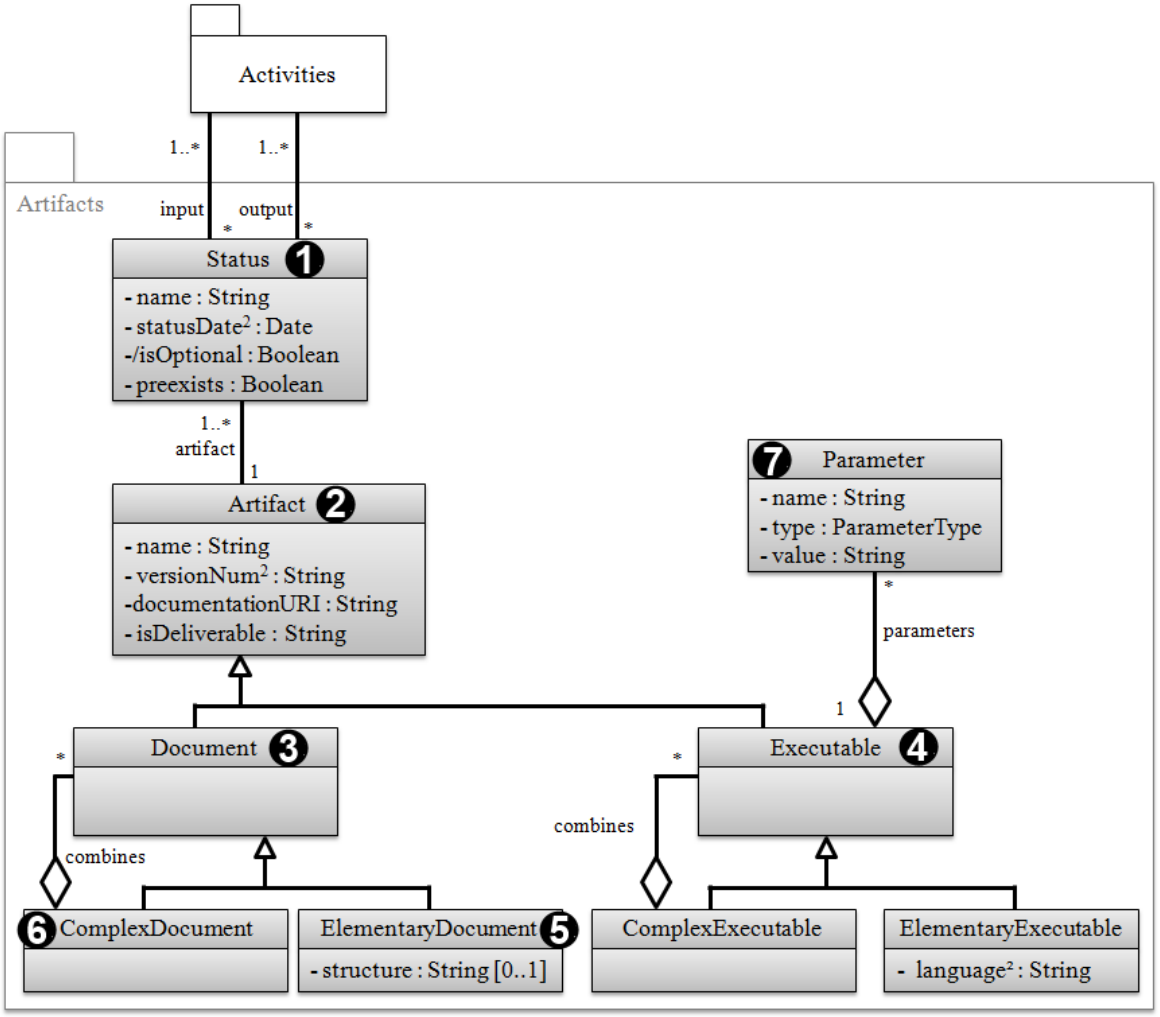

Figure 43 - Le paquetage Artefacts

de processus comment cette contrainte est vérifiée. La date d'un statut permet simplement de conserver une information sur le moment de la création de ce statut : elle est donc valorisée lors de l'exécution du modèle de processus. L'attribut estOptionnel (« */isOptional* ») est valorisé en fonction des chemins possibles dans le modèle de processus : s'il existe un chemin qui ne requiert pas ce statut, il est défini comme optionnel. Enfin, un statut peut être préexistant, ce qui permet de définir des éléments qui ne sont pas produits par l'équipe de développement, mais dont elle se sert, comme des programmes créés dans un autre projet, ou un dossier d'étude des besoins fourni par le client. Ces artefacts pouvant cependant évoluer au fil du projet (et donc changer de statut) ou être requis à un certain statut dans une activité et optionnels à un autre statut dans une autre activité, l'attribut estOptionnel doit bien prendre place dans la classe Statut. La préexistence possible des artefacts permet d'offrir une forme de réutilisation.

Un artefact ❷ a un nom (« *name* »), un numéro de version (« *versionNum* ») et peut être livrable (« *isDeliverable* »). Comme les activités, il a un attribut indiquant le chemin vers une documentation (*documentationURI*), ce qui permet d'offrir des possibilités d'apprentissage supplémentaires. Le numéro de version permet de gérer les itérations sur

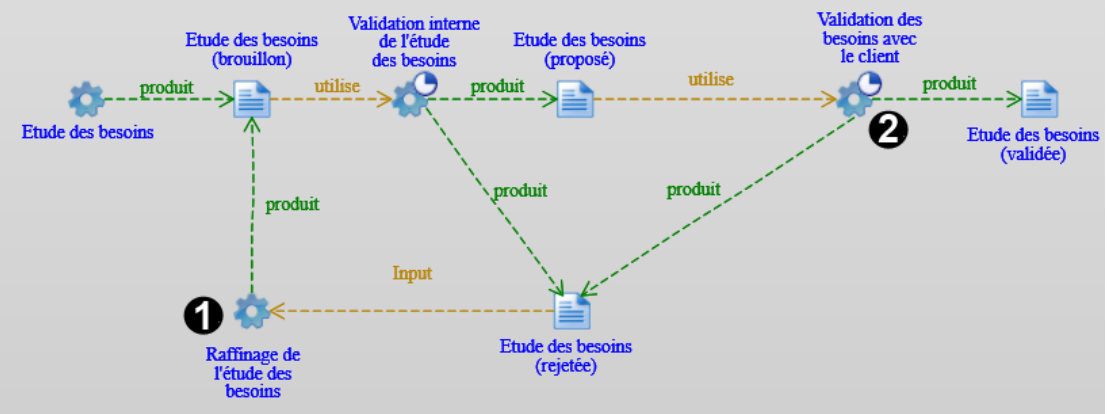

Figure 44 - Un processus de création de l'étude des besoins

un document. Dans l'exemple proposé ci-dessus, nous avons dit que l'activité « Validation des besoins avec les utilisateurs » produit une étude des besoins au statut « validée » ou « rejetée ». Lorsque le statut est « rejetée », le modèle de processus propose de retravailler le dossier (figure 44, ❶). Après avoir retravaillé cette étude, elle doit de nouveau entrer dans la phase de validation. Mais en réalité, lors de l'exécution du modèle de processus, il s'agit d'une nouvelle version de l'étude. Le modèle de processus ne représente pas les versions, car il s'agit d'une représentation conceptuelle de ce qui se déroulera à l'exécution. Cependant, nous voulons permettre qu'au cours du processus, il puisse être possible de mémoriser le déroulement réel des actions. Cet attribut numéro de version relève donc de l'instanciation profonde.

Une activité peut produire des quantités variables d'artefacts. Par exemple l'activité « Etude des besoins » produit (en général) un dossier décrivant les besoins des utilisateurs. A l'inverse, l'activité « Questionnaires utilisateurs » produit un ensemble de réponses. De même, produire un projet en Java, c'est réaliser un certain nombre de paquetages et de classes. Pour permettre de représenter ces différentes possibilités, la classe Artefact (fig. 43, ❷) a été définie comme une classe abstraite que spécialisent deux patrons composites : les Documents (« *Document* », fig. 43, ❸) et les Exécutables (« *Executable* », fig. 43, ❹).

Les documents sont spécialisés en documents élémentaires (« *ElementaryDocument* », fig. 43, ❺) et en documents complexes (« *ComplexDocument* », fig. 43, ❻), qui eux-mêmes agrègent d'autres documents. On retrouve une structure similaire dans les exécutables. Cette structuration permet ici aussi de définir différents niveaux de détails. Un autre intérêt de cette modélisation est de permettre d'associer une activité Système avec un artefact exécutable, de façon à exécuter ce dernier. Un document élémentaire peut avoir une structure décrite dans son attribut *structure*, qui est une chaîne de caractère contenant une description de l'organisation du document en XML. Un exécutable est décrit par son langage de programmation, de façon à pouvoir le compiler si besoin et l'exécuter.

Des paramètres, comme un mot de passe ou le chemin d'un modèle, peuvent être requis par un exécutable, qu'il soit complexe comme un projet Java ou élémentaire

comme une classe. Ils sont modélisés dans la classe Paramètre (« *Parameter* », fig. 43, ❼), qui contient trois attributs : le nom du paramètre, son type (entier, booléen,…) et sa valeur (enregistrée sous la forme d'une chaîne de caractères et convertie si besoin). Cette structure ne permet pas de gérer des paramètres de type complexe (par exemple, un exécutable pourrait prendre un modèle en entrée et exécuter une transformation). Cependant, dans le cadre de nos travaux, nous avons toujours pu nous ramener à une information de type élémentaire (une chaîne de caractère donnant le chemin d'accès au modèle à la place du modèle lui-même, par exemple). Notre objectif étant centré sur la flexibilité à l'exécution et non sur une portée universelle du métamodèle, nous avons choisi de ne pas aller au-delà de cette structure simple.

Nous avons maintenant décrit la structure des activités et celle des artefacts. Il reste à donner quelques précisions sur les liens entre ces deux éléments. En premier lieu, les multiplicités des associations « input » et « output » permettent qu'une activité n'ait pas d'artefact en entrée et/ou en sortie. Ainsi, la toute première activité d'un processus n'a habituellement pas d'artefact en entrée, puisque l'équipe n'a encore rien produit (elle peut cependant utiliser des artefacts préexistants). De même, une activité, en particulier une activité purement humaine, peut ne produire aucun artefact, comme « Informer le client de la mise à disposition d'une nouvelle version du produit ».

L'exemple indiqué sur la figure 44 nous permet aussi de revenir sur l'attribut « *outputDistribution* », que nous n'avions pas détaillé dans la section sur les activités. Cet attribut permet d'indiquer si, parmi les artefacts (plus précisément, les statuts d'artefacts) en sortie d'une activité, tous sont requis ou non. Ainsi, dans l'exemple ci-dessus, il semble évident que l'activité « Validation des besoins avec les utilisateurs » ne pourra pas faire évoluer le statut de l'étude des besoins à la fois au statut « validée » et « rejetée ». Il était donc nécessaire d'indiquer qu'un seul des deux statuts serait produit (figure 44, ❷). L'attribut « *outputDistribution* » permet de représenter cette information. Il peut prendre trois valeurs (fig. 40, ❿) : tous (tous les statuts seront produits), certains (certains des statuts seront produits, mais à la conception on ignore combien), un (un seul parmi tous les statuts sera produit).

5.6.1 Le paquetage Rôles

La figure 45 montre le paquetage Rôles de M2Flex. La structure que nous proposons ici a été simplifiée par rapport aux besoins d'une application réelle. Cependant, elle correspond à nos besoins pour des travaux sur la flexibilité des modèles de processus à l'exécution.

Une activité est éventuellement associée à un ou plusieurs rôles. Lorsqu'aucun rôle n'est défini, cela lève toute contrainte sur la personne qui réalisera l'activité lors de la mise en œuvre du processus. Lorsque des rôles sont définis, seuls les agents qui seront associés à l'un de ces rôles pourront participer à la réalisation de l'activité. Trois classes spécialisent la classe Rôle : les rôles internes (« *Internal* », ❶), les rôles externes (« *External* », ❷) et les outils (« *Tool* », ❸). Les deux premiers sont des rôles réservés à des agents humains, ils sont internes quand ils correspondent à des compétences ou des

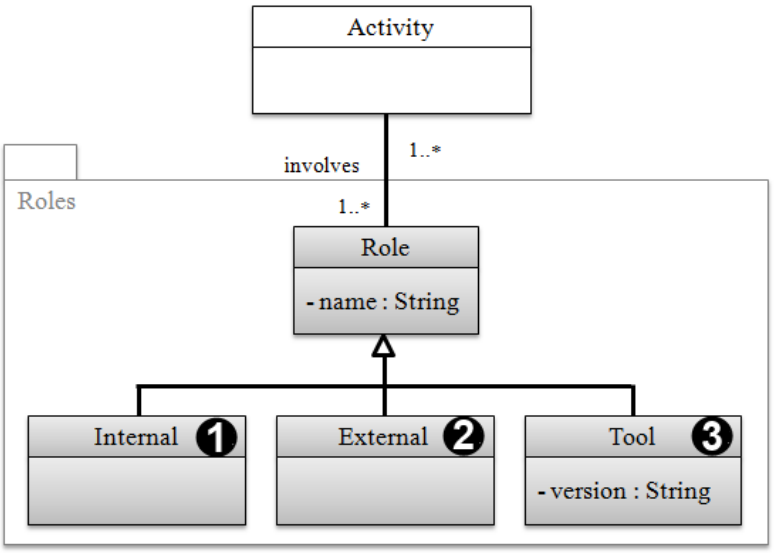

Figure 45 - Le paquetage Rôles

fonctions relevant de l'équipe de développement (chef de projet, développeur, concepteur,…) et externe lorsqu'ils traitent des autres parties prenantes (client, utilisateur final,...). Conceptuellement, ces deux classes sont similaires : elles servent à identifier les compétences requises par une activité et les compétences offertes par une personne. La division en deux classes présente cependant un bénéfice qui nous semble intéressant : elle permet de déterminer le nombre d'activités dans lesquelles les compétences des utilisateurs sont attendues, et permet ainsi de mesurer à quel point un modèle de processus est centré sur l'utilisateur. Ceci ne relève pas de la flexibilité, mais de l'approche que nous entendons proposer lors de la mise en application du métamodèle.

La dernière classe qui spécialise la classe Rôle est la classe Outil (« *Tool* », ❸), elle permet d'exprimer qu'en plus de compétences humaines, un outil (un logiciel, par exemple), est nécessaire pour réaliser une activité.

La description des paquetages de M2Flex est maintenant complète. Dans les sections qui suivent, nous commencerons par présenter comment un modèle de processus qui instancie M2Flex est structuré temporellement, et comment il est donc possible de définir l'enchaînement des activités (ou des artefacts, selon la focale choisie), puis nous détaillerons les formes de flexibilité offertes par le métamodèle.

5.7 Exécutabilité du modèle de processus

La conformité d'un modèle de processus à M2Flex ne suffit pas à assurer sa cohérence ni sa capacité à être exécuté concrètement. La figure 46 donne un exemple de modèle de processus conforme à M2Flex mais irréalisable par une équipe de développement : une stratégie « Développer » ❶ permet d'aller du but de départ ❷ au but de fin ❸. Cette stratégie est concrétisée en une activité « Développement » ❹, qui est elle-même raffinée en une séquence d'activités ❺ composée de l'étude des besoins ❻ et du codage ❼. L'étude des besoins produit le dossier décrivant les besoins des utilisateurs ❼, et le

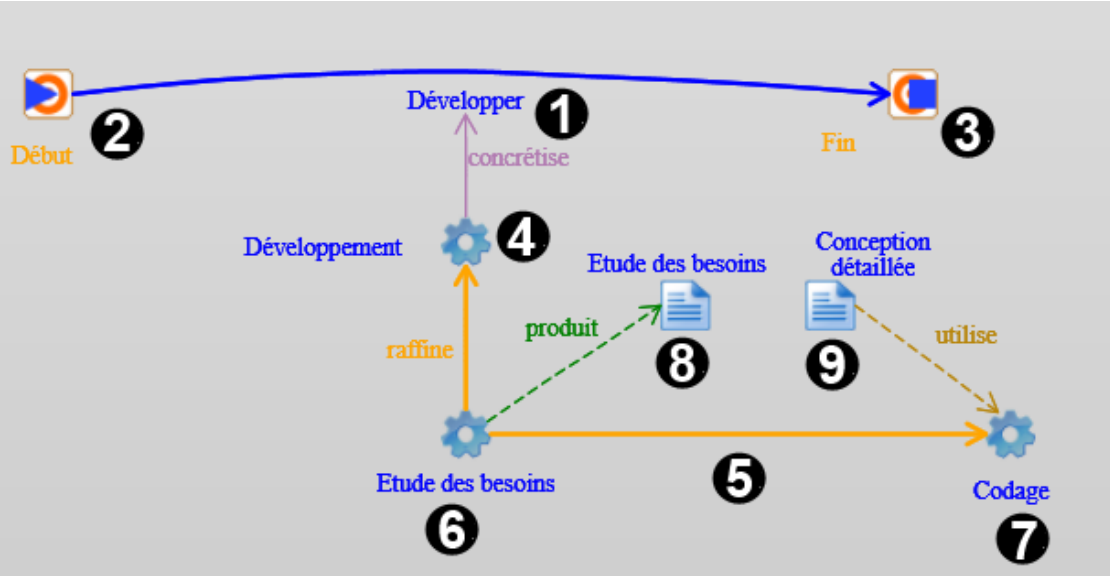

Figure 46 - Un modèle de processus conforme à M2Flex mais non exécutable

codage se base sur la conception détaillée ❽. Ce modèle de processus est conforme à M2Flex, puisqu'il instancie correctement tous les éléments requis par le métamodèle. Il n'est cependant pas exécutable, puisque l'activité codage prend en entrée un dossier de conception détaillée qui n'est jamais produit : l'équipe de développement ne pourra jamais réaliser cette activité si elle se conforme au modèle de processus. De plus, ce modèle ne produit jamais le logiciel que l'on pourrait attendre de lui, puisque l'activité codage ne produit aucun artefact.

On voit ainsi qu'il est nécessaire de rajouter des contraintes dans M2Flex pour assurer l' « exécutabilité » du modèle, que nous appellerons ci-dessous sa validité.

L'exemple ci-dessus nous montre que la première contrainte est que tout artefact (tout statut d'artefact) pris en entrée par une activité *a1* doit être produit par une autre activité *a2* ou être préexistant, ce que nous traduisons sous la forme suivante :

(1)
```
∀ a1 ∈ Activity, ∀ s ∈ Status,
s ∈ a1.input ⇒ ((∃ a2 ∈ Activity, s ∈ a2.output) ∨ (s.preexists))
```

Par ailleurs, il faut que cette activité a2 ne soit pas dans une stratégie incompatible avec la stratégie dans laquelle se trouve a1. Deux stratégies sont incompatibles lorsqu'elles ont le même but source ou le même but cible : dans ces deux cas, une seule d'entre elles pourra être sélectionnée à l'exécution. Dans la figure 47, les stratégies S1, S2, S6 et S7 sont incompatibles entre elles. Mais deux stratégies peuvent aussi être incompatibles sans avoir de but commun : toujours dans la figure 47, les stratégies S4 et S7 sont incompatibles alors qu'elles n'ont aucun but en commun. En effet, si à l'exécution la stratégie S7 est choisie, alors il n'est plus possible de choisir S4, et inversement.

Un modèle de processus définit plusieurs chemins possibles, qui correspondent à des séquences de stratégies. Le modèle de processus de la figure 47 définit les chemins, suivants pour aller de But1 à But4 : {S6}, {S1, S3, S5}, {S1, S4}, {S2, S3, S5} et {S2, S4}. En nommant σ(a) la fonction qui donne la stratégie à laquelle appartient l'activité a

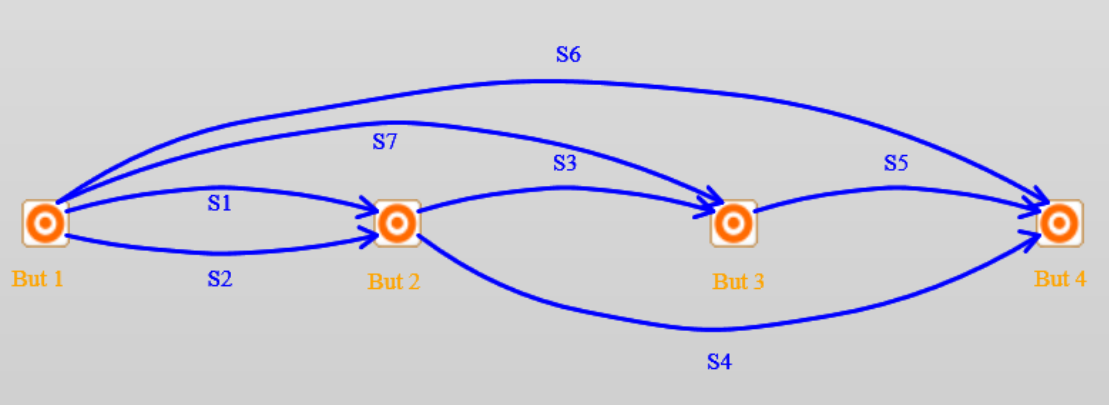

Figure 47 - Exemple de stratégies entre 4 buts

et φ(s) la fonction qui donne tous les chemins contenant la stratégie s, la contrainte (1) doit donc s'écrire :

(1') $\forall\ a1 \in Activity,\ \forall\ s \in Status,\ s \in a1.input \Rightarrow (\exists\ a2 \in Activity,\ s \in a2.output,\ \sigma(a2) \in \varphi(\sigma(a1))) \vee (s.preexists)$

L'activité a2 doit appartenir à au moins une stratégie S2, telle que S2 fait partie d'au moins un chemin C1 qui contient l'une des stratégies auxquelles appartient a1 et dans C1, S2 doit précéder S1.

Cette condition n'est cependant toujours pas suffisante pour valider le modèle de processus. Par exemple, dans une approche centrée sur l'utilisateur, plusieurs techniques peuvent être proposées. Un opérateur de choix inclusif permettra lors de l'exécution, de sélectionner les pratiques qui seront effectivement mises en œuvre. La pratique des questionnaires est alors optionnelle : elle peut ne pas être choisie. En conséquence, les artefacts résultants de cette activité (les questionnaires remplis par les participants) sont eux aussi optionnels. Dans ce cas, l'activité « Dépouiller les questionnaires » peut ne pas disposer des questionnaires remplis qu'elle prend en entrée. Elle doit donc être elle aussi optionnelle, alors qu'elle ne dépend pas directement d'un opérateur de choix. Ceci impose alors que le modèle de processus offre au moins un chemin qui ne transite pas par l'activité de dépouillement des questionnaires, faute de quoi il pourrait être impossible dérouler le processus jusqu'au bout.

D'autres contraintes sont identifiables, par exemple : une stratégie n'a de sens que si elle appartient à au moins un chemin allant du but de départ au but terminal, à défaut de quoi elle ne pourra jamais être utilisée. Nous ne définirons pas plus avant l'ensemble de ces conditions, car elles relèvent plus de la théorie des graphes que de la flexibilité des modèles de processus. Nous préférons donc, dans la section suivante, montrer comment M2Flex offre les six formes de flexibilité.

5.8 La flexibilité offerte par M2Flex

Nous avons présenté certaines formes de flexibilité au fur et à mesure des explications sur la structure de M2Flex. Dans cette section, nous résumons ces formes et ajoutons celles qui n'ont pas pu être présentées en lien direct avec un paquetage. La figure 48 présente un sous-ensemble de M2Flex : toutes les classes ne sont pas représentées, et tous les attributs ne sont pas présents. Seuls les éléments liés à la flexibilité ou à l'articulation des classes entre elles et des paquetages en eux sont représentés. Le paquetage Artefacts en particulier a été simplifié : seule la structure générale composite des artefacts a été reprise, sans la différentiation entre documents et exécutables. Cette figure nous servira à illustrer notre propos dans les sections suivantes.

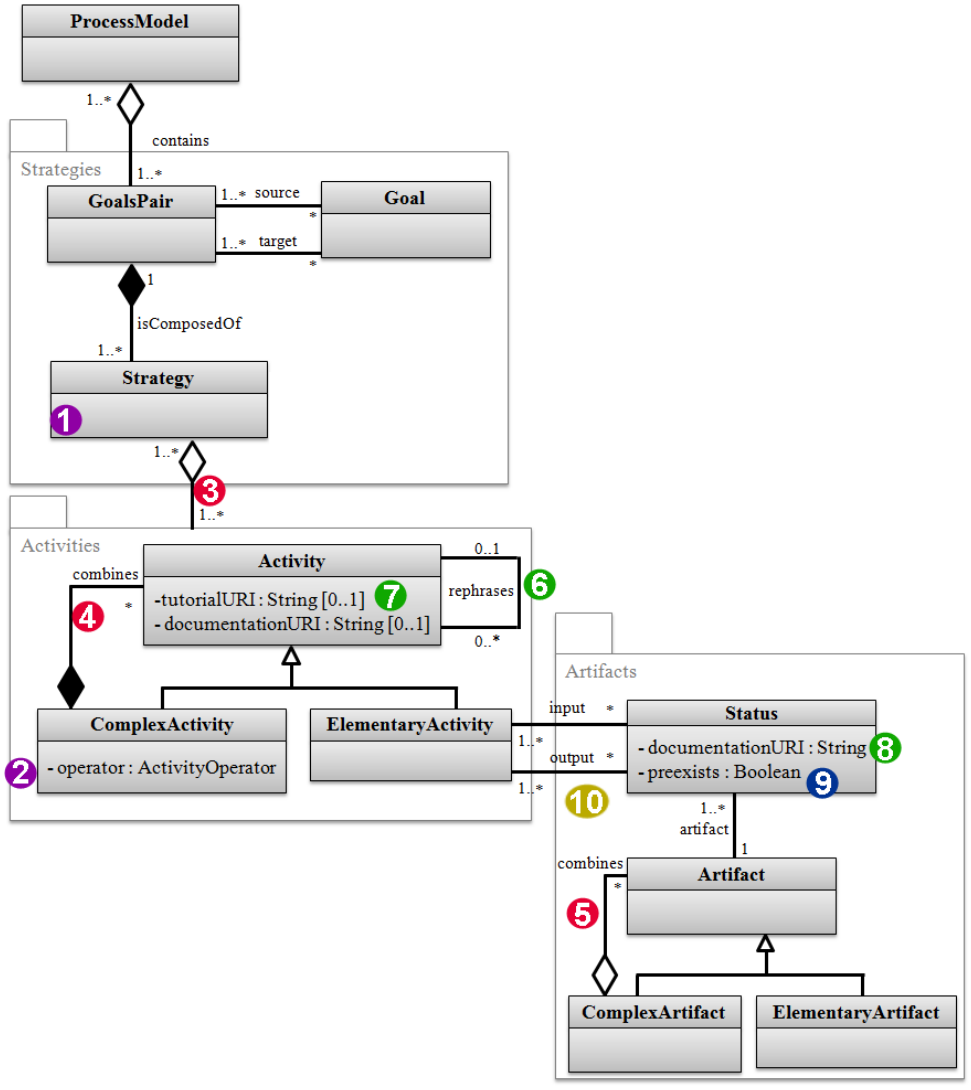

Figure 48 - La flexibilité offerte par M2Flex

5.8.1 Variabilité

La variabilité est définie dans Promote comme la capacité d'un modèle de processus à proposer différents chemins parmi les éléments qui le composent. Nous avons vu ci-

dessus que cette capacité est présente à deux niveaux dans M2Flex : d'une part, grâce aux différentes stratégies ❶ conduisant d'un but à un autre but et d'autre part grâce aux opérateurs de choix exclusif ou inclusif ❷ dans la structure des activités.

La structure de M2FLex rend possible pour un modèle de processus d'offrir des variants, mais n'impose pas qu'il y ait des variants : entre deux buts, il est possible de ne proposer qu'une stratégie (la multiplicité de la composition « *isComposedOf* » entre une paire de buts et des stratégies et de 1..*), il n'est pas non plus requis d'utiliser les opérateurs de choix entre des activités. Il aurait été possible de rendre la variabilité obligatoire, par exemple en imposant au moins deux stratégies entre des buts et en définissant une métrique comptant le pourcentage d'activités représentant des choix et en imposant une valeur minimale à cette métrique pour qu'un modèle de processus soit considéré comme valide.

Cependant, il n'est pas dans notre objectif d'imposer une flexibilité minimale aux modèles de processus. Au contraire, nous souhaitons pouvoir représenter des modèles rigides, soit afin de les « flexibiliser », soit afin d'en réutiliser des portions pertinentes dans des modèles plus flexibles. Cette contrainte n'a donc pas été définie dans M2Flex.

M2Flex permet donc la création de modèle de processus comportant de nombreux variants bien définis dans la plupart de ses parties (figure 48, n° ❶ et ❷), ce qui correspond à l'échelon le plus élevé de la graduation de la variabilité.

5.8.2 Granularabilité

La granularabilité est la capacité d'un modèle de processus à offrir différentes granularités, autrement dit différents niveaux de détails dans les éléments qui le constituent.

Comme nous l'avons mentionné dans les sections précédentes, une stratégie doit être concrétisée en activités ❸, ce qui offre une première granularité. Par ailleurs, les activités sont des structures composites ❹, qui permettent de définir autant de niveaux de détails que souhaité. Il en va de même pour les artefacts ❺.

M2Flex autorise donc la création d'autant de niveaux de détails que souhaité, aussi bien dans les activités que dans les artefacts (figure 48, n° ❸, ❹ et ❺), ce qui est la plus haute granularabilité possible selon notre graduation.

5.8.3 Stratégies d'apprentissage

Les stratégies d'apprentissage sont les différentes modalités d'apprentissage incluses dans un modèle de processus. Nous avons vu que M2Flex permet de reformuler une activité dans un langage adapté à un concepteur ou un développeur novice grâce à l'association de reformulation (« *rephrases* », ❻). De plus, une activité et un artefact peuvent avoir un tutoriel (attribut « *tutorialURI* » ❼) et une documentation (attribut « *documentationURI* » ❽), qui permettent de faciliter la prise en main du modèle de processus. La décomposition d'une activité en activités plus simples ❹ est aussi une façon d'aider un concepteur ou un développeur novice à

découvrir ce qu'il doit faire pour réaliser l'activité. Enfin, la structure composite des artefacts ❺, lorsqu'elle est mise en œuvre, peut permettre de mieux comprendre ce que le concepteur ou le développeur doit produire.

Pour éclairer comment ces différentes structures peuvent être combinées, nous proposons l'exemple de la création d'une base de données, comme sur la figure 49 : au plus haut niveau, une activité propose la création d'une base de données ⓐ, à partir d'entretiens menés avec les utilisateurs finaux. Cette activité est décomposée en une séquence d'activités plus simples ⓑ, comportant une modélisation entités - associations, suivie de la transformation de ce modèle en modèle relationnel. Le concepteur du modèle de processus estimant que la création d'un modèle entités - associations n'est pas triviale, il propose une reformulation de cette activité avec un vocabulaire plus simple. La séquence de reformulation comporte alors des activités « identifier les concepts, les objets » ⓒ dans les entretiens menés avec les utilisateurs, puis « identifier les relations entre ces concepts » ⓓ, etc.

M2Flex, à travers les différents niveaux de granularité (figure 48, n° ❸, ❹ et ❺), les reformulations et les documentations associées aux éléments (figure 48, n° ❻, ❼ et ❽), offre trois stratégies d'apprentissage, ce qui correspond à la seconde plus haute valeur de la graduation de cette forme de flexibilité.

5.8.4 Distensibilité

La distensibilité est la capacité d'un modèle de processus à être étendu ou réduit pendant son exécution.

Il ne s'agit donc pas d'une qualité intrinsèque du modèle de processus exprimable avec des classes et des associations : ce sont en réalité les contraintes de validité et de cohérence qui font que le modèle de processus reste valide après modification, et l'outillage qui permet de mettre en œuvre les modifications et de vérifier les contraintes.

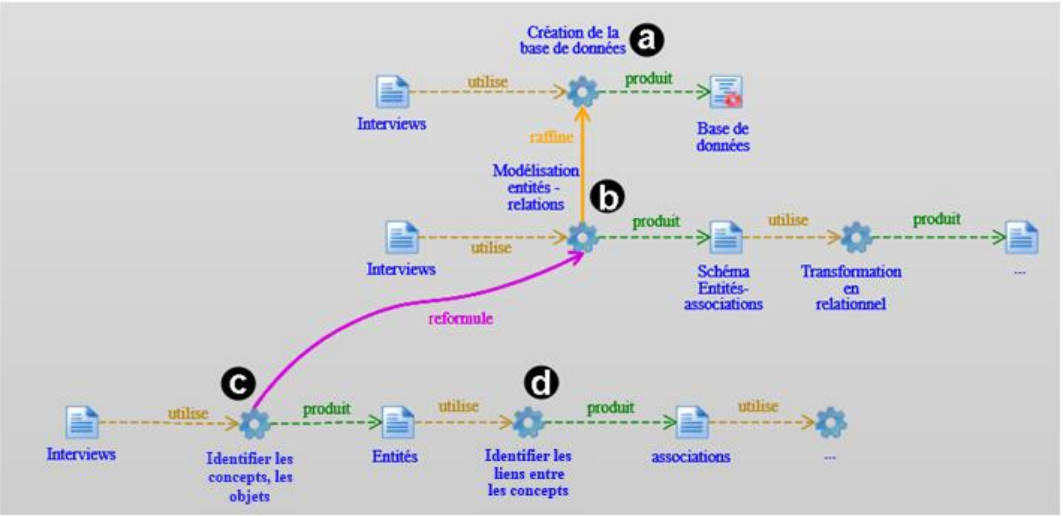

Figure 49 - Exemple de stratégies d'apprentissage

Nous avons vu les contraintes définies pour M2Flex dans la section 5.7. Au niveau conceptuel, la modification du modèle de processus repose sur des procédures, un « méta-processus ». Nous avons défini huit procédures, qui concernent l'ajout et la suppression de buts, de stratégies, d'artefacts et d'activités, et qui sont exprimées sous des formes algorithmiques (elles doivent être traduites en code dans l'outillage du modèle de processus). Les procédures permettant la modification du modèle de processus pendant son exécution sont présentées en annexe.

L'ensemble de ces procédures permet, pendant l'exécution, d'étendre, modifier ou contracter le modèle de processus à volonté, dans la seule limite où ces modifications ne rendent pas le modèle de processus incohérent ou incompatible avec l'instance en cours d'exécution. M2Flex, associé à ces procédures, ces limites et contraintes, offre donc la distensibilité maximale.

5.8.5 Réutilisation

La réutilisation mesure la proposition d'éléments de solutions dans un modèle de processus. En définissant que des artefacts puissent être préexistants (fig. 48, n°❾), M2Flex donne à un modèle de processus la possibilité de proposer des éléments de solution. En effet, un artefact préexistant peut, par exemple, être un modèle embryonnaire que l'équipe de développement pourra enrichir et adapter, ou un programme permettant de créer des éléments utiles pour le projet. Ainsi, un modèle de processus peut proposer une structure de profils d'utilisateurs avec la gestion des droits associés ou un éditeur permettant de générer du code à partir d'un diagramme de classes.

Le modèle de processus instance de M2Flex peut donc potentiellement proposer des éléments de solution comme point de départ pour la création de tous les artefacts. Ce métamodèle est donc évalué comme offrant le niveau le plus élevé de la graduation de la réutilisation.

5.8.6 Polymorphisme

Un modèle de processus est dit polymorphique s'il permet à ses utilisateurs de l'aborder selon ses différents constituants, grâce à des présentations multiples.

M2Flex est structuré sur des buts et des stratégies. Ces stratégies sont des agrégations d'activités (fig. 48, n°❸), elles-mêmes associées à des artefacts (fig. 48, n°❿). Par navigation, il est donc possible d'outiller le métamodèle pour présenter en premier ces différents constituants. De plus, les activités, les artefacts et les outils peuvent être associés à une documentation et un tutoriel, ce qui permet d'offrir de l'aide quelle que soit la perspective. M2Flex offre donc le potentiel de présenter trois perspectives, ce qui correspond à l'échelon maximal de la graduation du polymorphisme défini dans Promote. Ce potentiel reste bien sûr à réaliser grâce à l'outillage qui permettra l'incarnation de M2Flex, et que nous verrons dans le chapitre suivant.

5.9 Conclusion

La figure 50 présente la synthèse comparative des évaluations de modèles et métamodèles de processus présentée dans l'état de l'art (voir figure 35 page 93), et complétée avec M2Flex. La figure présente pour chaque approche, de gauche à droite la variabilité (en bleu), la granularabilité (en vert), les stratégies d'apprentissage (en orange), la distensibilité (en gris), la réutilisation (en violet) et le polymorphisme (en pourpre).

Ce diagramme met en évidence (a) que, contrairement aux autres modèles de processus, M2Flex propose toutes les formes de flexibilité et (b) que pour chacune des formes, M2Flex est, pour cinq formes de flexibilité sur six, à la valeur maximale et, pour la dernière forme, largement au dessus des autres propositions.

M2Flex propose une couverture complète des formes de flexibilité avec une portée largement originale, ce qui constitue aussi notre troisième contribution.

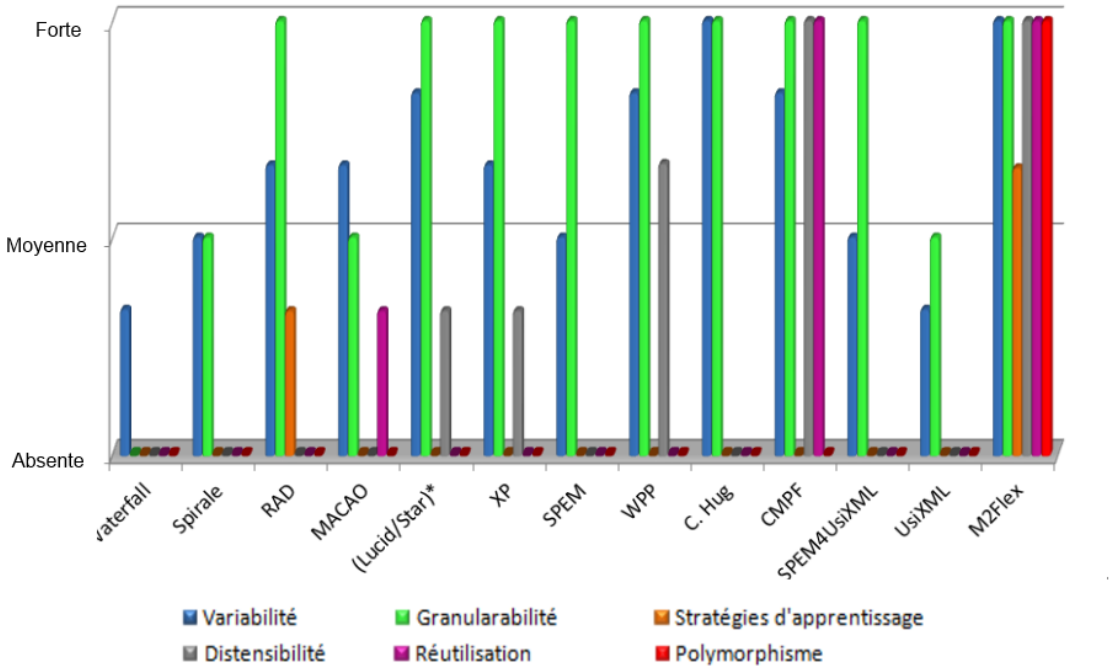

Figure 50 - Comparaison graphiques des flexibilités de 12 modèles et métamodèles de processus

.

Résumé du chapitre 5

Dans ce chapitre, nous avons présenté M2Flex, un métamodèle de processus flexible.

Nous avons montré que M2Flex offre la variabilité à travers la possibilité de définir plusieurs stratégies pour atteindre un but cible à partir d'un but source, ainsi qu'à travers les choix exclusif ou inclusif entre activités. M2Flex offre une granularabilité maximale, grâce à la possibilité de concrétiser une stratégie en activités, activités qui peuvent elles-mêmes être raffinées autant qu'il en est besoin, et grâce à la structure composite des artefacts produits et consommés par les activités. Au-delà de la possibilité de raffiner une activité en activités plus simples, deux autres stratégies d'apprentissages peuvent être incluses dans le modèle de processus qui instancie M2Flex, grâce à la possibilité de reformuler une activité dans un langage plus simple, et d'associer des tutoriels et de la documentation à une activité. Nous avons aussi présenté les procédures qui permettent de modifier dynamiquement le modèle de processus et ses instances, offrant ainsi la distensibilité, ainsi que la possibilité de réutiliser des éléments de solutions : nous avons montré que les artefacts peuvent être définis comme préexistants, et servir de base de travail pour une activité. Enfin, nous avons montré que par navigation, M2Flex permet de représenter le modèle de processus par ses stratégies, par ses activités ou par ses artefacts, ce qui correspond à un polymorphisme potentiel maximal.

6 M2FLEX EN ACTION : CONCEPTION ET EXECUTION DE MODELES DE PROCESSUS FLEXIBLES

Ce chapitre présente l'outillage permettant de créer et d'exécuter des modèles de processus instanciant M2Flex. Cet outillage est composé de deux logiciels : D2Flex, pour la conception et R2Flex pour l'exécution des modèles de processus. La conception de modèle de processus flexibles consiste à composer les buts, les stratégies, les activités etc. qui définissent comment le développement d'un logiciel peut être réalisé. L'exécution du modèle de processus ainsi défini consiste à mettre en œuvre ces stratégies, ces activités etc. Un outil d'exécution de modèle de processus comme R2Flex présente les actions à réaliser, les choix à effectuer, les artefacts à construire aux développeurs et aux concepteurs. Il prend en compte leurs réalisations précédentes pour proposer les étapes suivantes.

Nous avons réalisé deux logiciels, car, même si nous souhaitons offrir la même flexibilité durant l'exécution que durant la conception, les contraintes pesant sur ces deux temps ne sont pas les mêmes : lors de l'exécution, il est nécessaire de gérer l'instanciation du processus en plus de sa conception. Autrement dit, lors de l'exécution il est nécessaire de connaître l'état de chaque activité, de chaque artefact pour pouvoir proposer les étapes suivantes aux ingénieurs.

Dans D2Flex et R2Flex, un modèle de processus est considéré comme un graphe : les classes de M2Flex sont représentées comme des nœuds et les associations comme des arcs. La première section présente une vue d'ensemble des fonctionnalités de ces outils, elle est suivie par une présentation de leurs architectures. La section 3 présente une analyse des formes de flexibilités intégrée dans D2Flex et R2Flex. La dernière section présente une étude de cas menée avec des étudiants en vue d'évaluer la compréhensibilité et l'utilisabilité de ces deux produits.

6.1 Fonctionnalités offertes

D2Flex est dédié à la conception de modèle de processus : il permet d'instancier au niveau du modèle de processus les éléments définis dans M2Flex. R2Flex permettant la modification du modèle de processus durant son exécution, ses fonctionnalités étendent celles de D2Flex pour prendre en compte l'instanciation du modèle de processus. Nous commencerons donc par présenter D2Flex, puis les extensions apportées par R2Flex.

6.1.1 Fonctionnalités de D2Flex

Nous présentons les fonctionnalités de D2Flex à travers la création d'un modèle de processus simple, comportant cinq grandes étapes : l'étude des besoins, la conception générale, la conception détaillée, le codage et les tests. La figure 51 montre le résultat obtenu dans D2Flex.

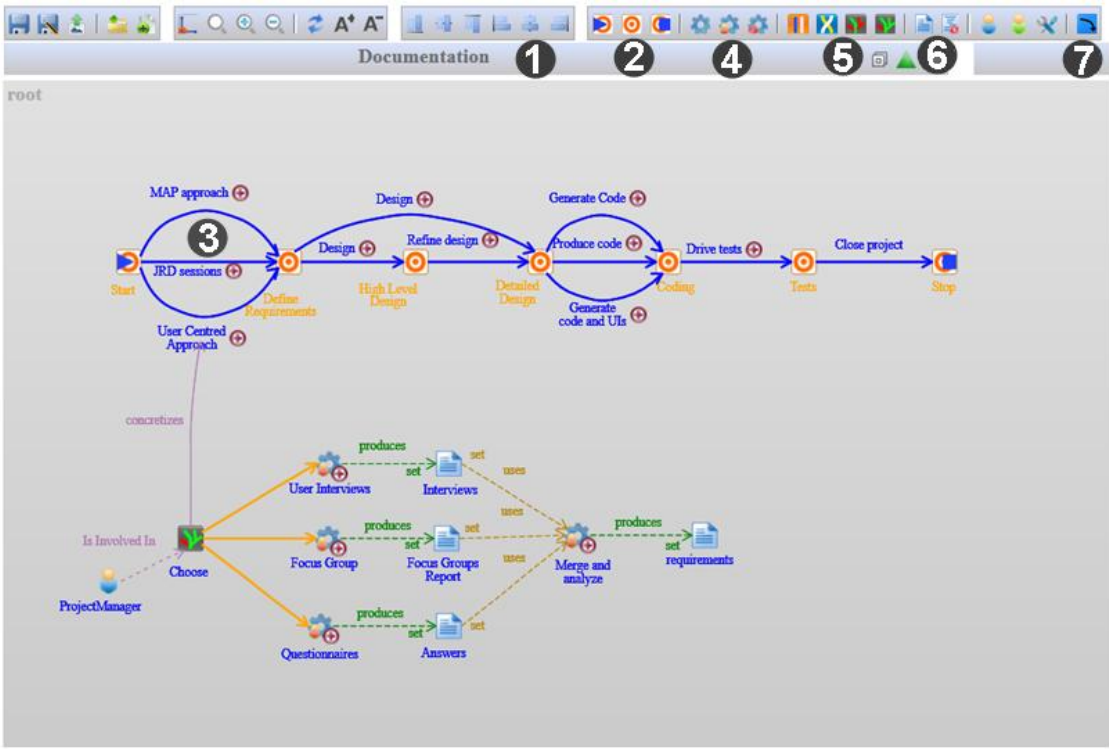

Figure 51 - Copie d'écran de D2Flex

Dans un premier temps, il convient de créer les buts, qui correspondent aux cinq grandes étapes, auxquels on ajoute le but de départ et le but d'arrivée. Cette création se fait grâce à la barre d'outils ❶ qui propose les différents constituants d'un modèle de processus. Les buts sont représentés comme des cibles orange, avec un symbole « démarrer » sur le but de départ et un symbole « arrêter » sur le but final. La seconde étape est la création des stratégies entre les différents buts. Les stratégies sont représentées comme des arcs entre les buts ❷. Comme on le voit sur la figure 51, il peut y avoir plusieurs stratégies entre deux buts, et plusieurs chemins entre différents buts. On peut voir par exemple qu'entre le but de départ et l'étude des besoins, trois stratégies sont proposées : une approche centrée sur l'utilisateur (Norman & Draper 1986), l'approche de la Map (Rolland et al. 1999) et les sessions JRD proposées dans Rapid Application Development (Martin 1991). On peut voir aussi qu'entre l'étude des besoins et la conception détaillée, deux chemins sont possibles ❸, l'un passant par une conception générale et l'autre allant directement à la conception détaillée. La description associée aux stratégies permet de préciser que le chemin direct est adapté pour des petits projets dans un domaine bien connu.

Une fois les stratégies définies, il devient possible de les concrétiser en activités (le système supporte que l'on crée une activité avant de créer les stratégies, mais le modèle est alors – temporairement – dans un état incohérent). Il y a trois types d'activités ❹, correspondant aux trois allocations définies dans M2Flex : les activités peuvent être manuelles (réalisées par un être humain), interactives (réalisées par un être humain en interaction avec le système) ou réalisées par le système seul. Les activités complexes comportant des opérateurs sont représentées avec les icônes incarnant ces opérateurs : on

voit ❺ de gauche à droite les opérateurs de parallélisme, d'entrelacement, de choix exclusif et de choix inclusif. La figure 51 présente plusieurs activités : la stratégie proposant une approchée centrée sur l'utilisateur est ici concrétisée par un choix inclusif entre des entretiens avec les utilisateurs (« *User Interviews* »), des « *focus groups* » et des questionnaires. Le choix est associé à un rôle (« *Project Manager* »), ce qui signifie que seul un acteur incarnant ce rôle pour effectuer le choix. Dans le cas où aucun rôle n'est associé à une activité, comme c'est le cas pour les questionnaires par exemple, il n'y a pas de contrainte sur les rôles : tous les acteurs intervenant dans l'exécution de ce modèle de processus pourront réaliser l'activité.

Les activités produisent des artefacts, qui sont symbolisés par un petit document, accompagné d'un engrenage lorsqu'il s'agit d'artefacts exécutables ❻. On voit par exemple sur la figure que l'activité « Questionnaires » produit des artefacts nommés « Réponses » (« *Answers* ») : le lien « produit » (« *produces* ») entre l'activité et le document signale que ce document est un résultat de l'activité. Plus précisément dans le modèle de processus représenté ici, l'activité produit un ensemble de documents : cette particularité est indiquée par le mot « ensemble » (« *set* ») sur le lien. Les activités consomment aussi des artefacts : c'est le cas ici de l'activité de fusion et d'analyse (« *Merge and analyze* ») des documents produits par les activités précédentes. La figure montre que le lien « *utilise* » (« uses ») porte lui aussi la mention « ensemble » : cela signifie que l'activité utilise l'ensemble des réponses en une fois. Il serait possible de définir que l'activité utilise les documents un par un (« *unit* ») ou par groupes (« *subset* »), permettant ainsi d'indiquer que les réponses seront analysées une par une, ou par paquets. L'activité concernée serait dans ces deux cas répétée itérativement.

La copie d'écran ne présente pas de barre d'outils avec les différents types de liens : pour créer un lien entre deux éléments, l'utilisateur indique qu'il veut créer un lien ❼ puis spécifie, en cliquant dessus, les deux éléments concernés. Le système calcule les liens possibles, et en cas d'ambiguïté, propose les différentes solutions à l'utilisateur. Tous les liens définis dans M2Flex sont implantés dans D2Flex : raffinement, reformulation, utilisation et production d'artefacts, activation, désactivation etc.

Pour éviter de surcharger l'écran, il est possible de définir les différents niveaux de détails dans une autre fenêtre. Ces niveaux de détails peuvent être consultés sur cette autre fenêtre, ou affichés dans une fenêtre superposée à la fenêtre principale, comme le montre la figure 52. Lorsqu'une stratégie ou une activité est raffinée, D2Flex l'indique en ajoutant le symbole « + » sous l'icône de l'élément.

Lors de la création d'un élément, D2Flex affiche une fenêtre permettant de le définir. Pour les activités ou les artefacts, il est ainsi possible de définir un nom, une description, une condition (un texte libre dans cette version), et de téléverser sur le serveur des fichiers considérés comme des tutoriels ou des documentations. Pour les liens, il est aussi possible de définir leurs points de départ et d'arrivée (la zone de l'élément source d'où le lien doit partir, par exemple).

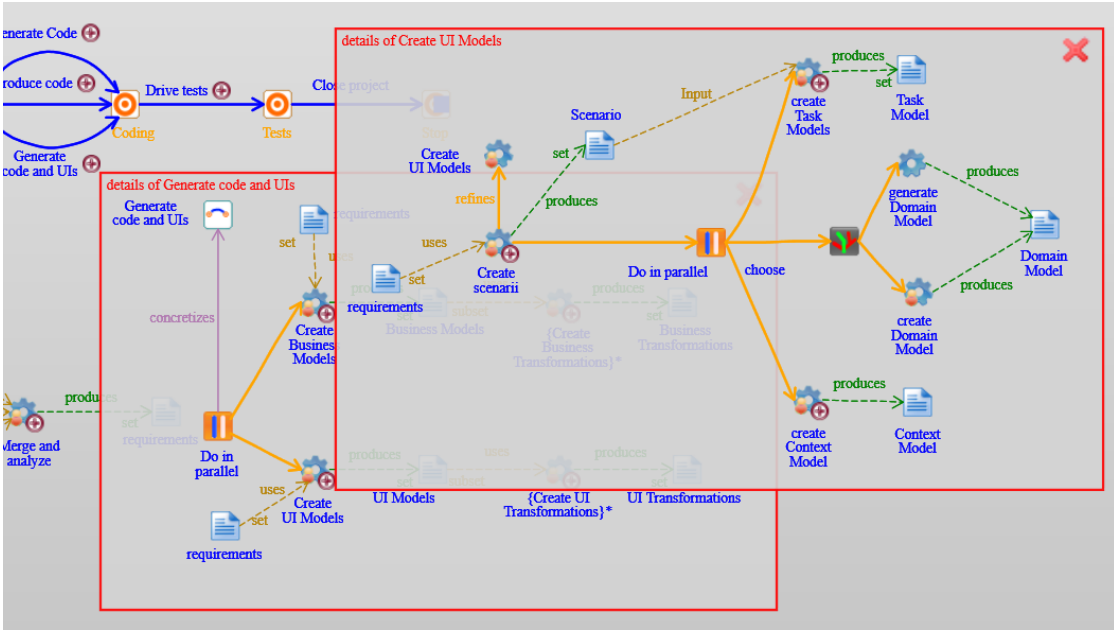

Figure 52 - Niveaux de détails dans D2Flex

Enfin, pour agir sur les éléments du modèle de processus, une barre d'outils contextuelle est affichée à l'aide du clic droit : l'utilisateur peut ainsi demander à modifier la description de l'élément, afficher sa documentation, supprimer l'élément etc.

6.1.2 Fonctionnalités spécifiques à R2Flex

Lorsque l'utilisateur se connecte à R2Flex, il veut exécuter un modèle de processus. R2Flex lui demande de s'identifier et de choisir un « projet » (une instance du modèle de processus) ou d'en créer un nouveau, en choisissant le modèle de processus à instancier. R2Flex parcourt alors le modèle de processus et propose les différents rôles possibles à l'utilisateur, en ajoutant un rôle spécifique, celui de gestionnaire de processus, qui donne l'habilitation pour modifier le processus.

R2Flex propose les mêmes fonctionnalités que D2Flex, avec des contraintes supplémentaires sur les modifications d'éléments déjà instanciés : il est par exemple impossible de supprimer une activité dont l'instance est déclarée réalisée.

La prise en compte de l'exécution du modèle de processus ajoute des fonctionnalités, comme le montre la figure 53 : l'utilisateur dispose d'une nouvelle barre d'outils ❶, qui lui permet de voir ce qu'il peut faire, de définir les membres de l'équipe (s'il a le rôle de chef de projet), de voir son propre profil, etc. Le bouton « démarrer » (en forme de triangle) de cette barre d'outils permet l'affichage d'une fenêtre ❷ qui présente à l'utilisateur toutes les actions qu'il peut réaliser à cet instant : choisir une stratégie, s'assigner à une activité, réaliser une activité etc.

Lorsque l'utilisateur déclare qu'une activité est réalisée, le système lui demande d'indiquer le résultat obtenu. Par exemple, si l'activité produit des artefacts, l'utilisateur est invité à téléverser les fichiers correspondants. Si l'activité est un choix, R2Flex

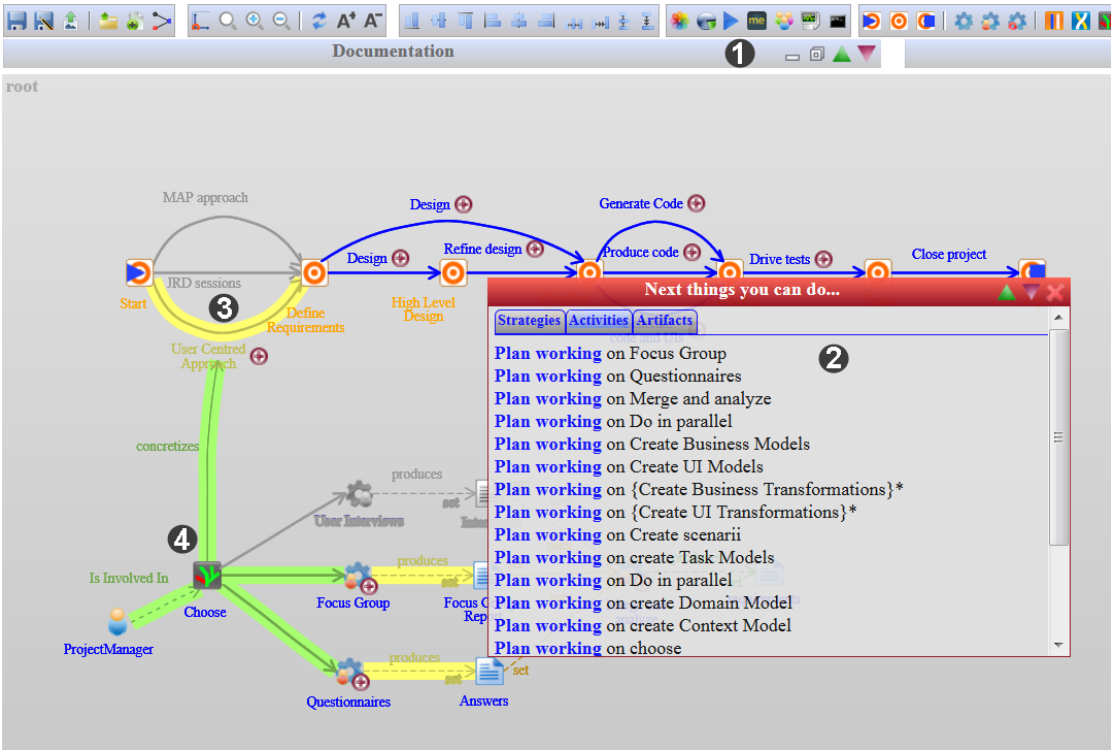

Figure 53 - Copie d'écran de R2Flex

propose les différents choix, par exemple avec des cases à cocher dans le cas des choix multiples, et demande à l'utilisateur d'indiquer les options sélectionnées.

Un jeu de couleurs est utilisé pour indiquer l'état d'avancement : les options rejetées au cours d'un choix (sélection d'une stratégie, choix parmi plusieurs activités) sont grisées, comme on le voit sur la figure 53 pour les stratégies concurrentes de l'approche centrée sur l'utilisateur qui a été choisie. Les éléments en cours de réalisation sont visualisables grâce aux liens surlignés en jaune. Sur la figure 53, on voit ainsi que la stratégie centrée sur l'utilisateur final est en cours ❸. Les liens surlignés en vert indiquent que l'élément source est terminé : le choix entre les différents activités centrées sur l'utilisateur final ayant été faits, les liens qui partent de l'opérateur de choix sont surlignés en vert ❹. L'acteur peut aussi indiquer ses motivations lorsqu'il effectue un choix. Ces motivations peuvent être capitalisées au titre d'exemple pour les futurs projets, même si cette fonctionnalité n'est pas implantée actuellement.

Enfin, D2Flex et R2Flex sont des outils collaboratifs, c'est-à-dire qu'ils permettent à plusieurs utilisateurs de travailler en même temps et à distance sur le même modèle de processus ou sur le même processus. Les modifications effectuées par un utilisateur dans son environnement sont reproduites sur l'interface des autres utilisateurs. R2Flex permet ainsi à ses utilisateurs d'être guidés tout au long du déroulement du processus et d'être informés des actions des autres membres de l'équipe.

La section suivante présente l'architecture des produits.

6.2 Architecture de D2Flex et R2Flex

Le désir de proposer des outils collaboratifs a été l'une des raisons majeures du choix de technologies Internet. Les deux outils sont donc développés en HTML 5 et JavaScript. En raison de nombreuses fonctionnalités communes, les architectures de D2Flex et R2Flex sont très proches l'une de l'autre. Plus précisément, R2Flex est conçu comme une extension de D2Flex : il comporte des modules supplémentaires permettant de gérer les instances du modèle de processus.

La figure 54 présente l'architecture des deux produits. Il y a deux contrôleurs de dialogue, l'un pour la conception du modèle de processus et l'autre pour la gestion des instances durant l'exécution du processus. Le contrôleur de dialogue de D2Flex utilise une classe ModeleDeProcessus qui contient les nœuds et les arcs du modèle de processus, ainsi que les classes spécialisées représentant les activités, les opérateurs, les artefacts et les diverses relations entre ces éléments.

Le contrôleur de dialogue de R2Flex gère les fonctionnalités lors de l'exécution du processus. Il instancie un projet, qui contient le modèle de processus et les instances des nœuds et des arcs. Lorsqu'un utilisateur demande à créer un nouveau projet, le modèle de processus choisi est parcouru de façon à créer une instance pour chacun de ses nœuds. Les associations sont remplacées par des attributs dans les nœuds. Ainsi, une instance d'activité contient deux collections d'instances d'artefacts : l'une pour les artefacts en entrée et l'autre pour les artefacts en sortie. Chaque élément instancié comporte un lien

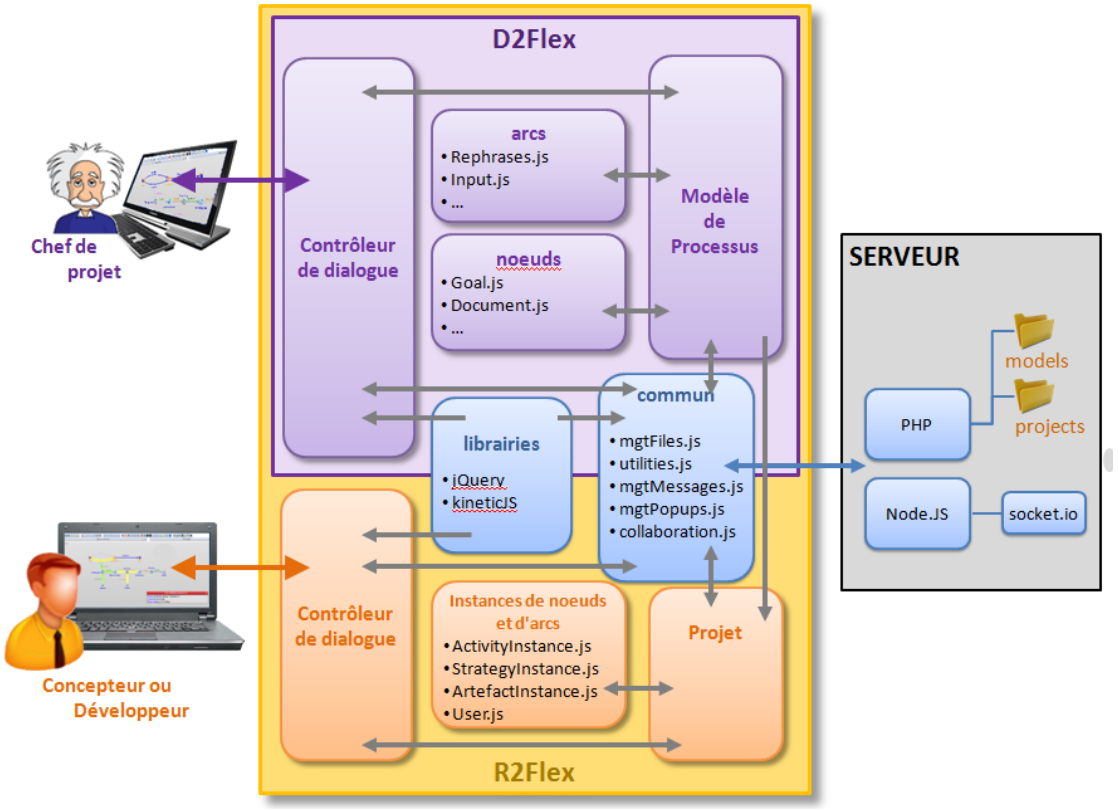

Figure 54 - Architecture de D2Flex / R2Flex

vers l'élément conceptuel qu'il représente, ce qui permet de refléter dynamiquement les changements que l'utilisateur provoque soit dans l'instance soit dans le modèle.

Les artefacts sont instanciés d'une façon particulière. En effet, lors de la conception du modèle de processus, il est possible d'indiquer qu'une activité produit soit un artefact individuel, soit un ensemble d'artefacts. Par exemple, une activité de mise en œuvre de questionnaires produit un ensemble de réponses. Une activité peut aussi accepter en entrée un artefact individuel (une étude des besoins), un ensemble d'artefacts (les réponses aux questionnaires) ou un sous-ensemble d'artefact (trois cas d'utilisation parmi tous ceux définis), sous-ensemble dont le traitement peut alors être itératif. Lors de l'exécution, il n'est pas possible de connaître la cardinalité des ensembles avant que l'utilisateur de R2Flex ne l'indique concrètement. R2Flex contient une classe définissant un ensemble d'artefacts qui est instanciée lors de l'analyse du modèle de processus, les artefacts individuels étant instanciés lorsque l'activité qui les produit est marquée comme terminée.

Les contrôleurs de dialogue, le modèle de processus et le projet s'appuient sur des modules communs (dans le groupe nommé commun) pour gérer les fonctionnalités partagées, comme les téléchargements (par exemple, pour les artefacts préexistants), la gestion de la persistance, l'affichage des messages ou la collaboration. Les modules de gestion de fichiers communiquent avec le serveur à l'aide de requêtes Ajax activant des programmes PHP. La collaboration entre les différents dispositifs clients est assurée par Node.js pour faire transiter des instructions JavaScript. Ainsi, lorsque sur un poste un utilisateur décide de supprimer l'activité n°214 (son identifiant généré automatiquement), le système génère une instruction « process.deleteNode(214) », qui est exécutée localement et transmise aux autres postes, qui l'exécutent aussi. L'activité 214 est alors supprimée de tous les postes. L'intérêt de Node.js dans cette architecture est que la multiplicité des protocoles supportés permet d'assurer la communication quel que soit le navigateur. Cette organisation permet par ailleurs de gérer l'annulation des actions. Lorsque le système génère la commande de suppression de l'activité 214, il génère aussi la commande permettant de faire l'opération inverse et la stocke dans une pile. Si l'utilisateur demande à annuler son action, la dernière instruction d'annulation stockée dans la pile est exécutée et transmise aux autres utilisateurs. L'activité 214 est alors « ressuscitée » sur tous les postes.

6.3 La flexibilité dans D2Flex et R2Flex

6.3.1.1 Variabilité

Différentes stratégies peuvent relier deux buts : D2Flex permet de définir autant de stratégies que souhaité entre un but source et un but cible, dans la limite où deux contraintes sont respectées : (a) il doit y avoir un (et un seul) but de départ et un (et un seul) but de fin et (b) il doit y avoir au moins un chemin reliant le but de départ et le but d'arrivée. Lorsque plusieurs stratégies ont le même but source, seule l'une d'entre elles pourra être sélectionnée lors de la mise en œuvre du processus. Il en va de même pour des

stratégies qui auraient le même but cible. En d'autres termes, lors de l'exécution, il faudra choisir une (et une seule) façon d'atteindre un but et une (et une seule) façon d'en repartir.

Par ailleurs, il est possible de définir des choix entre différentes activités. La suite logicielle offre les deux opérateurs de choix de M2Flex : choix inclusif (plusieurs activités parmi les options proposées peuvent être choisies) et choix exclusif (une seule des options peut être choisie).

Les deux formes de variabilité définies par M2Flex sont implantées.

6.3.1.2 Granularabilité

D2Flex et R2Flex offrent autant de granularités que souhaité. Au plus haut niveau, les stratégies sont concrétisées en activités. A un niveau plus fin, les activités peuvent être raffinées en activités plus simples, comme nous l'avons vu sur la figure 52.

6.3.1.3 Stratégies d'apprentissage

D2Flex et R2Flex proposent les stratégies d'apprentissage définies dans M2Flex : la reformulation des activités, la possibilité de documenter un élément et la possibilité d'associer un tutoriel à un élément (voir section 5.8.3). La figure 55 montre par exemple que l'activité « *Analyze entities and relationships* » (analyser les entités et les relations) ❶ est reformulée, grâce au lien « *rephrases* », en « *Identify concepts and links between these concepts* » (identifier les concepts et les liens entre ces concepts), avec un niveau de vocabulaire plus simple, destiné à un concepteur ou un développeur novice.

On peut aussi voir sur la figure qu'un menu contextuel donne accès à trois types d'aide ❹ :

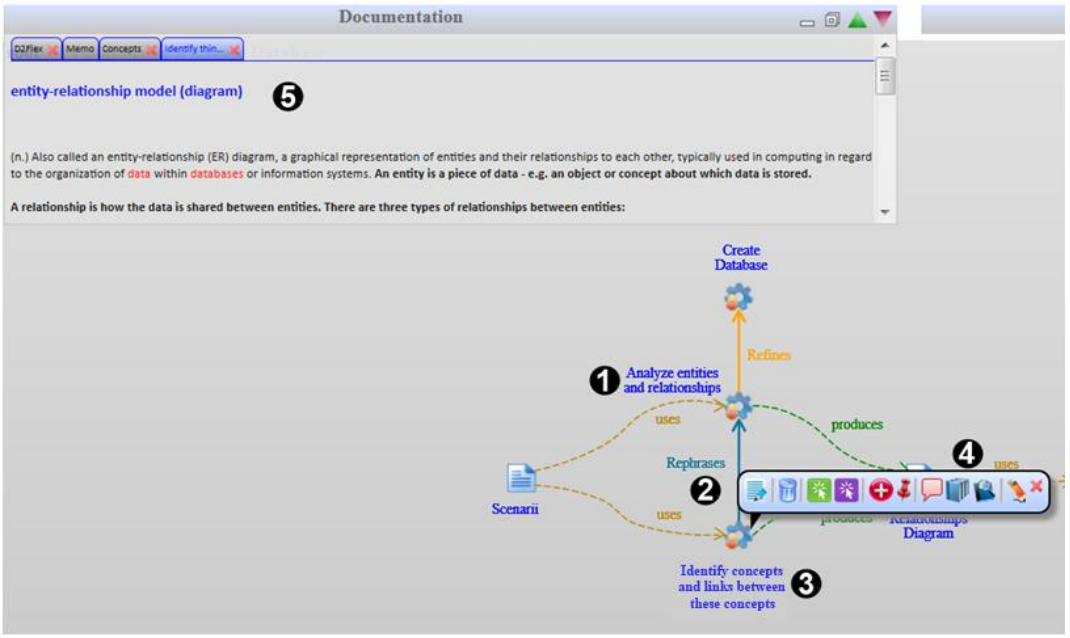

Figure 55 - Stratégies d'apprentissage dans D2Flex

- une « *aide rapide* », calculée par le système, qui donne des indications sur un élément : sa classe (« activité manuelle », « activité interactionnelle »,…), la possibilité de le raffiner, l'existence de niveaux de détails, les différentes actions possibles soit à l'aide du menu contextuel soit par des combinaisons de touches soit par des actions avec la souris…),
- une documentation, qui décrit par exemple le travail à effectuer,
- un tutoriel, qui montre un exemple de réalisation.

Ces différentes aides sont affichées dans l'espace de documentation ❺, sous différents onglets.

6.3.1.4 Distensibilité

Nous avons vu dans la présentation des fonctionnalités de R2Flex qu'il offre les mêmes fonctionnalités que D2Flex en termes de modifications du modèle de processus : on peut ajouter, supprimer ou modifier les éléments du modèle comme dans D2Flex. R2Flex se préoccupe cependant de contraintes supplémentaires : le modèle étant en cours d'exécution, les éléments qui le composent sont instanciés. Par exemple, des stratégies sont sélectionnées, des activités sont en cours de réalisation et d'autres terminées, certains artefacts sont au statut de brouillon et d'autres sont validés. R2Flex doit maintenir la cohérence entre le modèle de processus et son instance. Il limite donc les modifications de façon à ne pas arriver dans une situation incohérente. Par exemple, il n'est pas possible de supprimer une activité du modèle alors qu'elle est au statut « réalisée » dans l'instance.

La figure 56 montre la réaction du système lors de modifications pendant l'exécution : l'utilisateur a sélectionné la stratégie « Approche centrée utilisateur » ❶ et a indiqué son choix dans l'opérateur ❷ : il a sélectionné « User Interview » ❸ et « Focus Group » ❹ et rejeté la troisième option (le lien ❺ est grisé). Dans le menu contextuel sur « User

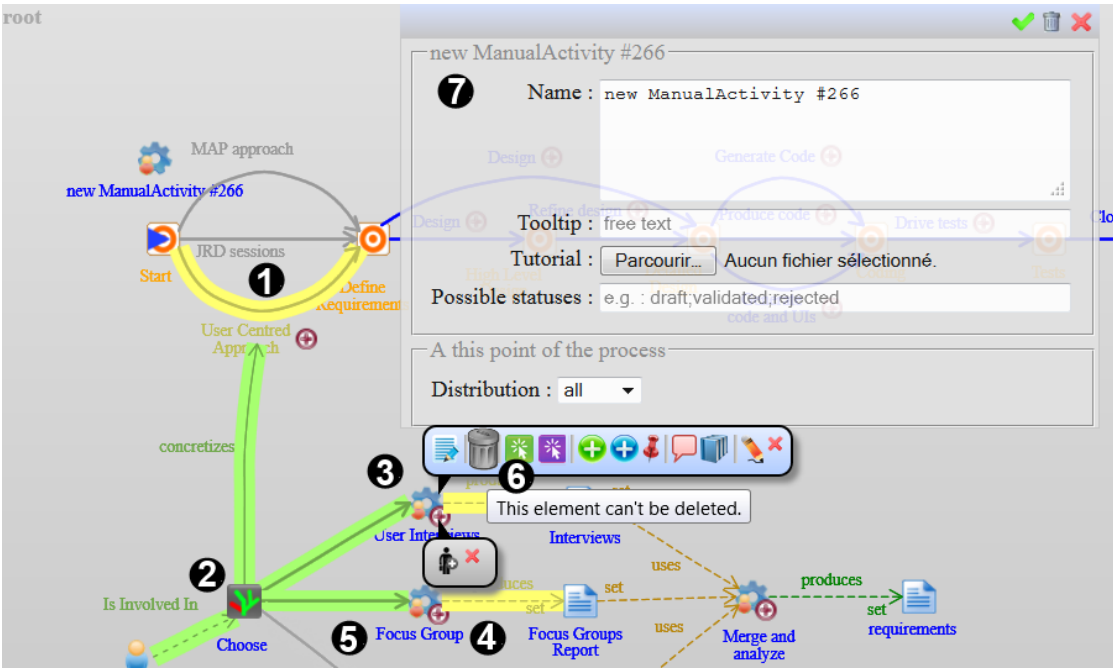

Figure 56 - Distensibilité dans R2Flex

Interview », le bouton « suppression » ❻ (représenté par une corbeille) est grisé et un message indique qu'il n'est pas possible de supprimer l'élément. L'instance de cette activité a en effet été marquée comme sélectionnée, ce qui interdit sa suppression. Dans le même temps, le système autorise la création d'une activité manuelle et affiche la fenêtre ❼ permettant de la définir.

6.3.1.5 Réutilisation

La suite logicielle propose trois modes de réutilisation.

Lors de la conception du modèle de processus, un artefact peut être déclaré préexistant. Le système demande alors de l'associer avec un fichier, qu'il peut ensuite proposer lorsqu'un utilisateur commence à travailler sur une activité qui utilise ce fichier.

Lors de l'exécution du modèle de processus, il est aussi possible de téléverser le ou les fichiers correspondant aux artefacts produits par une activité, comme le montre la figure 57. Ces fichiers pourront dès lors être réutilisés ultérieurement comme artefacts préexistants.

Enfin, les outils permettent d'utiliser des activités « système ». Ces activités permettent de décrire des programmes qui peuvent être exécutés automatiquement au cours du processus. Par exemple, il serait imaginable de générer des profils d'utilisateurs finaux à partir de services d'annuaire. Lors de la conception du modèle de processus, la définition d'une telle activité comporte la définition de ses paramètres éventuels (nom et type de donnée). Dans R2Flex, lorsque le système analyse que cette activité peut être démarrée, il peut soit demander les valeurs des paramètres et lancer l'exécution de l'activité système, soit permettre de télécharger l'application correspondante. Les activités système peuvent alors produire des artefacts à partir d'autres informations, ce qui permet de réutiliser simplement ces dernières. Dans l'état actuel de notre développement, il est possible de définir les activités système, mais leur exécution n'a pas encore été implantée.

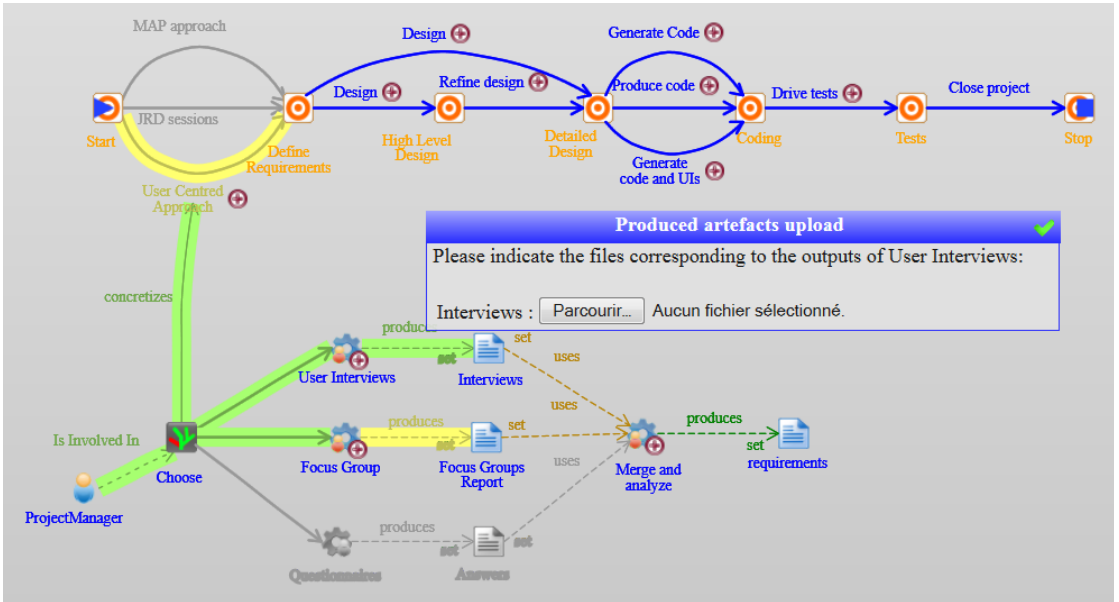

Figure 57 - Téléversement d'artefacts produits par des activités dans R2Flex

6.3.1.6 Polymorphisme

R2Flex peut montrer à l'utilisateur les prochaines actions qu'il peut réaliser. Un bouton « Montrer ce que je peux faire ensuite » (avec un icône « *Play* ») permet d'afficher la liste de ces actions. Cette liste est présentée sous la forme d'une fenêtre comportant trois onglets (voir figure 58). Le premier onglet est centré sur les stratégies, le second sur les activités et le troisième sur les artefacts. Dans chacune de ces listes, R2Flex indique au concepteur ou au développeur les actions qu'il peut réaliser.

La liste des artefacts est générée à partir de la liste des activités sur lesquelles le concepteur ou le développeur peut travailler : il s'agit des artefacts produits par ces activités. Le système interprète la sélection d'une action dans cet onglet afin de la traduire en action sur les activités. Cette traduction en termes d'activités est transparente pour l'utilisateur, qui a le sentiment de travailler sur les artefacts.

Pour que cette impression corresponde pleinement à une réalité, il faudrait que le graphe représentant le modèle de processus soit présenté comme un graphe d'artefacts et non comme un graphe d'activités. R2Flex comporte toutes les structures nécessaires à cette représentation et a le potentiel de présenter le graphe sous l'une ou l'autre forme au choix de l'utilisateur. La représentation du modèle de processus sous forme de graphe d'artefacts n'est cependant pas encore implantée.

Le polymorphisme offert par R2Flex reste donc élémentaire.

D2Flex et R2Flex offrent donc toutes les formes de flexibilité définies dans M2Flex. La section suivante présente une évaluation de ces outils.

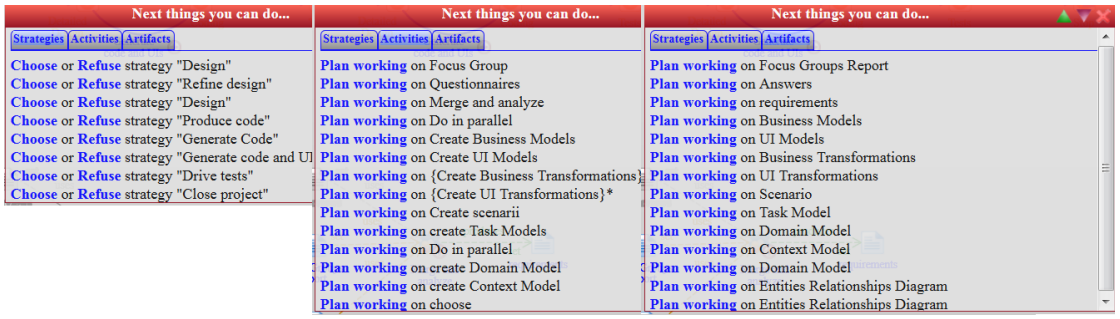

Figure 58 - Listes d'action centrées sur les activités ou les artefacts

6.4 Evaluation de compréhension et d'utilisation de D2Flex

Nous avons mené une expérience en 2013 afin de valider la compréhensibilité et l'utilisabilité de D2Flex, ainsi que le pouvoir génératif de Promote.

Cette expérience s'est déroulée en trois séances :

1. Octobre 2013 : présentation de Promote et de design-methods.net (voir section 3.6). Les participants devaient ensuite étudier un modèle de processus et réaliser sa classification ;
2. Novembre 2013 : présentation et utilisation de D2Flex. La présentation se faisait sous la forme d'un exercice : nous avons proposé aux participants d'étudier un modèle de processus de conception d'IHM qu'ils devaient appliquer à la conception d'une interface pour la gestion d'un commerce. L'énoncé leur suggérait d'étudier la mise en œuvre d'une stratégie centrée sur les utilisateurs, avec en particulier la mise en place de questionnaires. Cette séance mettait en œuvre :
    - un entretien individuel d'une durée de 20 minutes, destiné à évaluer la compréhension du modèle de processus présenté. Pendant la première moitié de l'entretien, les participants s'exprimaient librement sur leur compréhension du modèle de processus. La seconde moitié de l'entretien était consacrée à compléter les points abordés. Les participants étaient alors invités à s'exprimer sur les thèmes suivants : les objectifs du modèle de processus, l'intérêt de la visualisation des différents niveaux de détails, l'intérêt des variants proposés, leur estimation de l'aide apportée par le modèle de processus et l'impact sur leurs pratiques
    - un entretien de groupe, décliné en trois temps : une étude personnelle des activités superflues ou manquantes dans le modèle de processus ; un temps de partage où chaque participant expliquait aux autres ses propositions de suppressions ou d'ajout ; un temps d'élaboration en commun, où les participants débattaient de ces propositions ;
    - un questionnaire de compréhension et d'utilisabilité, que nous détaillerons plus loin ;
    - le recueil des actions effectuées par les participants dans leur interaction avec D2Flex.
3. Décembre 2013 : les participants devaient présenter la classification du modèle de processus qu'ils avaient étudié, ainsi que leurs propositions pour augmenter sa flexibilité.

Six étudiants de Master 2 Recherche et un chercheur ont participé à cette expérience. Au total, nous avons recueilli les notes prises pendant les entretiens individuels, le débat en groupe et les présentations de classification, six questionnaires et 3159 interactions avec D2Flex.

6.4.1 Analyse des questionnaires

Le questionnaire que nous avons proposé était divisé en cinq parties :

1. Ressenti et utilisabilité de D2Flex (voir tableau 7)
2. Intérêt et portée de D2Flex

	1= no	2	3	4	5= yes
1 - I would like to use D2Flex frequently	0	3	1	1	1
2 - I think that D2Flex is unnecessarily complex	2	4	0	0	0
3 - I think that this tool is easy to use	0	0	2	2	2
4 - I think I could need the help of a technician for using D2Flex	3	0	1	2	0
5 - I think that the functions of the tool are well integrated to each other	0	0	2	3	1
6 - I think there are too many inconsistencies in D2Flex	3	2	1	0	0
7 - I guess most people can learn how to use D2Flex very quickly	0	2	1	2	1
8 - I feel confident with this tool	1	0	2	1	2
9 - I had to learn lots of things before being able to use this tool	1	0	4	0	1

Tableau 7 - Actions d'accès au système

3. Perception, clarté et utilité des concepts manipulés (buts, stratégies, activités,…)
4. Perception, clarté et utilité des formes de flexibilité
5. Commentaires sur la mise en œuvre des formes de flexibilité lors de l'analyse du modèle de processus

L'analyse des réponses montre les éléments suivants :

- D2Flex est perçu comme simple à utiliser (moyennement simple pour deux participants, assez simple pour deux autres et très simple pour les deux derniers). L'assistance d'un technicien pour l'utilisation est cependant souhaitée par deux participants. Les différentes fonctions sont bien intégrées. Un participant n'a pas du tout confiance dans l'outil (sans qu'il l'explique), les autres se sentent plutôt confiants. L'utilisation de l'outil ne nécessite pas d'apprentissage important : un participant estime qu'il a dû apprendre beaucoup avant de l'utiliser, quatre estiment que l'apprentissage est « moyen » et un estime qu'il n'a pas eu besoin d'apprentissage.
- D2Flex est évalué comme rendant possible l'intégration de la flexibilité lors de la conception de modèles de processus, avec comme le montre le tableau 8, trois répondants tout à fait d'accord avec cette idée et trois autres plutôt d'accord. Les répondants ne pensent pas que l'utilisation de D2Flex augmente le temps de conception d'un modèle de processus (deux répondants « pas du tout d'accord », deux « plutôt pas d'accord » avec une augmentation du temps de conception et un dans une estimation intermédiaire). Enfin, les répondants sont plutôt d'accord pour penser que D2Flex contribue à l'émergence de

	1=no	2	3	4	5=yes
This tool makes flexibility design possible	0	0	0	3	3
This tool increases the time needed for the design	2	2	1	0	0
This tool helps having new ideas	1	0	2	2	1

Tableau 8 - Evaluations de l'apport pour la conception de la flexibilité

nouvelles idées en termes de flexibilité, avec une réponse « pas du tout d'accord », deux réponses qui ne prennent pas position, deux réponses « plutôt d'accord » et une réponse « tout à fait d'accord ».

- Parmi les 21 concepts manipulés par l'outil, les participants en ont utilisé en moyenne 12, en particulier les buts, les activités, les documents, les rôles internes. De façon cohérente, les participants déclarent majoritairement (4 réponses, les autres ne se prononçant pas) ne pas avoir vu les statuts d'activité, qui sont accessibles uniquement dans R2Flex et donc impossibles à voir dans D2Flex, ni l'association « déclare », qui n'existe pas. Ils n'ont pas non plus perçu les statuts des artefacts, pourtant accessibles en affichant la fiche de l'artefact, ni l'opérateur d'entrelacement, qui n'était pas utilisé dans le modèle de processus qu'ils ont étudié. La clarté des définitions de ces concepts se partage en deux groupes très marqués, les répondants estimant le plus souvent qu'elles sont soit « tout à fait claires » soit « pas du tout claires ». Ce résultat est cependant dépendant du fait d'avoir ou non vu les concepts dans l'outil : ceux qui n'ont pas été vus sont jugés pas du tout clairs (comme les statuts d'artefacts) alors que ceux qui ont été le plus perçus sont jugés tout à fait clairs (les documents, les buts, les stratégies,…). Les concepts perçus sont jugés tout à fait utiles (la moyenne des réponses sur ces concepts est de 4,82, soit très proche de la réponse maximale de 5).
- La perception des différentes formes de flexibilité a été variable : cinq répondants sur six ont vu et utilisé la variabilité et la granularabilité, deux seulement ont vu la distensibilité. Ici aussi, les définitions sont jugées plus claires lorsque les participants ont vu les formes de flexibilité, tandis que l'utilité de ces formes de flexibilité est jugée forte (toutes les répondants sauf un sont tout à fait d'accord avec cette proposition, le dernier estimant que toutes les formes de flexibilité sont tout à fait inutiles).
- Ces évaluations sont confirmées par les commentaires sur la mise en œuvre des formes de flexibilité : les participants expriment par exemple qu'ils ont utilisé la variabilité dans les stratégies, qui « *proposent différents chemins* » et offrent « *différentes approches et différents choix* ». Les autres formes de flexibilité sont très peu commentées, seul un répondant s'exprime sur la granularabilité, en estimant que « *le modèle de processus couvre les principales phases tout en les subdivisant en sous-parties* ». Les réponses sont restées très générales, redonnant la définition des formes de flexibilité sans exprimer de réelle position.

6.4.2 Analyse des entretiens

Les entretiens individuels ont permis de recueillir la compréhension du modèle de processus par les étudiants. La figure 59 montre les buts et stratégies du modèle de processus présenté aux étudiants : le premier but est de « créer des scénarios », qui sont ensuite utilisés pour créer « des modèles principaux », lesquels permettent ensuite de générer une IHM abstraite puis une IHM concrète et enfin une IHM finale. Les résultats

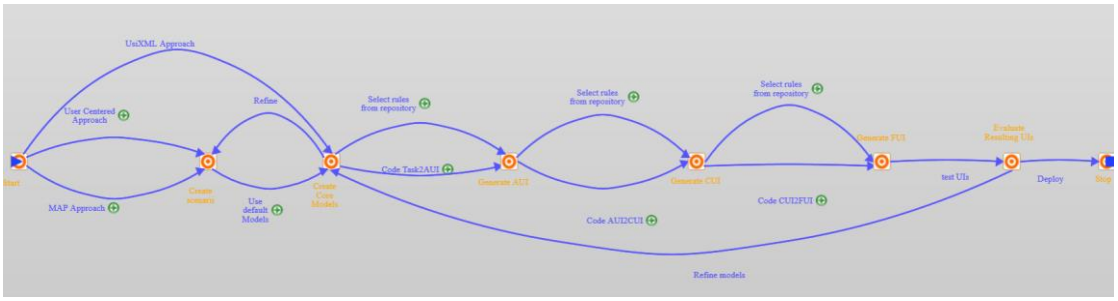

Figure 59 - Vue d'ensemble du modèle de processus proposé pendant l'expérience d'utilisabilité de D2Flex.

sont enfin évalués et le processus se termine. Différentes stratégies permettent de lier les buts. Dans les vues de concrétisation des stratégies, différents choix étaient proposés entre des activités. La stratégie centrée sur l'utilisateur pour la création des scénarios proposait ainsi de mettre en œuvre des entretiens, des focus group ou des questionnaires. En une dizaine de minutes de manipulation sur D2Flex, les étudiants ont tous compris le principe du modèle de processus, l'existence de différents chemins, les propositions de choix et de différents niveaux de détails. Ils ont compris l'utilisation des artefacts comme entrée et sortie des activités. La perception générale du modèle de processus est donc bien établie, même si l'ensemble des répondants estime que les dénominations choisies pour les stratégies ou les activités ne sont pas claires : ils ne voient pas ce qu'est une « AUI », une « CUI » ou une « FUI », ce qui correspond bien au besoin numéro 5 identifié dans le chapitre 2 (disposer de différents niveaux complexité et de différents langages). Trois d'entre eux estiment que la vision du plus haut niveau ne sert à rien sans la vision des concrétisations des stratégies : selon eux, à ce niveau, « *on ne comprend pas ce qu'il faut faire* », ce qui correspond au besoin numéro 4 de granularabilité. Ils estiment que la possibilité de consulter différents niveaux de détails répond à leur besoin, même si deux d'entre eux estiment que l'affichage des détails sur une nouvelle fenêtre fait perdre le lien avec les niveaux parents.

Quatre répondants aimeraient disposer de plus d'aide sur les éléments présentés (légende sur les symboles, documentation détaillée sur les artefacts, exemples d'artefacts, explicitation de l'impact des différents choix…). Le modèle de processus présenté dans l'expérience proposait une description détaillée de chaque activité, mais pas des artefacts. Les remarques sur le besoin d'aide confirment le besoin de stratégies d'apprentissage (besoin numéro 5 des concepteurs et développeurs) et la demande d'exemples confirme le besoin de réutilisation (besoin numéro 3, pouvoir s'inspirer d'éléments de solutions préexistants). L'un des répondants aimerait que l'outil efface les choix non retenus au fur et à mesure de l'avancement du processus, de façon à avoir une vision simplifiée de ce qu'il a réalisé, tout en conservant les différentes possibilités dans ce qui reste à faire. Un autre répondant aimerait avoir la possibilité d'afficher le graphe complet, avec tous les niveaux de détails, sur un seul écran.

Action	Occurrences
Création / modification / suppression des stratégies	**5**
Création de Stratégie	5
Modification d'une stratégie	0
Création / modification / suppression des nœuds	**50**
Création d'un nouveau nœud	8
Création d'une représentation de nœud	21
Modification d'un nœud	14
Suppression d'un nœud	3
Suppression d'une représentation d'un nœud	0
Annul. d'action (ctrl/z après création d'un nœud)	4
Création / modification / suppression des arcs	**61**
Création d'un nouvel arc	3
Ajout d'un nœud à un nouveau lien	32
Annulation d'action (ctrl/z après création d'un arc)	9
Modification d'un arc	13
Suppression d'un arc	3
Echec de la création d'un lien	1
Consultation des niveaux de détails	**210**
Affichage de sous-niveaux sur la fenêtre courante	145
Affichage de sous-niveaux sur une nouvelle fenêtre	47
Utilisation du fil d'Ariane	18
Documentation	**104**
Affichage de documentation	104
Divers	**38**
Echappement de l'action en cours	32
Enregistrement du modèle de processus	6
Création d'un fenêtre de détail pour un nœud	0
Annulation d'action (ctrl/z sur autres actions)	0

Tableau 9 - Actions de manipulation du modèle de processus

6.4.3 Analyse des interactions

Nous avons recueilli 3159 actions de l'utilisateur sur l'interface de D2Flex au cours de la préparation du module et des manipulations effectuées par les étudiants. Ces actions ont été réalisées au cours de 159 sessions de travail, par dix utilisateurs, soit six étudiants, trois enseignants-chercheurs et un doctorant.

Nous les avons réparties deux groupes : les actions de manipulation du modèle de processus (affichage du détail d'un élément, consultation de documentation, modification d'un élément du modèle de processus,…) et les actions de manipulation de la représentation (changement de taille de police, glisser-déplacer de la zone de travail autour du graphe, agrandissement et réduction de la taille du graphe, déplacement d'un élément du modèle de processus…).

Le tableau 9 présente les occurrences des actions de manipulation du modèle de processus. Nous avons regroupé les actions en fonction des catégories d'éléments qu'elles concernent. Nous avons donc :

- 5 créations de stratégies ;
- 50 actions de création, modification ou suppression de nœuds, concernant 11 activités, 8 buts, 3 rôles, 2 outils, 3 opérateurs d'entrelacement. Quatre de ces actions ont été annulées après avoir été effectuées ;
- 61 actions de création, modification ou suppression d'arcs, c'est-à-dire des liens entre les nœuds, comme la relation « input » entre une activité et un artefact. Une création d'arc a échoué car l'utilisateur a demandé la création d'un lien entre deux nœuds entre lesquels il n'existe pas de relation. Neuf actions ont été ensuite annulées.
- 210 actions concernent l'affichage des différents niveaux de détail, dont 145 sur la fenêtre courante et 47 sur une nouvelle fenêtre, ce qui confirme l'utilité des deux représentations.
- 104 affichages de documentation sur les différents éléments du modèle de processus

Ces actions peuvent être rapprochées des remarques exprimées par les participants dans les questionnaires et les entretiens. En particulier, les participants exprimaient avoir besoin de détails, ce qui correspond bien aux 210 affichages de niveaux de détails (soit 45% des actions). De même, les répondants signalaient qu'ils avaient besoin d'aide sur les éléments et on retrouve 104 actions en ce sens, soit 22% de l'ensemble des manipulations faites sur le modèle de processus.

Nous avons utilisé ces remarques pour faire évoluer D2Flex et R2Flex, afin de rendre l'affichage des niveaux de détails et de l'aide plus efficace. Nous avons ainsi ajouté une « aide rapide », qui est générée par le système. Cette aide permet de donner des informations sur la nature de l'objet (c'est une stratégie, un but), quelques éléments sur son sens (« *une stratégie est une approche globale pour atteindre un but* »), et les possibilités d'interaction avec l'élément (« *Des détails sont disponibles en cliquant sur le bouton « plus » à côté du nom* »). Nous avons aussi défini un « tooltip » sur chaque élément qui indique les différentes possibilités pour obtenir de l'aide (« *aide par clic-droit puis boutons aide rapide, documentation ou tutoriel* »). Nous avons aussi amélioré l'affichage des détails d'un élément : la fenêtre de détail qui s'affiche dans la fenêtre courante est maintenant déplaçable et elle passe au premier plan lorsqu'on clique dessus, ce qui permet d'afficher plusieurs fenêtres pour consulter différentes parties du modèle de processus sans que la superposition ne devienne inutilisable.

Les actions de manipulation de la présentation sont présentées dans le tableau 10. Elles permettent de remarquer l'importance du nombre de « zoom » sur le graphe (pour augmenter ou diminuer sa taille), à l'aide des deux boutons « zoom+ » (augmentation de la taille) et « zoom-« (diminution de la taille) ou de la molette de la souris, qui est le principal moyen utilisé (94% des actions de zoom). La possibilité de déplacer le graphe a aussi été largement utilisée. Il faut cependant noter que dans cette version, c'était la seule façon de diminuer la superposition du graphe et des fenêtres de détail. Dans la version actuelle, les fenêtres sont aussi déplaçables, ce qui offre une autre possibilité d'éviter les superpositions gênantes.

Actions	Occurrences
Zoom par la souris	845
Déplacement du graphe	625
Sélection d'un Nœud	213
Déplacement de la sélection au clavier	111
Déplacement d'un nœud	37
Déplacement d'un ensemble de nœuds	37
Bouton zoom out	30
Bouton zoom in	26
Déplacement d'un point intermédiaire dans une stratégie	14
Augmentation de la taille de police	11
Diminution de la taille de police	9
Alignement horizontal de la sélection	6
Alignement de la sélection sur le bas	6
Alignement de la sélection sur la gauche	5
Déplacement du nom d'un arc	4
Déplacement d'un point intermédiaire dans un arc	3
Export du modèle comme image	2
Alignement de la sélection sur la droite	1
Sélection de nœuds	1
Désélection totale	1
Alignement des points intermédiaires d'un arc	0
Alignement des points intermédiaires d'une stratégie	0
Alignement de la sélection sur le haut	0
Alignement vertical de la sélection	0
Distribution horizontale de la sélection	0
Déplacement de la cardinalité source d'un arc	0
Déplacement de la cardinalité cible d'un arc	0
Déplacement du nom d'une stratégie	0
Sélection d'une représentation supplémentaire	0
Sélection d'une représentation	0

Tableau 10 - Actions sur l'interface de D2Flex

Résumé du chapitre 6

Ce chapitre a été consacré à la présentation de D2Flex et R2Flex, les sites Internet qui outillent M2Flex et permettent la conception et l'exécution de modèles de processus. Nous avons présenté l'architecture de ces outils, qui permet de proposer les six formes de flexibilité de Promote à l'utilisateur.

Nous avons présenté une évaluation des deux outils qui a mis en évidence la satisfaction des utilisateurs face à ces outils, et pointé des possibilités d'amélioration que nous avons intégrées dans la dernière version de D2Flex et R2Flex.

Troisième Partie

Troisième Partie

La troisième partie de cette thèse est consacrée aux applications de nos contributions, en particulier, au développement d'interfaces homme-machine plastiques.

Elle comporte les chapitres suivants :

- Le septième chapitre traite de la « flexibilisation » du modèle de processus de UsiXML (UsiXML Consortium 2011), une méthode de développement d'IHM plastiques.
- Le chapitre 8 présente notre environnement support au développement flexible d'IHM plastiques. En particulier, il présente les apports de D2Flex et R2Flex à FlexiLab, notre atelier de développement d'IHM plastiques.
- Le chapitre 9 est consacré à la valorisation de notre approche, c'est-à-dire à son transfert vers la formation et l'industrie.

7 APPLICATION A LA CONCEPTION D'IHM PLASTIQUES

La multiplication des dispositifs - en particulier mobiles - et des usages - le besoin de disposer d'applications capables d'accompagner l'utilisateur aussi bien dans ses déplacements que dans ses utilisations - est plus fort que jamais. Nous nous intéressons à la capacité d'adaptation des applications sous l'angle de l'interaction homme-machine. En Interaction Homme-Machine, la *plasticité* des Interfaces Homme-Machine (IHM) (Thevenin & Coutaz 1999) dénote la capacité d'une IHM à s'*adapter* à son *contexte d'usage* dans le respect des *bonnes propriétés* pour l'utilisateur.

Ces « bonnes propriétés » sont définies selon des points de vue fonctionnel et non fonctionnel : les services attendus par l'humain et leur qualité de service. Ces propriétés sont les exigences de l'utilisateur, c'est-à-dire la *définition du problème* selon une perspective utilisateur. Le *contexte d'usage* a pour vocation de cerner les *contraintes sur la solution*. Il est classiquement défini par un triplet <utilisateur, plateforme, environnement> (Calvary et al. 2001) où :

- L'utilisateur représente l'utilisateur du système interactif. Il peut être décrit par des données générales telles que son âge, sa taille, etc. mais aussi par ses compétences métier, informatiques, etc. Bien entendu, ses compétences peuvent évoluer au fil du temps et militer, en conséquence, pour une adaptation de l'IHM ;
- La plateforme représente les équipements matériels et logiciels disponibles pour l'interaction. Toute évolution peut de même questionner la pertinence d'une adaptation de l'IHM : par exemple, lorsque la batterie du téléphone faiblit, on peut envisager de migrer l'application courante du téléphone à la plateforme la plus proche ;
- L'environnement se réfère à l'espace physique qui accueille l'interaction. Il peut être décrit par ses conditions lumineuses, sonores, sociales, etc.

Cette caractérisation du contexte d'usage montre que la réponse classique consistant à développer autant d'applications que de plateformes ciblées ne peut plus répondre au défi : l'évolutivité des contraintes et la démesure des combinaisons possibles à prendre en compte rendent impossible le développement d'un ensemble de solutions pertinentes. Le défi est alors de calculer dynamiquement la *meilleure solution* pour l'utilisateur, compte tenu du problème et des contraintes sur la solution. Il devient dès lors impératif de permettre au système de raisonner, pendant l'exécution, sur sa propre conception : il ne s'agit plus seulement de percevoir le contexte d'usage et de commuter vers l'IHM préfabriquée la plus appropriée, mais, si nécessaire, d'élaborer une IHM conforme aux besoins de l'utilisateur et compatible avec les contraintes du contexte d'usage courant. C'est pour répondre à ces défis que la notion de *plasticité* a été élaborée.

Doter un système de capacités de raisonnement sur sa propre conception nécessite la mise en œuvre de modèles sur lesquels il peut s'appuyer (Myers et al. 2000). Ce sont ces modèles que nous présentons rapidement dans la section suivante.

7.1 Les modèles pour la plasticité des IHM

Un des premiers résultats des travaux sur la plasticité fut l'identification des niveaux d'abstraction auxquels l'adaptation peut avoir lieu. Ces niveaux s'appuient sur l'architecture générale des « *Model-Based Interface Design Environment* » (MB-IDE) (Szekely 1996). Ils distinguent le niveau domaine (tâches de l'utilisateur et concepts) et les interfaces abstraite, concrète et finale :

- **Les tâches utilisateur et concepts du domaine** décrivent l'interaction Homme-Machine selon une perspective domaine, sans préjuger, si possible, d'une quelconque représentation. Des outils tels que CTT (Paternò et al. 1997) pour les tâches et les diagrammes de classe UML (Object Management Group 2011) pour les concepts sont ici classiques. Un modèle de tâches CTT est une décomposition récursive des tâches en sous-tâches. Les tâches sont reliées entre elles par des opérateurs logiques (choix exclusif, choix inclusif) et/ou temporels (séquence, entrelacement). Elles peuvent être décorées de propriétés telles que la fréquence, l'itération, l'optionalité, etc.
- **L'interface abstraite** structure l'IHM en espaces de dialogue et fixe la navigation entre ces espaces. Un espace de dialogue est un « *lieu d'activité virtuel offrant les éléments nécessaires à la réalisation d'une ou plusieurs tâches* » (Normand 1992).
- **L'interface concrète fait le choix d'interacteurs pour :**
  - Les espaces de dialogue alors réifiés en fenêtres ou canevas dans le cas d'IHMs graphiques ;
  - Le contenu des espaces de dialogue : les tâches élémentaires et les concepts du domaine sont classiquement concrétisés par des boutons radio, cases à cocher, texte, etc. Le contrôle est typiquement assuré par les boutons « Valider » et « Annuler ». La navigation entre espaces est classiquement incarnée par des séparateurs (espace ou trait) ou des objets de navigation (boutons, liens hypertexte, etc.). Le guidage est souvent textuel.
- **L'interface finale est le code correspondant à l'interface concrète.** Elle prend en compte un environnement de programmation et d'exécution disponible sur le dispositif ciblé.

Ces quatre niveaux d'abstraction sont généraux. Aussi ont-ils été repris en plasticité. En particulier, le projet européen CAMELEON en a fait sa colonne vertébrale. Le passage d'un niveau d'abstraction à un autre se fait à l'aide de transformations. Le cadre de référence unifié (Calvary et al. 2001) élaboré au cours du projet CAMELEON et présenté sur la figure 60 définit plusieurs types de transformations : (a) celles qui sont dédiées à la réification, qui est la transformation d'un modèle en un autre modèle plus concret, c'est-à-dire plus proche du code ; (b) celles qui mettent en œuvre l'abstraction, l'opération inverse de la réification, qui permet donc de partir d'un modèle concret pour générer un modèle plus abstrait; et (c) celles qui concernent la translation, c'est-à-dire l'adaptation d'un modèle vers un autre modèle du même niveau et qui permettent par exemple une

adaptation partielle, comme une modification des couleurs des interacteurs pour prendre en compte un changement de luminosité.

Toutes ces transformations sont elles-mêmes considérées comme des modèles, ce qui permet de raisonner aussi à leur niveau et, par exemple, de sélectionner et d'adapter des règles de transformation élémentaires pour les agréger et obtenir la transformation qui permet de passer d'un modèle à un autre.

Pour pouvoir s'exécuter (et pour permettre le raisonnement sur les modèles), les transformations requièrent l'existence d'un langage de définition des modèles. Plusieurs projets internationaux ont œuvré à la définition des métamodèles auxquels les différents modèles doivent être conformes. On peut citer le projet CAMELEON, déjà mentionné, mais aussi Emode (Sottet et al. 2007) et plus récemment, UsiXML (Limbourg et al. 2005), dont les résultats sont en cours de standardisation auprès du W3C.

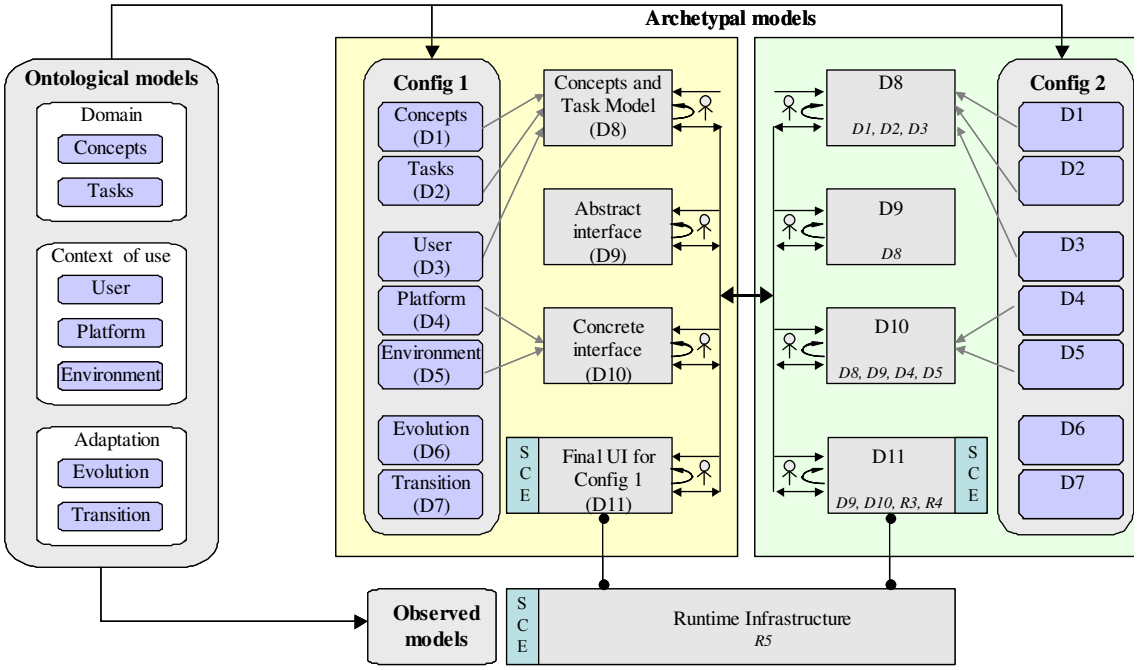

Figure 60 - Le cadre de référence unifié de CAMELEON (Calvary et al. 2001)

7.2 Concevoir une IHM plastique : un exercice complexe

Comme nous venons de le voir, la conception d'une IHM plastique nécessite une approche spécifique. Il faut en effet non plus étudier les informations que l'on veut présenter à l'utilisateur, mais réfléchir en termes d'interaction, c'est-à-dire en termes d'objectifs de l'utilisateur et de contraintes liées aux contextes d'usage potentiels. Il est nécessaire de définir l'ensemble des modèles de l'approche, et donc de percevoir finement la sémantique des métamodèles, et de spécifier les transformations qui permettront l'adaptation au contexte.

(Resnick et al. 2005) notent que la mise en œuvre d'un tel paradigme a un coût d'entrée élevé : le mode de pensée des concepteurs et des développeurs doit changer, les

techniques et technologies mises en œuvre doivent être apprises. Les différentes possibilités offertes par les transformations (réification, abstraction et translation) peuvent aussi déconcerter un concepteur ou un développeur habitué à réaliser des maquettes d'IHM et à les transcrire ensuite en code, et qui peut avoir des difficultés à déterminer d'où il doit partir concrètement dans sa situation propre.

UsiXML offre un guidage pour les concepteurs et les développeurs sous la forme d'une méthodologie (UsiXML Consortium 2011) que nous avons vue dans la section 4.4.5. Cependant, nous avons constaté que le modèle de processus proposé est très peu flexible. Son évaluation a été :

Variabilité :	Aucun variant
Granularabilité :	Une granularité
Stratégies d'apprentissage :	Une stratégie d'apprentissage
Distensibilité :	Aucune procédure
Réutilisation :	Pas d'éléments de solution
Polymorphisme :	Une seule vision sur le processus

Au regard de ce constat de faible adéquation avec les besoins des concepteurs et des développeurs, nous avons voulu « flexibiliser » cette approche en ajoutant les éléments nécessaires pour prendre en compte toutes les dimensions de la flexibilité.

7.3 Flexibilisation de UsiXML

Le modèle de processus de UsiXML (UsiXML Consortium 2011) s'étant avéré peu flexible, plusieurs ajouts étaient nécessaires pour augmenter son adéquation aux attentes des développeurs et concepteurs. En particulier, il convenait d'ajouter des variants, des granularités, des stratégies d'apprentissage, et des possibilités de réutilisation. La distensibilité serait acquise une fois le modèle de processus transcrit dans R2Flex, puisque les fonctionnalités de cet outil permettent la modification dynamique du modèle et de ses instances. De même, R2Flex permettant d'obtenir plusieurs représentations, centrées sur les stratégies, les activités et les artefacts, il n'était pas nécessaire d'incorporer cette dimension dans le modèle de processus lui-même : sa seule transcription dans D2Flex / R2Flex répondait au besoin.

Les sections qui suivent détaillent nos différentes propositions dans ces domaines.

7.3.1 Modélisation comme instance de M2Flex

La figure 61 présente le modèle de processus tel qu'il est défini dans (UsiXML Consortium 2011). Ce modèle de processus se décompose en quatre étapes (« *development steps* ») : Réification du modèle de tâches en modèle d'IHM abstraite[17] ❶, Réification du modèle d'IHM abstraite en modèle d'IHM concrète ❷, Génération du code ❸. Les losanges bleus ❹ représentent des activités (« *development sub-steps* »). Les nombres R1, R2, etc. ❺

[17] Plus précisément, le concepteur ou le développeur sélectionne ou crée les transformations nécessaires aux opérations de réification, abstraction ou translation. Nous avons conservé le vocabulaire du modèle de processus de UsiXML par souci de cohérence avec ce modèle.

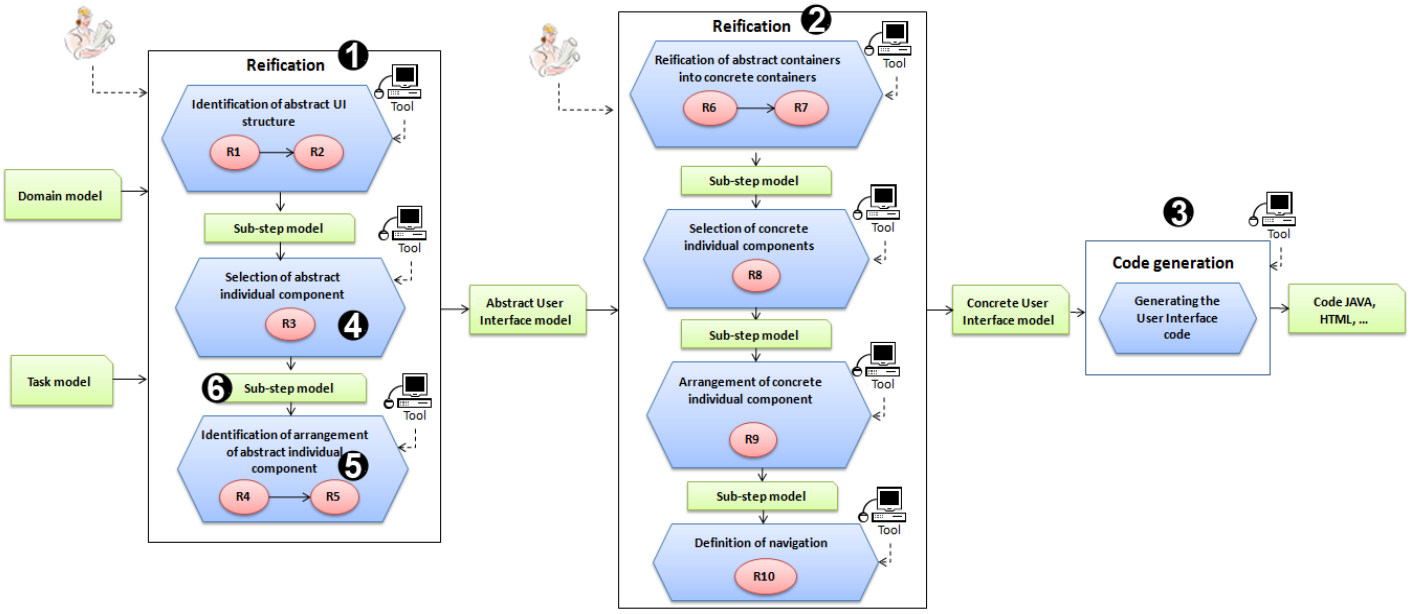

Figure 61 - Le modèle de processus de UsiXML (UsiXML Consortium 2011)

indiqués sur la figure dans des ovales roses représentent des règles à appliquer. Les artefacts intermédiaires sont indiqués par des rectangles verts ❻. Le document du Consortium est complété par une publication (Limbourg & Vanderdonckt 2009) qui propose des chemins alternatifs, explicite la construction du modèle de tâches et du modèle de domaine, et donne des indications complémentaires sur les activités du modèle de processus et sur les différentes règles à appliquer dans les activités.

Les chemins de Q. Limbourg et J. Vanderdonckt sont intégrés dans la transcription du modèle de processus de UsiXML dans D2Flex, à l'exception des chemins qui ne sont que mentionnés et pas concrètement spécifiés (« *retargeting* », « *middle-out development* », « *widespread development* » et « *round-trip development* » - voir section 4.4.5.2 pour plus de détails). Les étapes sont transcrites en buts et en stratégie.

La figure 62 montre la représentation du modèle de processus de UsiXML dans D2Flex. Une première stratégie ❶ correspond à la création des modèles de tâches et de domaine, qui provient de (Limbourg & Vanderdonckt 2009) mais, comme elle n'est pas raffinée en activités dans la publication, nous avons juste créé une activité qui a le même nom que la stratégie. Cette stratégie relie le but de départ et le but « créer les modèles de tâches et de domaine » ❷. Les autres stratégies de ce chemin sont « réifier les tâches et le domaine » en IHM Abstraite ❸, puis « réifier le modèle d'IHM abstraite » en IHM concrète ❹ et enfin produire le code final ❺. Nous avons aussi représenté les autres chemins de développement. Celui de la rétro-ingénierie part du code et génère une IHM concrète ❻, puis remonte jusqu'au modèle de tâches et de domaine par des abstractions successives ❼. Le chemin d'adaptation à un contexte d'usage permet d'adapter à différents niveaux un modèle à un nouveau contexte ❽.

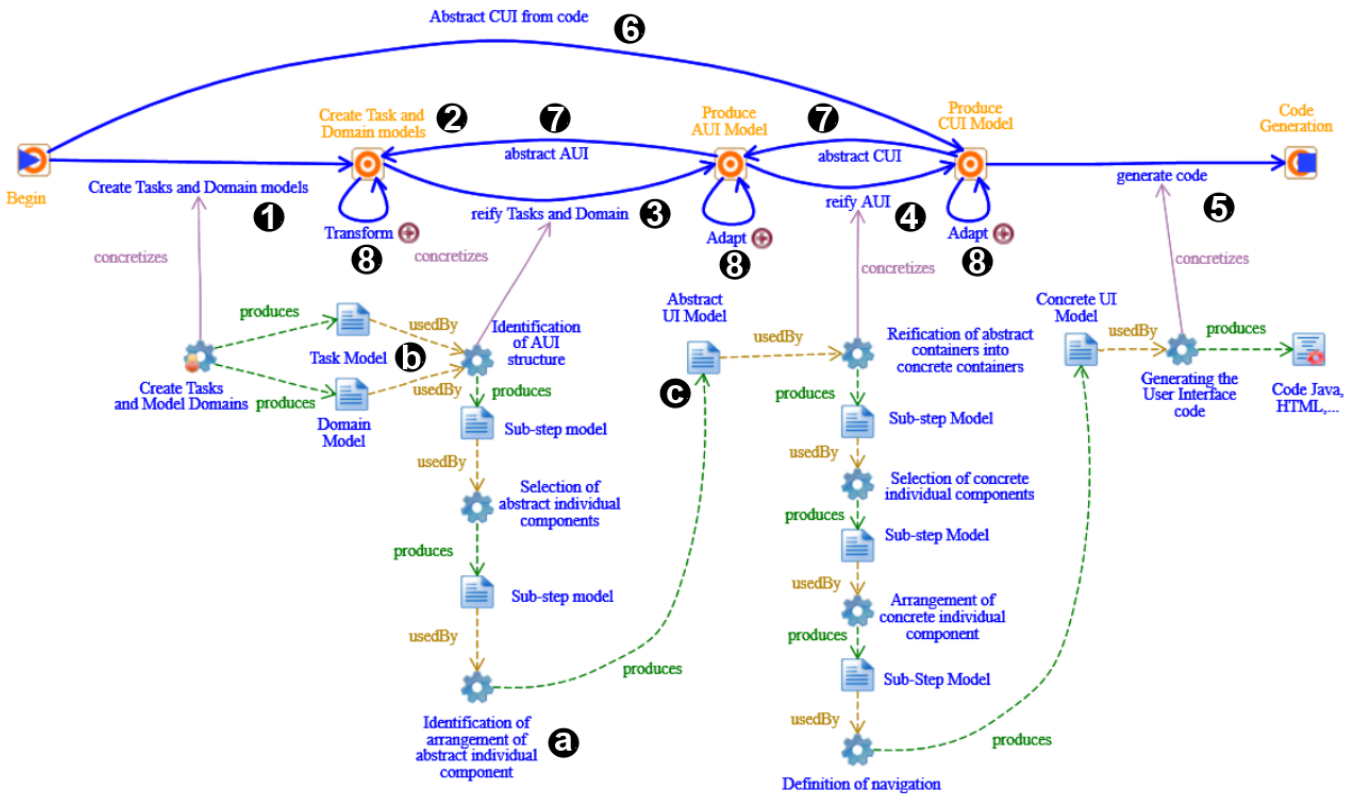

Figure 62 - Le modèle de processus de UsiXML représenté dans D2Flex

Les activités définies dans les étapes du modèle de processus de UsiXML sont transcrites en activités dans D2Flex. On retrouve par exemple l'activité « Identification de l'arrangement du composant abstrait individuel » ⓐ. Dans le modèle de processus original, les modèles (tâches, domaine,…) sont indiqués en entrée des étapes et non des activités : nous les reportons en entrée de la première activité de la stratégie ⓑ. L'artefact indiqué en sortie de l'étape est reporté comme production de la dernière activité ⓒ. Les indications trouvées dans (Limbourg & Vanderdonckt 2009) sont reportées comme documentation des activités. Les informations concernant les modèles, trouvées dans (UsiXML Consortium 2012), sont transcrites en documentation des artefacts.

A l'issue de cette modélisation, le modèle présente plusieurs rigidités et difficultés d'application. Ainsi, les activités permettant la création des modèles de tâches et de domaine ne sont pas définies. Si le modèle comporte plusieurs stratégies représentant les différents chemins de développement, il n'y a pas de variant au niveau des activités : une fois que l'équipe de développement a choisi une stratégie, elle entre dans un processus linéaire. La section suivante montre comment nous avons augmenté la flexibilité de ce modèle de processus.

7.3.2 Augmentation de la flexibilité

7.3.2.1 Augmentation de la variabilité

Notre objectif est ici de proposer des variants supplémentaires, principalement au niveau des activités. Nous avons étudié la possibilité de créer une IHM concrète avec des outils comme SketchiXML (Coyette et al. 2007) ou Balsamiq Mockups[18]. SketchiXML repose sur les métamodèles de la première version de UsiXML, qui ont sensiblement changé lors de la définition de la version 2, que nous flexibilisons ici. Il est néanmoins possible d'utiliser cet outil, de générer des modèles conformes à la version 1 de UsiXML et de les transformer ensuite en modèles conformes à la version 2. Balsamiq Mockups permet de créer des maquettes et de les exporter dans un fichier au format XML. Il est ensuite possible de transformer les maquettes issues de ces outils en modèles d'IHM concrète conformes aux métamodèles de UsiXML. Pour prendre en compte ces possibilités, nous ajoutons une stratégie permettant de générer l'IHM concrète à partir d'une maquette (« *abstract Concrete UI from mockup* »). La distinction entre les différentes possibilités (maquettes, IHM concrète issue de SketchiXML,…) est faite au niveau des activités.

Cette stratégie se raffine au premier niveau, comme nous l'avons vu ci-dessus en un choix, celui d'utiliser soit SketchiXML, soit Balsamiq Mockups. Cette dernière activité se raffine à son tour en une séquence d'activités plus simples : créer les maquettes, les télécharger, et utiliser une transformation nommée balsamiq2CUI pour générer l'IHM concrète. La création des maquettes est à son tour raffinée par d'autres activités : aller sur le site Internet, créer un compte ou se connecter à son compte s'il existe déjà, créer un nouveau projet etc. On peut noter qu'aucune des activités de ce dernier niveau n'est liée à un artefact : l'ensemble des actions se déroulant sur un site Internet, le concepteur d'IHM n'a pas accès aux artefacts intermédiaires. Toutes les activités de ce niveau sont donc liées par un opérateur d'activation. Par ailleurs, l'ajout de ces niveaux de détail a entraîné la création de nouveaux variants : le choix entre SketchXML et Balsamiq Mockups et le choix entre créer un compte ou utiliser un compte préexistant sur le site Internet.

Plusieurs autres éléments peuvent être ajoutés. Par exemple, dans la stratégie de création des modèles de tâches et de domaine, nous avons mentionné que le modèle de processus partait de scénarios utilisateurs. Il existe plusieurs façons de créer ces scénarios, comme une approche d'ingénierie des besoins par identification des buts (Rolland 2007), des sessions d'exploration (« *JRD Sessions* ») comme dans Rapid Application Development (Martin 1991) ou encore les pratiques définies dans les approches centrées sur l'utilisateur (Norman & Draper 1986). Dans cette dernière possibilité, on peut encore proposer différents choix, comme la mise en œuvre de questionnaires et/ou d'entretiens avec les utilisateurs et/ou des « focus groups ». La figure 63 présente ces ajouts dans le modèle de processus.

[18] voir http://balsamiq.com/products/mockups/

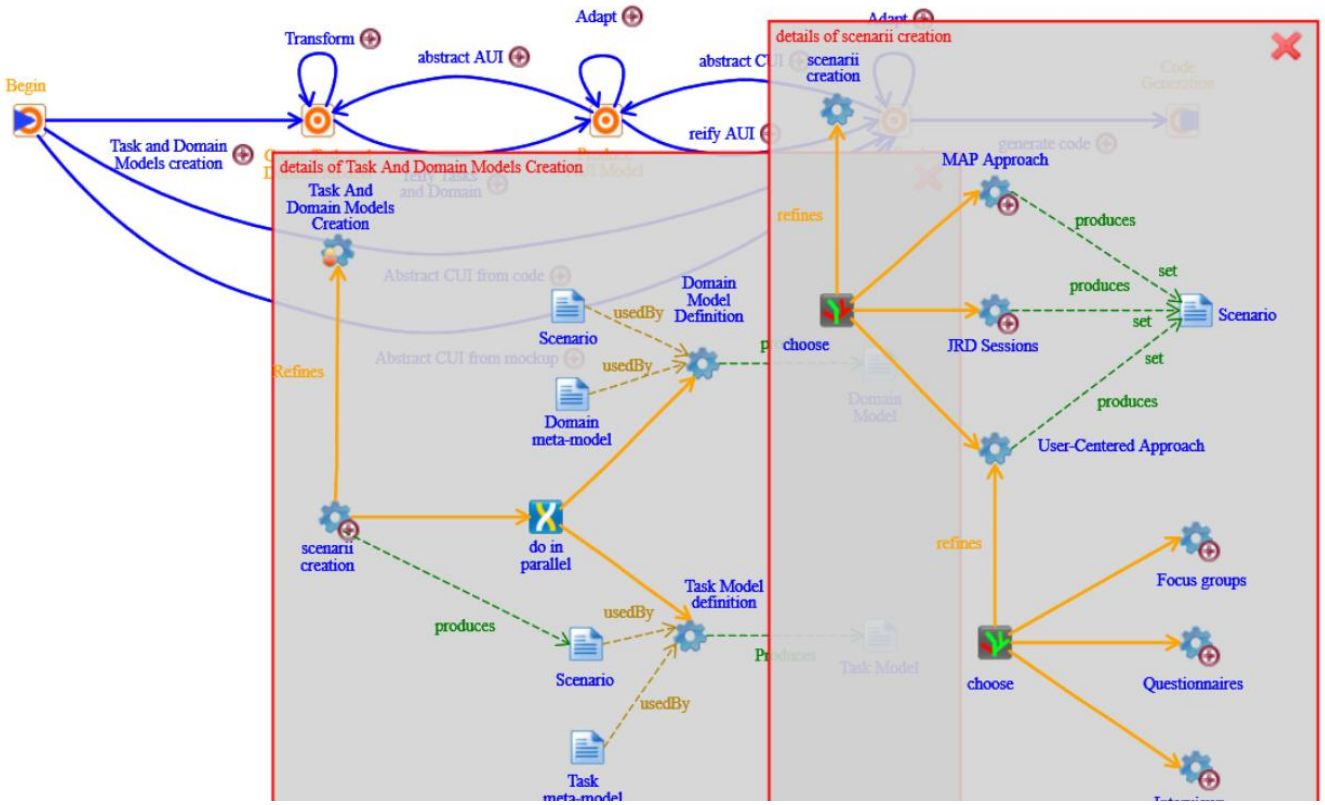

Figure 63 - Le modèle de processus de UsiXML augmenté avec des variants

7.3.2.2 Augmentation de la granularabilité

Nous avons pu voir dans section précédente qu'en ajoutant des variants, nous avons en même temps ajouté des niveaux de raffinement. La figure 63 montre un exemple de stratégie raffinée en activités à leur tour raffinées en activités plus simples.

Nous avons aussi ajouté des raffinements pour toutes les activités proposées. Par exemple, l'activité « Identification de la structure de l'IHM abstraite » (« *Identification of AUI Structure* »), qui fait partie de la stratégie de réification des modèles de tâches et de domaine en IHM Abstraite (voir figure 62, ⓑ) est détaillée sous forme textuelle dans (Limbourg & Vanderdonckt 2009). Nous avons transcrit cette description sous forme d'activités : « *identifier les groupes de tâches fortement liées les unes aux autres* », « *pour chacun de ces groupes, créer un composant abstrait* ». Traduit dans D2Flex, l'activité d'identification produit un artefact complexe qui est un ensemble de groupes de tâches. L'activité de création des composants abstraits itère sur ces groupes. L'activité d'identification des groupes peut à son tour être détaillée, puisque Q. Limbourg et J. Vanderdonckt mentionnent plusieurs approches possibles pour cette identification. La figure 64 montre ces différents niveaux de détails.

7.3.2.3 Augmentation des stratégies d'apprentissage

Nous avons vu en section 4.4.5 que le modèle de processus de UsiXML comporte une stratégies d'apprentissage, correspondant aux explications sur le modèle de processus

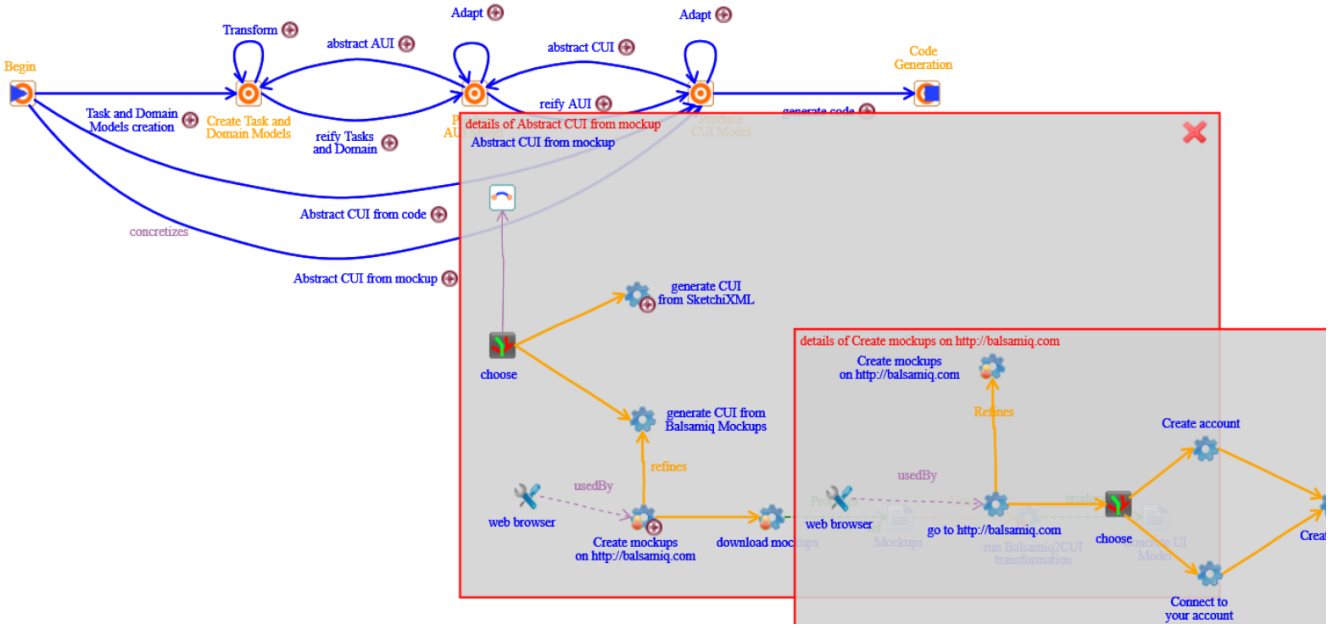

Figure 64 - Le modèle de processus de UsiXML augmenté avec des niveaux de détails supplémentaires

(UsiXML Consortium 2011) (Limbourg & Vanderdonckt 2009) et à la documentation sur les métamodèles (UsiXML Consortium 2012). Nous avons aussi vu, dans la même section, que le Consortium UsiXML a participé à des ateliers de conférences qui auraient pu constituer une seconde stratégie d'apprentissage, focalisée sur l'observation d'expérience, s'ils avaient été inclus dans le modèle de processus. Nous proposons donc d'inclure des références aux pages Internet qui retracent ces ateliers dans la documentation des activités, ce qui permet d'intégrer cette seconde stratégie d'apprentissage basée sur l'observation. Il reste cependant possible d'ajouter des stratégies d'apprentissages ciblées sur les autres formes d'apprentissage de Kolb, c'est-à-dire l'action et le ressenti. Pour cela, nous proposons d'ajouter des tutoriels dans le modèle de processus, contenant des exercices avec leurs corrigés. D2Flex et R2Flex supportant les tutoriels aux formats texte, HTML et PDF, il est possible de proposer un sujet d'exercice et de faire apparaître le corrigé à la demande de l'utilisateur, comme le montre la figure 66 qui présente un exercice proposé par G. Calvary à ses étudiants.

7.3.2.4 Augmentation de la réutilisation

Nous proposons plusieurs possibilités pour offrir des possibilités de réutilisation dans le modèle de processus de UsiXML.

Tout d'abord, il est par exemple possible de générer un modèle de domaine à partir d'une structure de données comme une base de données ou d'une définition de structure de document XML (*Document Type Definition* ou DTD). Les modèles de domaine ainsi obtenus seront sans doute mal adaptés à l'objectif de génération d'IHM, puisqu'ils représenteront la structure des données vues par le noyau fonctionnel et non les concepts manipulés dans l'interface, mais ils peuvent servir de point de départ, et être adaptés ou enrichis par l'équipe de développement. Nous avons ainsi créé une routine, DB2Domain,

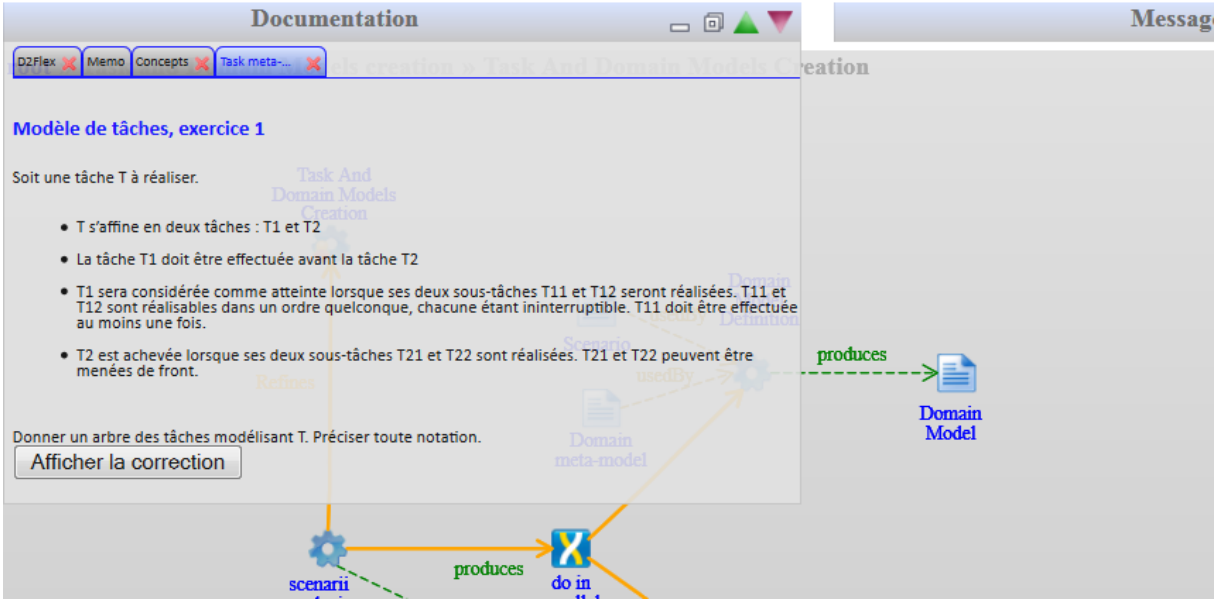

Figure 66 - Exercice sur les modèles de tâches dans R2Flex

que nous avons présentée dans (Céret, Calvary, et al. 2013), et qui analyse la structure d'une base de données relationnelle et produit un modèle de domaine conforme au métamodèle de UsiXML, comme le montre la figure 65.

Le modèle de tâches peut lui aussi être généré, au moins partiellement, par exemple par transformation d'un diagramme de séquences UML. Par ailleurs, un outil comme Compose (Gabillon et al. 2011) permet de générer des modèles de tâches à l'aide d'algorithmes de planification automatisés. Dans ce paradigme, un but est exprimé par l'utilisateur (le concepteur ou le développeur dans notre cas), la situation (les ressources disponibles, les actions possibles, etc.) sont décrites par des prédicats logiques. Ainsi, le but « Trouver un chemin sur une carte » repose sur un accès à Internet, lequel est décrit par le prédicat « internet » qui peut valoir vrai ou faux. Compose génère un modèle de tâches dans un format spécifique, il est donc nécessaire de créer les transformations pour générer le modèle de tâches conforme au métamodèle de UsiXML. Ces possibilités sont

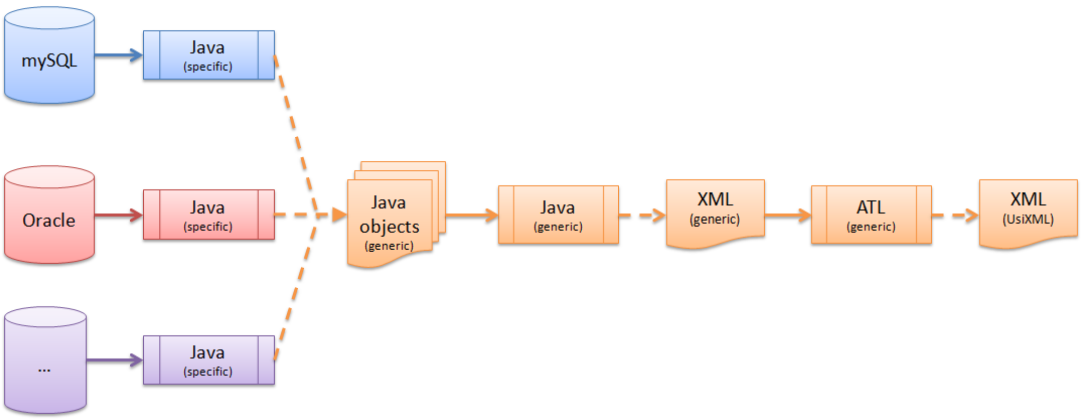

Figure 65 - Schéma de principe de DB2Domain

ajoutées dans la concrétisation de la stratégie de création des modèles de tâches et de domaine.

Par ailleurs, il est aussi possible de proposer de réutiliser des modèles « par défaut », c'est-à-dire des modèles répondant à des problématiques récurrentes. Par exemple, nous proposons un modèle de plateforme et un modèle d'utilisateur élémentaires. Ainsi, le modèle de plateforme décrit un dispositif avec une surface d'affichage, un système installé et des langages de programmation disponibles. Ce modèle de plateforme couvre des besoins simples dans le développement d'interfaces plastiques, mais permet à une équipe d'ingénieurs de disposer d'un point de départ.

Enfin, d'autres éléments peuvent être réutilisés d'un projet à l'autre, comme les transformations. Les transformations d'IHM concrète en code sont toutes réutilisables d'un projet à l'autre dans la mesure où les langages ciblés sont identiques, ainsi qu'une grande proportion des autres transformations, comme la transformation d'une espace de travail dédié à la sélection d'un élément dans un ensemble en un interacteur comme des boutons-radio ou une liste déroulante.

7.3.3 Evaluation de la flexibilité augmentée

Une fois intégré dans D2Flex et R2Flex, le modèle de processus comporte, comme nous l'avons vu, de nombreux variants bien définis. Nous avons ajouté plusieurs niveaux de détails, ainsi qu'une stratégie d'apprentissage. R2Flex apporte nativement la distensibilité, avec des procédures bien définies. Nous avons montré qu'il est possible d'ajouter des éléments de solutions à plusieurs étapes du processus, notamment les transformations qui sont utilisées dans de nombreuses activités. Enfin, les documentations et tutoriels associés aux artefacts permettent de proposer une

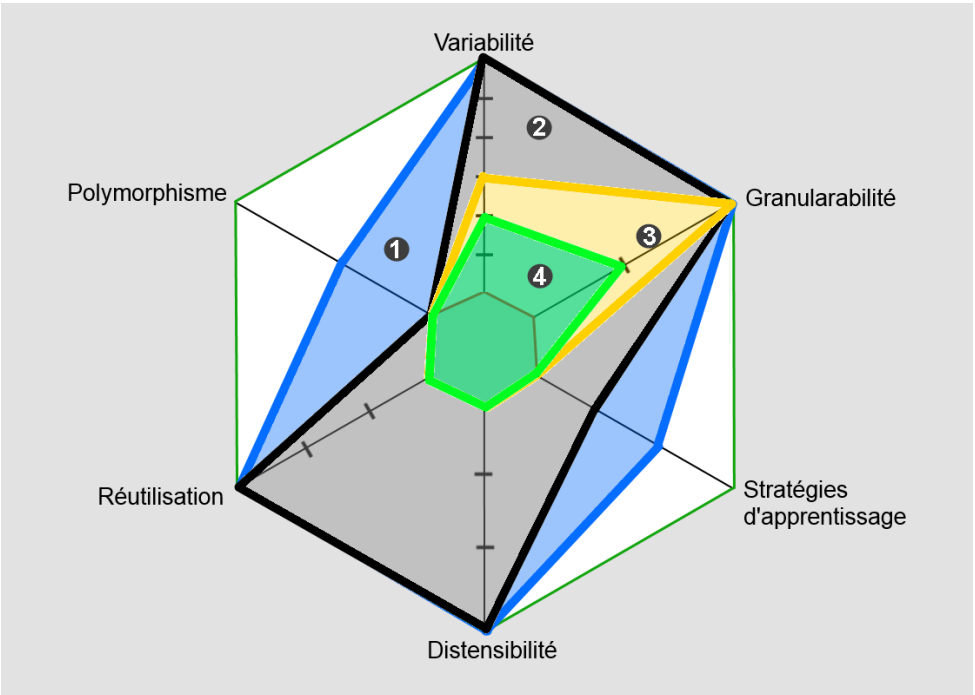

Figure 67 - Flexibilité du modèle de processus UsiXML augmenté

représentation du processus centrée sur ces derniers. L'évaluation de ce modèle augmenté est donc :

Variabilité :	Nombreux variants bien définis
Granularabilité :	Plus de deux granularités
Stratégies d'apprentissage :	Trois stratégies d'apprentissage
Distensibilité :	Procédures bien définies avec validation
Réutilisation :	Nombreux éléments de solution
Polymorphisme :	Deux présentations du processus

La figure 67 représente cette évaluation dans la zone numérotée ❶. Les autres zones de la figure représentent le maximum atteint par les modèles et métamodèles de processus ❷ analysés dans notre état de l'art (voir chapitre 4), le métamodèle de l'approche UsiXML, SPEM4UsiXML ❸ et le modèle de processus UsiXML avant sa flexibilisation ❹. Le modèle de processus UsiXML flexibilisé couvre la plus grande superficie dans la figure, ce qui traduit graphiquement sa flexibilité plus importante.

Résumé du chapitre 7

Dans ce chapitre, nous avons présenté une application de la flexibilité des modèles de processus telle qu'elle est définie par Promote. Nous avons montré comment nous pouvions "flexibiliser" le modèle de processus proposé par le Consortium UsiXML (UsiXML Consortium 2011).

Nous avons présenté l'augmentation des différentes formes de flexibilité, par l'ajout de variants, de niveaux de détails, de stratégies d'apprentissage et d'éléments de solutions.

La flexibilité du modèle de processus résultant a été évaluée et nous avons montré qu'il présente une flexibilité non seulement très supérieure à celle définie initialement dans UsiXML (UsiXML Consortium 2011) mais aussi à celles des modèles de processus que nous avons présentés dans l'état de l'art.

8 FLEXILAB : ATELIER FLEXIBLE DE DEVELOPPEMENT D'IHM PLASTIQUES

Nous avons vu dans le chapitre précédent comment intégrer des formes de flexibilité dans le modèle de processus UsiXML. Mais cette intégration ne peut prendre sens que si elle peut être mise en œuvre concrètement. Par exemple, les embryons de solution ne peuvent constituer un apport que s'ils peuvent être réellement utilisés dans le développement d'IHM plastiques.

Dans ce chapitre, nous présentons FlexiLab, un atelier de conception et d'exécution d'IHM plastiques. FlexiLab permet de générer des IHM adaptées à leur contexte d'usage par transformations successives d'un modèle de tâches (qui décrit les objectifs de l'utilisateur final) et d'un modèle de domaine (qui décrit les concepts manipulés sur l'IHM) en IHM abstraite, puis en IHM concrète et enfin en code. Les IHM ainsi générées sont alors affichées sur le dispositif de l'utilisateur final. Le développement de ces IHM consiste à définir les modèles et les transformations. Nous avons décrit dans le chapitre précédent le modèle de processus flexible qui définit les différentes tâches à accomplir pour développer une IHM plastique. FlexiLab a été conçu pour permettre l'exploitation des formes de flexibilité définies dans ce modèle de processus. En particulier, FlexiLab permet de mettre en œuvre les différents variants et éléments réutilisables que nous avons vus dans les sections 7.3.2.1 et 7.3.2.4.

Pour clarifier les liens entre les différents outils que nous proposons, la prochaine section présente leur utilisation dans un cycle complet allant de l'ingénierie d'un processus de développement à son application pour le développement d'IHM plastiques.

8.1 Positionnement de FlexiLab par rapport à D2Flex et R2Flex

La figure 68 montre le positionnement FlexiLab dans notre suite d'outils. Le développement d'une IHM plastique est guidé par un modèle de processus défini dans D2Flex et mis en œuvre grâce à R2Flex.

La première étape relève donc de l'ingénierie des processus : il s'agit de la création du modèle de processus de développement ❶ à l'aide de D2Flex ❷, ce modèle de processus étant une instance du métamodèle M2Flex ❸. Sur la figure, nous avons indiqué que c'est un chef de projet qui définit le modèle de processus qui sera utilisé dans le projet qu'il dirige.

Ce modèle de processus est ensuite exécuté, c'est-à-dire mis en œuvre par des développeurs et des concepteurs en ingénierie des IHM plastiques, grâce à R2Flex ❹. R2Flex instancie le modèle de processus pour le projet, et propose aux développeurs et aux concepteurs du guidage pour développer ces IHM. L'ingénieur s'adresse donc à R2Flex pour savoir ce qu'il doit faire ❺.

FlexiLab permet de générer des IHM par transformations de modèles ❻, qui sont indiqués sur la figure avec le nom « M1 de … ». R2Flex indique donc au concepteur ou au développeur comment réaliser ces modèles et ces transformations. En particulier, il

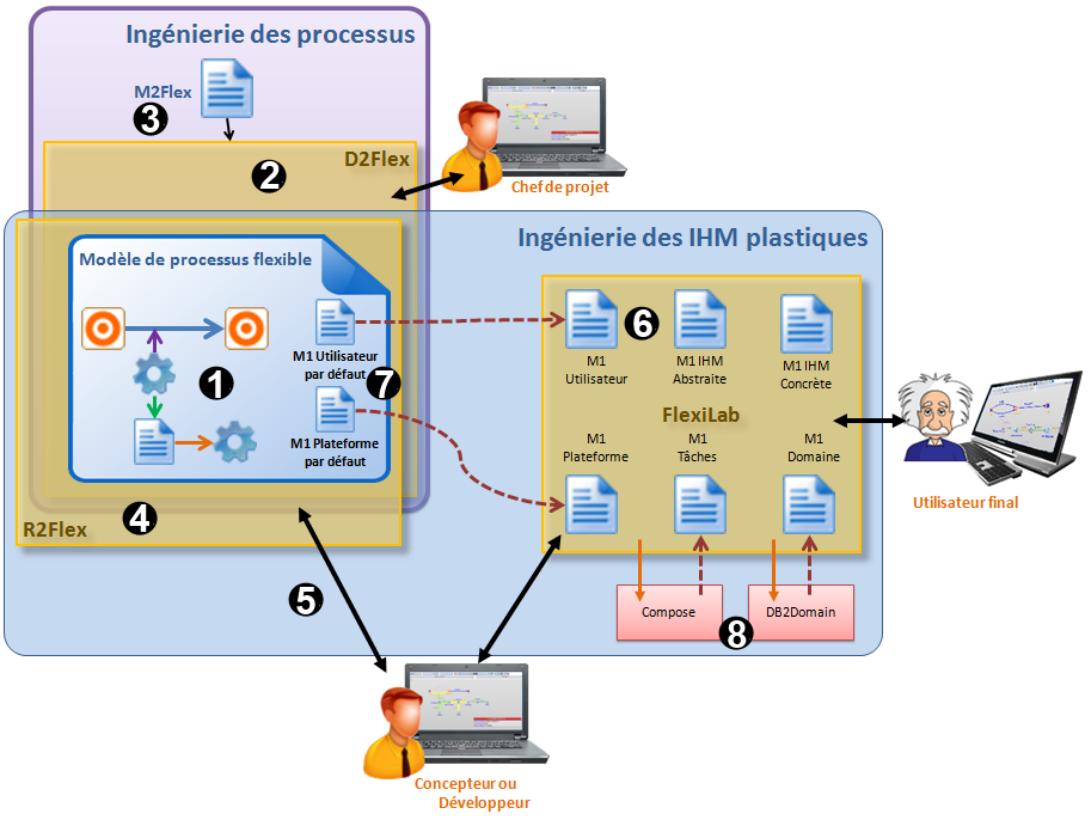

Figure 68 - Positionnement de FlexiLab dans la suite d'outils

peut proposer d'utiliser un modèle de plateforme ou un modèle d'utilisateur « par défaut » ❼, comme nous l'avons vu dans la section 7.3.2.4. Ces « modèles par défaut », qui sont élémentaires, sont copiés par le concepteur ou le développeur dans l'environnement de FlexiLab puis, si besoin, enrichis. Idéalement, nous aimerions que R2Flex soit capable de réaliser cette copie, mais cette fonctionnalité n'est pas implantée dans la version actuelle des produits.

R2Flex peut aussi proposer d'utiliser des applications extérieures comme Compose (Gabillon et al. 2011) ou DB2Domain pour générer des modèles. Nous verrons dans la suite de ce chapitre comment le concepteur ou le développeur peut réaliser une telle tâche.

8.1.1 Architecture de FlexiLab

L'architecture de FlexiLab est basée sur celle de UsiComp (García Frey et al. 2012), dont il reprend en particulier les services traitant des transformations ATL. Il étend l'architecture de UsiComp pour pouvoir mettre en œuvre les différents variants et éléments réutilisables que nous avons définis dans ce modèle de processus flexible.

FlexiLab est structuré autour de services implantés selon la spécification de OSGI (OSGI Alliance 2010). La figure 69 représente le schéma d'architecture de FlexiLab. Lors de la première étape du développement d'une application avec des interfaces plastiques, l'équipe de développement crée d'une part le noyau fonctionnel et d'autre part les modèles d'IHM et de transformations. FlexiLab est conçu pour gérer plusieurs applications qu'il

propose à l'utilisateur final. Chacune de ces applications dispose de ses propres métamodèles, ce qui permet de les personnaliser en cas de besoin.

Lorsque l'utilisateur se connecte à FlexiLab, il fait appel à un service « serveur » qui reçoit ses requêtes. Dans un premier temps, le serveur envoie la liste des applications disponibles pour permettre à l'utilisateur de choisir celle qu'il souhaite. Puis le serveur fait appel au service nommé « Contrôleur », qui va utiliser le modèle de transformation de l'application choisie pour lancer toutes les actions nécessaires à la génération de l'IHM. Il peut en particulier faire appel au service « Invocateur » pour obtenir des éléments d'une application extérieure comme Compose (Gabillon et al. 2011) ou notre module DB2Domain. Nous verrons dans la section suivante comment ce modèle de transformation a été adapté pour permettre des appels au service Invocateur, qui ne sont pas prévus dans le modèle de transformation de UsiXML. Il peut aussi faire appel au service « Contrôleur de dialogue » pour contacter le noyau fonctionnel, ce qui lui permet par exemple d'obtenir les instances du modèle de domaine. Enfin, il fait appel au service « Composeur » qui est chargé de l'exécution des transformations ATL. Ce service utilise lui-même trois autres services, chargés respectivement de vérifier la conformité entre les modèles et métamodèles, de compiler les transformations ATL et de les exécuter. Lorsqu'il a terminé la génération du code de l'IHM finale, il renvoie le chemin d'accès à ce fichier au contrôleur, qui se charge d'appeler si besoin le service « Compilateur » pour générer un exécutable. Enfin, le contrôleur appelle le service « Déployeur », qui va se charger de rendre l'exécutable

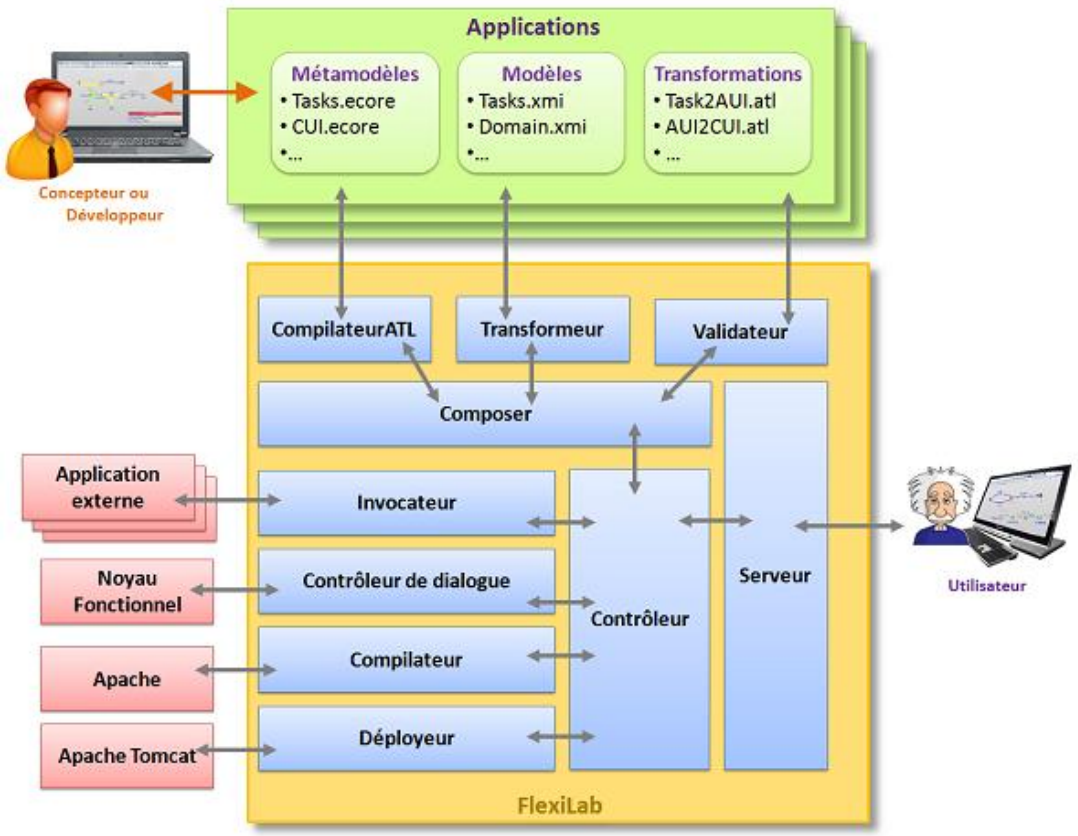

Figure 69 - Schéma d'architecture de FlexiLab

disponible pour l'utilisateur, en l'insérant dans Apache Tomcat ou en le copiant dans un dossier de Apache. Le déployeur renvoie une adresse qui est transmise au service « Serveur » puis au dispositif de l'utilisateur, qui accède à l'IHM à travers son navigateur Internet.

La prise en compte de la ou les plateformes de l'utilisateur n'a pas été représentée sur ce schéma. Dans la version actuelle de FlexiLab, l'utilisateur exécute une application cliente, écrite en Java, qui collecte les informations sur la plateforme et les transmet au serveur. Ce dernier garde en mémoire une représentation des utilisateurs et de leurs plateformes.

8.1.2 Adaptation du modèle de transformation UsiXML

Dans l'approche UsiXML, le processus de génération de l'IHM finale est défini par le modèle de transformation, qui spécifie les différentes opérations de transformation à mettre en œuvre pour générer l'IHM. Il permet de représenter des transformations ATL et des transformations de graphe. Ce modèle de transformation, tel qu'il est défini dans UsiXML (UsiXML Consortium 2012) ne permet pas d'intégrer les formes de flexibilité du modèle de processus UsiXML flexibilisé. Par exemple, il n'est pas possible d'indiquer qu'il faut générer le modèle de domaine à partir d'une base de données avant de l'utiliser dans les transformations. Il convenait donc de définir une version du modèle de transformation supportant ce type d'opérations.

Dans la description qui suit, nous illustrerons nos propos avec un cas d'étude, qui est une application dont nous avons développé les interfaces avec FlexiLab. Cette application s'appelle FindADoctor. Le besoin de son utilisateur est de trouver un médecin près de lui. FlexiLab génère une IHM adaptée à la plateforme, en listant les 10 médecins les plus proches de l'utilisateur (sa localisation étant fournie par son téléphone, par exemple). Elle affiche aussi une carte sur laquelle elle place des repères pour localiser les médecins.

Au moment où nous avons créé FlexiLab, le modèle de transformation de UsiXML n'était pas stabilisé. Nous avons donc créé une version simplifiée qui correspondait à nos besoins. L'objectif général est de décrire une séquence d'actions, qui peuvent être des transformations de modèles en modèles, des transformations de modèle en code, ou des descriptions représentant les différents variants : un appel à un programme extérieur comme Compose (Gabillon et al. 2011), la génération du modèle de domaine à partir d'une base de données, etc.

Le modèle de transformation FlexiLab agrège des actions, comme le montre la figure 70. Une action a deux attributs : une description, qui n'est pas utile pour le processus de génération mais qui facilite la lecture du processus de génération par un concepteur, et un booléen indiquant si l'action est active dans le processus. Cet attribut est surtout utilisé pendant le développement, il permet d'isoler des parties du processus pour les tester, en déclarant uniquement les actions concernées par le test actives. Une action est ensuite spécialisée en cinq classes :

- Les appels au noyau fonctionnel (« *FuncCoreCall* »). Il s'agit ici d'appeler un service, et dans notre version simplifiée nous n'avions pas besoin de faire appel à un dictionnaire de services. C'est donc le nom du service qui est géré dans la classe. Les services peuvent nécessiter des paramètres, qui ont un nom et une valeur. La valeur est ici aussi simplifiée, il s'agit d'une chaîne de caractères, dans laquelle peuvent être stockés si besoin des entiers ou des réels. L'ordre des paramètres n'importe pas : nous avons structuré ces services de façon à ce qu'ils acceptent en entrée un flux XML qu'ils interprètent. C'est aussi cette structure qui permet de nous dispenser de typer les paramètres : le service convertit les formats si besoin. Les appels au noyau fonctionnels sont utilisés dans FindADoctor pour obtenir les instances du modèle de domaine, c'est-à-dire les médecins avec leur localisation. Ces instances sont produites dans un format spécifique, propre au noyau fonctionnel, une transformation se chargeant de leur conversion dans un format conforme au métamodèle de domaine.
- Les exécutions d'applications extérieures (« *Execution* »), qui sont utilisées pour faire appel à des programmes extérieurs à FlexiLab. Comme il s'agit d'application dont le fonctionnement peut varier, les paramètres sont ici ordonnés, pour respecter l'ordre dans lequel l'application les attend. Dans

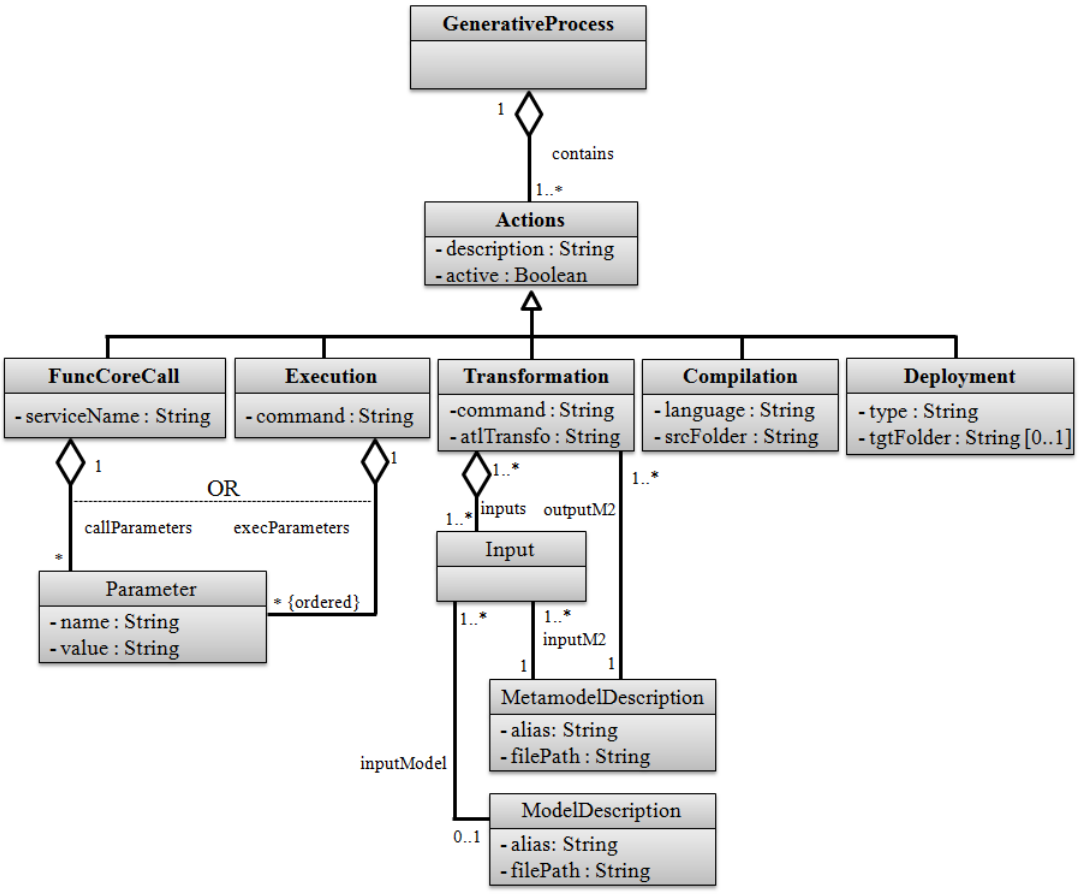

Figure 70 - Modèle de transformation de FlexiLab

FindADoctor, l'application demande

- la génération du modèle de tâches à l'application Compose (Gabillon et al. 2011). Nous avons demandé l'aide de Y. Gabillon pour produire les prédicats correspondant à notre cas d'étude. Le modèle de tâches issu de Compose est conforme à un métamodèle propre à cette application, il est donc ensuite transformé en modèle de tâches conforme au métamodèle de FlexiLab.
- la génération du modèle de domaine à partir d'une base de données mySQL. Cette base de données contient trois tables : Ville, Spécialité et Professionnels. Les professionnels peuvent avoir différentes spécialités et ils ont une adresse postale qui référence l'une des villes de la table Ville. Le module DB2Domain parcourt les métadonnées de la base pour générer un modèle de domaine.

La génération du modèle de tâches et du modèle de domaine dynamiquement pendant la génération de l'IHM pose un problème particulier. En effet, l'approche UsiXML repose sur la définition d'un modèle de correspondance (« *Mapping Model* ») qui permet d'identifier le concept manipulé par une tâche. En générant dynamiquement ces deux modèles, il n'était pas possible de demander au système d'inférer des correspondances sans point de repère. Nous avons donc structuré la transformation du modèle de domaine de Compose en modèle de domaine UsiXML de façon à produire des noms de concepts similaires à ceux que nous avions dans la base. Il s'agit évidemment d'une astuce qui ne pourrait pas être utilisée dans un projet réel : ce serait un élément de rigidité qui interdirait la modification du modèle de domaine et du modèle de tâches. Cependant, dans un projet réel, il semblerait utopique de générer dynamiquement ces modèles, à moins de disposer d'ontologies décrivant les univers de Compose et de la base de données et permettant d'inférer leurs correspondances. La démarche dans un projet réel serait de générer le modèle de tâche et/ou le modèle de domaine de façon statique et de créer ensuite le modèle de correspondance. Nous n'avons pas retenu cette solution, car elle rendait plus complexe les démonstrations des concepts de flexibilité : il aurait fallu soit ne pas montrer la génération des modèles, soit créer manuellement le modèle de correspondance pendant la démonstration, ce qui n'aurait pas été possible en raison du temps consommé. Le contournement que nous avons utilisé permet de montrer toutes les opérations qui s'enchaînent directement.

- Les transformations (« *Transformation* »), qui permettent de compiler si besoin et d'exécuter ensuite des règles ATL. Une règle ATL prend normalement en paramètre d'entrée des couples <métamodèle, modèle> définissant les modèles sources, ainsi que des métamodèles définissant les structures des modèles cibles. Dans notre cas, nous avons ajouté la possibilité que dans les structures sources, il n'y ait pas de modèle, car nous avons défini un héritage au sein des métamodèles, ce qui permet de considérer tous les éléments comme des

spécialisations d'une classe mère commune. Nous pouvons ainsi définir un unique métamodèle de « mapping » entre deux modèles. Le modèle de « mapping » permet ainsi de savoir à quelle tâche se réfère un composant de l'IHM abstraite.

Pendant le développement de FlexiLab, nous avons de créé une transformation ATL permettant de convertir un fichier Balsamiq Mockups en IHM concrète conforme à notre métamodèle. Balsamiq Mockups est un utilitaire disponible sur Internet[19] qui permet de créer des maquettes d'IHM. Cette transformation a été réalisée au cours d'un stage, et, si elle n'est pas intégrée dans FindADoctor, elle est disponible pour être réutilisée dans une autre application.

- Les compilations (« *Compilation* »), qui permettent de compiler le code généré en tant qu'IHM finale. Une action de compilation contient un attribut définissant le langage utilisé, ce qui permet de choisir le compilateur adapté. Dans FindADoctor, deux langages ont été mis en œuvre, Vaadin dans un premier temps et HTML dans un second temps. Vaadin est une extension de Java qui permet de produire du code HTML dans des servlets. La seconde version est donc une simplification du processus de génération de l'IHM, puisque nous évitons le passage par Vaadin et produisons directement du code HTML. L'attribut « dossier source » (« *srcFolder* ») permet d'identifier où la transformation de génération de code doit placer le fichier résultant.
- Les déploiements (« *Deployment* ») permettent de déployer l'application pour être mise à disposition de l'utilisateur. Dans le cas de FindADoctor, le code Vaadin compilé était déployé dans Apache Tomcat. Ce déploiement était malheureusement très long : 25 secondes dans une première version puis 2 secondes après optimisation, mais cette durée était instable, ce qui était un argument de plus pour générer directement du HTML. Dans la version actuelle, le déploiement du fichier HTML consiste simplement à le copier dans un répertoire prédéfini du serveur Internet. Ce répertoire est connu grâce à l'attribut « dossier cible » (« *tgtFolder* »).

FlexiLab, dans sa version actuelle, permet de mettre en œuvre les différents variants que nous avions étudiés : génération du modèle de tâches par une application extérieure, génération du modèle de domaine par notre module DB2Domain et transformation d'une maquette réalisée avec Balsamiq Mockups en IHM concrète.

Par ailleurs, FlexiLab est une extension de UsiComp, qui a été créé par A. Garcia-Frey (García Frey et al. 2012). Nous avons en particulier ajouté le modèle de transformation supportant les différentes actions et le service Invocateur. Le module de conception de UsiComp permet de concevoir les modèles et transformations nécessaires à la génération de l'IHM. Il comporte un dictionnaire de règles élémentaires de transformation, que le concepteur peut réutiliser pour définir les différentes réifications de modèles. Ces

[19] Voir http://balsamiq.com/products/mockups/

transformations élémentaires peuvent donc être réutilisées pour différentes applications. A l'issue du processus de conception, les règles élémentaires sont agrégées pour produire des transformations ATL. FlexiLab utilise ces transformations. L'agrégation de règles élémentaires pendant la conception présente cependant un inconvénient pour la flexibilité de l'approche : il n'est plus possible, pendant la génération de l'IHM de remplacer une règle élémentaire par une autre en fonction de la situation. Par exemple, dans FindADoctor, on ne peut pas adapter dynamiquement les interacteurs au nombre de médecins proches de l'utilisateur. La transformation créée lors de la conception produit une liste et il ne serait pas possible d'utiliser un autre interacteur, par exemple s'il n'y avait qu'un seul médecin proche de l'utilisateur. Il semblerait donc préférable dans FlexiLab de ne pas modéliser les transformations directement, mais plutôt les règles élémentaires et de procéder à leur agrégation dynamiquement, après avoir éventuellement remplacé une règle par une autre.

Résumé du chapitre 8

Ce chapitre a présenté une application de la flexibilité des modèles de processus telle qu'elle est définie par Promote. Nous avons montré comment nous pouvions "flexibiliser" le modèle de processus proposé par le Consortium UsiXML (UsiXML Consortium 2011).

Nous avons présenté l'augmentation des différentes formes de flexibilité, par l'ajout de variants, de niveaux de détails, de stratégies d'apprentissage et d'éléments de solutions. Ces ajouts nous ont permis de créer de nombreux chemins nouveaux dans le modèle de processus, comme la génération de l'IHM abstraite à partir de maquettes ou différentes approches pour créer les modèles de tâches et de domaine. Nous avons montré que ces différentes activités étaient raffinées en plusieurs niveaux de détails, permettant au concepteur ou au développeur de travailler avec des activités plus ou moins simples. Nous avons aussi montré qu'une nouvelle stratégie d'apprentissage basée sur des tutoriels en forme d'exercices corrigés était intégrable dans D2Flex. Enfin, nous avons ajouté des éléments de solutions en prenant comme exemple la génération du modèle de tâches par Compose (Gabillon et al. 2011) et celle du modèle de domaine par un module d'analyse des structures de bases de données que nous avons réalisé, DB2Domain.

La flexibilité du modèle de processus résultant a été évaluée et nous avons montré qu'il présente une flexibilité non seulement très supérieure à celle définie initialement dans UsiXML (UsiXML Consortium 2011) mais aussi à celles des modèles de processus que nous avons présentés dans l'état de l'art.

Enfin, nous avons présenté FlexiLab, un atelier de conception et d'exécution d'IHM plastiques, qui est structuré pour pouvoir mettre en œuvre les différents variants définis dans le modèle de processus. FlexiLab permet ainsi d'exécuter des applications comme Compose (Gabillon et al. 2011) pour générer le modèle de tâches ou DB2Domain pour générer le modèle de domaine. Il peut aussi, grâce à son modèle de transformation spécifique, mettre en œuvre les différents chemins de développement (et donc d'exécution) définis dans (Limbourg & Vanderdonckt 2009). FlexiLab est donc un atelier flexible permettant la mise en œuvre d'IHM plastiques.

La flexibilisation du modèle de processus de UsiXML constitue notre cinquième contribution.

9 VALORISATION

Dans la première partie de cette thèse, nous avons présenté Promote, notre taxonomie des modèles de processus, et dans les deux chapitres précédents, nous avons présenté le modèle de processus UsiXML flexibilisé et FlexiLab, notre atelier de développement d'IHM plastiques. Au-delà des évaluations que nous avons réalisées au cours de cette thèse sur ces différentes propositions, il nous paraissait particulièrement important que nos travaux dans les domaines de l'ingénierie des processus et des IHM plastiques soient utilisables par des équipes de développement. Nous avons donc mené des actions de dissémination dans ces deux domaines, aussi bien en formation qu'auprès des industriels. Nous avons aussi mené une action en vue de la maturation industrielle de FlexiLab. Ces valorisations sont présentées dans les sections suivantes.

9.1 Valorisation de l'ingénierie des processus

Promote, notre taxonomie des modèles de processus (Céret, Dupuy-Chessa, et al. 2013) a été utilisée dans le module *Ingénierie Avancées des Systèmes d'Information* du Master 2 Recherche option SIGAL (*Systèmes d'Information et Ingénierie Avancée du Logiciel*) de l'Unité de Formation et de Recherche en Informatique, Mathématiques et Mathématiques Appliquées de Grenoble (UFR IM²AG), en 2010-2011, 2011-2012 et 2013-2014. Lors de cette dernière année, D2Flex et R2Flex ont aussi été utilisés par les étudiants de ce module pour découvrir des modèles de processus et augmenter leur flexibilité.

C'est dans le cadre de ces enseignements que nous avons réalisé les différentes études de cas sur Promote (voir section 3.5) et sur D2Flex/R2Flex (voir section 6.4).

Par ailleurs, nous avons présenté notre approche de la flexibilité des modèles de processus au cours des Journées Neptune en juin 2013, sur le thème de « *L'ingénierie Dirigée par les Modèles : l'âge de raison ?* ». A la suite de cette présentation, nous avons été invités à faire une présentation similaire lors d'une journée d'échanges du Club Automation en octobre 2013, dont le thème était « *Ingénierie : nouvelles méthodes de conception hors du cycle en V: Mythe ou réalité ?* ».

Ces rencontres ont été particulièrement enrichissantes. Les ingénieurs et entrepreneurs présents faisaient en effet part de leurs difficultés dans la mise en œuvre des méthodes. Les contraintes contractuelles auxquelles ils étaient soumis leur imposaient en même temps de mettre en œuvre un modèle de processus donné et de le transgresser. Ainsi, l'un des participants rapportait que son client avait exigé l'utilisation d'un cycle en V (McDermid & Ripken 1984), qui prévoit normalement une livraison unique en fin de projet et n'envisage pas la participation des utilisateurs, tout en lui demandant d'intégrer un utilisateur final à l'équipe de développement et en contractualisant des livraisons de versions intermédiaires. D'autres intervenants expliquaient leurs difficultés dans la mise en œuvre d'approches agiles pour des clients qui exigeaient la rédaction en début de projet de spécifications semi-formelles, ce qui entrait en contradiction avec l'esprit de ces approches, dans lesquelles les

fonctionnalités à développer sont précisées au fur et à mesure de l'avancement du projet. Nous avons ainsi recueilli de nombreux retours sur les besoins de flexibilité dans les modèles de processus.

9.2 Valorisation de l'ingénierie des IHM plastiques

Dans le cadre du projet d'Investissement d'Avenir CONNEXION (COntrôle Commande Nucléaire Numérique pour l'EXport et la rénovatION), nous avons utilisé notre approche de la flexibilité pour faciliter l'apprentissage des concepts de la plasticité par différents partenaires. En partant de la culture déjà maîtrisée par les acteurs (notions de structure de données, d'espaces de travail, d'interacteurs, construction d'IHM classiques), nous avons présenté le processus de développement d'IHM plastiques, les différents métamodèles et les transformations.

Une fois les concepts essentiels transmis, nous avons mené une étude de faisabilité portant sur la rétro-ingénierie d'IHM existantes dans ADACS, une solution logicielle de la société Atos pour le contrôle-commande de centrales nucléaires, en vue de permettre la création d'une version plastique de ces IHM à destination des utilisateurs mobiles. Cette étude a fait l'objet d'une publication (Céret, Calvary, et al. 2013).

Cette formation et cette étude de faisabilité ont été assez convaincantes pour que la société Atos décide du développement d'interfaces plastiques. Une équipe spécialisée dans la modélisation a créé les modèles et transformations nécessaires pour la génération de quelques IHM.

Ces différentes études ont permis de mettre en exergue que les différents potentiels offerts par la plasticité ne sont pas tous convoités au même titre par nos partenaires dans le domaine du nucléaire.

Nous avons pu identifier trois catégories (Céret et al. 2014), que nous présentons ci-dessous.

La valeur sûre : le Multi*

Dans le potentiel de l'approche, c'est le multi-ciblage qui ressort comme valeur sûre, pertinente et attendue :

- ***l'aspect multi-utilisateurs***, en particulier en termes de métiers et d'expertises. La capacité, à partir d'un même ensemble de modèles, de générer autant d'interfaces que de métiers différents, est perçue comme une force dans le domaine du nucléaire où de nombreux métiers cohabitent et où la cohérence des interfaces et des données manipulées dans ces interfaces est nécessaire pour la sécurité du système. L'idée est de pouvoir diffuser et adapter les interfaces de surveillance du procédé pour des agents mobiles, par exemple les différents acteurs de la maintenance. La possibilité d'adapter le degré de guidage de l'utilisateur en fonction de son expertise déclarée ou perçue est également identifiée comme une force de l'approche, la formation des opérateurs étant longue et délicate ;

- ***L'aspect multi-plates-formes***, particulièrement en termes de portées et d'échelles. Pouvoir générer autant de variantes d'IHM que de plates-formes et d'usages considérés est là aussi une force de l'approche pour le projet Connexion. En effet, de nouveaux dispositifs (bornes d'information, tablettes, …) sont envisagés en renfort des plates-formes actuelles, qui sont fixes et centralisées en salle de commande. Des dispositifs dispersés comme des bornes d'information permettraient aux agents de maintenance d'accéder aisément et rapidement à une information adaptée au dispositif, à la situation du site et à l'utilisateur. Les informations diffusées sont cependant nombreuses et complexes, le coût de cette approche serait donc la création et l'instanciation d'un modèle de domaine complexe, ainsi que la création d'un modèle d'interacteurs spécifiques. Ce prix à payer, sans doute important, semble acceptable étant donnés les retours sur investissement pressentis : formation des opérateurs, évolutivité des informations, interopérabilité et bien entendu plasticité ;
- ***L'aspect multi-environnements***, en particulier en termes de luminosité et de bruit. Par exemple, lorsque plusieurs opérateurs collaborent, il est important de réduire les retours sonores.

La valeur convoitée : le Trans pré-calculé.

Au-delà du Multi*, les changements dynamiques de contexte d'usage sont une réalité dans le domaine du nucléaire. Par exemple, un agent est régulièrement amené à interrompre sa tâche courante pour répondre à une demande d'intervention urgente, comme une action de maintenance préventive ou la manipulation d'un équipement pour adapter l'état de l'installation à la production d'électricité souhaitée. Il est probable, à cette occasion, que l'agent soit amené à se déplacer et éventuellement à changer de dispositif, par exemple à passer d'une borne d'information fixe à une tablette. Ces contraintes posent le besoin de plasticité. La piste aujourd'hui envisagée est celle de contextes d'usage et de changements de contextes d'usage totalement prévus à la conception, ne laissant aucune place à l'imprévu pour des raisons de sécurité et de validation des IHM. Il s'agit donc d'adaptation aux contextes pré-calculés.

La valeur « diabolisée » : le Trans dynamiquement calculé.

La génération dynamique d'IHM pour faire face à des contextes d'usage non prévus à la conception n'a pas été jugée pertinente en exploitation dans les centrales nucléaires. Néanmoins, ce potentiel peut garder sa place en phase de conception pour générer des variantes d'IHM et peut-être inspirer les concepteurs.

9.3 Maturation industrielle

Depuis avril 2013, nous avons, en parallèle des actions décrites ci-dessus, lancé une étude de maturation industrielle de FlexiLab, avec le soutien de Gravit, un dispositif mutualisé de transfert de technologies. Cette étude a mené à la définition d'un plan d'action pour la valorisation de FlexiLab.

En particulier, nous avons mené une analyse du marché en 2013, basée sur la rencontre de plusieurs entreprises, auxquelles nous avons présenté FlexiLab et dont nous avons recueilli les remarques. Nous avons ainsi rencontré (1) BPM, qui a confirmé la complémentarité de FlexiLab avec son outil FlowMind, spécialisé en modélisation des processus ; (2) VisioGlobe, qui a confirmé l'intérêt de FlexiLab pour les éditeurs logiciels ; (3) Motwin, qui a affirmé son intérêt pour des formulaires plastiques destinés à la présentation et à la saisie de données, notamment pour des déclarations d'incidents auprès d'assurances ; (4) Bonitasoft, qui a marqué son intérêt pour la génération d'IHM ergonomiques et insisté sur l'importance de la standardisation (prise en compte des normes dans les métamodèles), d'une approche incrémentale (possibilité de raffiner l'IHM peu à peu) et d'une étude comparative de FlexiLab par rapport au Responsive Design en particulier.

A la suite de ces rencontres, nous avons dans un premier temps procédé à l'étude comparative avec le Responsive Design, et montré les limites de cette technique par rapport à une approche de plasticité : l'adaptation ne concerne que la surface d'affichage et ne permet de traiter que les sites Internet.

Par la suite, ces différents éclairages nous ont permis de formuler une demande de maturation à Gravit. La demande initiale consistait en la consolidation du moteur de plasticité de FlexiLab, qui est un démonstrateur académique de concept, afin de pouvoir traiter des cas industriels. Gravit nous a accordé un financement permettant :

(1) le financement d'un poste d'ingénieur pendant quatre mois, destiné à la consolidation du moteur de plasticité en vue de la création d'un démonstrateur industriel. Cette consolidation démarrera le 1 er juin 2014 jusque fin septembre 2014 ;
(2) le financement d'une analyse de marché et une scénarisation de transfert

Figure 71 - Annonce de l'entrée en maturation de FlexiLab.

industriel, qui ont été confiés au cabinet Merkapt. Le cabinet a démarré son étude le 25 avril 2014 et restituera ses conclusions mi-juillet 2014.

L'entrée en maturation de FlexiLab a été annoncée par Gravit dans sa lettre d'information d'octobre-novembre 2013 (voir figure 71).

Résumé du chapitre 9

Dans ce chapitre, nous avons présenté les différentes actions de valorisation que nous avons menées au cours de cette thèse. Ces actions se sont déroulées (1) dans le domaine de la formation, auprès d'étudiants et d'ingénieurs en informatique et (2) en termes de transfert vers l'industrie et de maturation industrielle.

Nous avons présenté l'étude de cas que nous avons conduite pour le développement d'IHM plastiques au service d'acteurs mobiles dans les centrales nucléaires, et le transfert de savoir-faire vers la société Atos.

Enfin, nous avons présenté une action de maturation industrielle de notre atelier de développement d'IHM plastiques. Cette action est actuellement en cours et se déroule sur deux plans : une étude permettant de cibler les besoins du marché et la consolidation du moteur de plasticité de FlexiLab, afin qu'il puisse passer à l'échelle d'un démonstrateur de niveau industriel.

10 CONCLUSION ET PERSPECTIVES

Ce chapitre rappelle les contributions de la thèse puis ouvre des perspectives pour des recherches futures.

10.1 Rappel des contributions

Cette thèse propose un ensemble de supports pour la flexibilité des modèles de processus.

Tout d'abord, nous avons présenté Promote, une taxonomie des modèles de processus permettant d'évaluer la flexibilité de ces derniers, à l'aide de six critères identifiés à partir des besoins de flexibilité exprimés par les concepteurs et les développeurs. Ces six critères sont :

1) la **variabilité**, c'est-à-dire la capacité d'un modèle de processus à proposer différents chemins parmi les éléments qui le composent,
2) la **granularabilité,** qui est la capacité d'un modèle de processus à offrir différentes granularités, autrement dit différents niveaux de détails dans les éléments qui le constituent,
3) les **stratégies d'apprentissage**, qui sont les différentes modalités d'apprentissage permises par un modèle de processus
4) la **distensibilité**, c'est-à-dire la capacité d'un modèle de processus à être étendu ou réduit pendant son exécution
5) la **réutilisation**, qui mesure la proposition d'éléments de solutions dans un modèle de processus
6) le **polymorphisme**, qui évalue si un modèle de processus permet à ses utilisateurs de l'aborder selon ses différents constituants, grâce à des présentations multiples.

Nous avons montré que l'évaluation de différents modèles de processus à l'aide de Promote permet de distinguer les formes de flexibilité qu'ils offrent, ce qui permet de comparer la flexibilité incluse dans ces modèles de processus.

Nous avons également proposé M2Flex, un métamodèle de processus flexibles permettant de créer des modèles de processus avec toutes les formes de flexibilité. Ce métamodèle permet de définir différentes stratégies pour réaliser les étapes du processus, et permet de définir des choix entre les activités qui concrétisent ces stratégies. Il offre autant de granularités que souhaité dans les activités et dans les artefacts et permet d'inclure différentes formes d'aide pour l'apprentissage. Il permet aussi de définir qu'un artefact est préexistant, ce qui autorise à réutiliser des résultats provenant de projets précédents ou à s'inspirer d'exemples prédéfinis. Ce métamodèle permet d'accéder, par navigation, directement aux stratégies, aux activités et aux artefacts, ce qui autorise à le représenter selon ces trois perspectives. Il est par ailleurs accompagné de procédures algorithmiques pour valider les éventuelles modifications que les concepteurs et les

développeurs pourraient définir pendant l'exécution du modèle de processus qui l'instancie. Le métamodèle que nous proposons permet donc de construire des modèles de processus comportant toutes les formes de flexibilité définies dans Promote.

Par ailleurs, nous avons proposé deux outils dédiés à la conception et à l'exécution des modèles de processus instances de M2Flex et montré qu'ils supportent toutes les dimensions de la flexibilité qui peuvent être incluses dans un modèle de processus.

Nous avons enfin présenté une application de la flexibilité des modèles de processus à UsiXML, une méthode de développement d'IHM plastiques. Nous avons montré comment flexibiliser ce modèle de processus en intégrant des variants, des niveaux de détails supplémentaires, des stratégies d'apprentissage (documentation, tutoriels interactifs, reformulation d'activités) et d'éléments de solution réutilisables. L'évaluation de la flexibilité de ce modèle augmenté a permis de montrer que, dans chacune des dimensions, il offre plus de flexibilité que les modèles de processus étudiés dans notre état de l'art.

Enfin, nous avons présenté FlexiLab, notre atelier de conception et d'exécution d'IHM plastiques, permet la mise en œuvre concrète des formes de flexibilité proposées par le modèle de processus de UsiXML augmenté. FlexiLab se positionne ainsi comme un atelier flexible permettant la mise en œuvre d'IHM plastiques.

10.2 Perspectives

Nous pouvons évoquer différentes pistes pour poursuivre les recherches menées dans cette thèse.

En ingénierie des méthodes, le métamodèle devra être complété pour prendre en compte les aspects de gestion de projet, c'est-à-dire les évaluations de charges et de coûts, ainsi que la planification et la répartition des activités sur les membres de l'équipe de développement. Les évaluations devront être complétées. En particulier, il serait intéressant de conduire une expérience d'utilisation d'un modèle de processus flexible par des équipes de développement novices et expérimentées, afin d'évaluer l'utilisabilité des outils en situation réelle.

Un processus d'amélioration continue devra être intégré dans le cycle de vie des modèles de processus, grâce à la capitalisation des retours d'expériences. Ainsi, lors de discussion avec un consultant en méthodes agiles, ce dernier s'est montré très intéressé par la possibilité d'ajouter des activités dans le modèle de processus pendant son exécution, en particulier pour mémoriser les motivations ayant conduit au choix d'une activité ou d'une technique, et ainsi améliorer les recommandations dans des projets ultérieurs. Dans le même ordre d'idées, il faudra inclure des indicateurs de qualité sur les différentes parties du modèle de processus, permettant ainsi aux équipes d'évaluer l'efficacité des options qu'elles auront choisies. Un travail de recherche sur la propagation des modifications faites par une équipe aux autres instances du modèle de processus sera aussi nécessaire à plus long terme.

En plasticité des interfaces homme-machine, l'atelier de développement FlexiLab devra être augmenté pour passer à l'échelle d'une utilisation en conditions réelles.

En particulier, un travail de recherche sera nécessaire pour permettre le développement d'IHM complexes par ingénierie dirigée par les modèles, par exemple des IHM en trois dimensions ou des interfaces permettant de coupler les mondes réels et virtuels.

11 BIBLIOGRAPHIE

Abbas, N., Gravell, A. & Wills, G., 2008. Historical Roots of Agile Methods: Where Did « Agile Thinking » Come From? In P. Abrahamsson et al., eds. *Agile Processes in Software Engineering and Extreme Programming*. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, p. 94–103.

Agerfalk, P.J. & Fitzgerald, B., 2006. Flexible And Distributed Software Processes: Old Petunias In New Bowls? *Commun. ACM*, 49(10), p.26–34.

Alexander, L.C. & Davis, A.M., 1991. Criteria for selecting software process models. In *Computer Software and Applications Conference, 1991. COMPSAC '91., Proceedings of the Fifteenth Annual International*. p. 521 –528.

Alford, M.W., 1977. A Requirements Engineering Methodology for Real-Time Processing Requirements. *IEEE Transactions on Software Engineering*, vol. 3 (n°. 1), p. 60–69.

Ambler, S.W., 2011. Choose the Right Software Method for the Job. *Agile Data*. En ligne : http://www.agiledata.org/essays/differentStrategies.html [Accès le 31/10/2011].

Anderson, D.J., 2004. *Agile management for software engineering: Applying the theory of constraints for business results*, Prentice Hall Professional.

Aquino, N. *et al.*, 2010. Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering. *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* p. 1-10, Bolzano-Bozen, Italy

Atkinson, C. & Kühne, T., 2001. The Essence of Multilevel Metamodeling. In M. Gogolla & C. Kobryn, eds. *UML 2001 — The Unified Modeling Language. Modeling Languages, Concepts, and Tools*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 19–33.

Avison, D.E. & Fitzgerald, G., 2003. Where Now for Development Methodologies? *Commun. ACM*, vol. 46 (n° 1), p.78–82.

Bach, J., 1995. The Challenge of « Good Enough » Software. *American Programmer*, vol. 8 (n°. 10), p. 2–11.

Barry, C. & Lang, M., 2001. A Survey of Multimedia and Web Development Techniques and Methodology Usage. *IEEE MultiMedia*, vol. 8 (n°. 2), p. 52–60.

Basili, V. & Turner, J., 1975. Iterative Enhancement: A Practical Technique for Software Development. *IEEE Trans. Software Eng.*, p. 390– 396.

Beck, K., 1999a. Embracing Change with Extreme Programming. *IEEE Computer*, vol. 32 (n°. 10), p. 70–77.

Beck, K., 1999b. *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, 190 p.

Beck, K., 2002. *Extreme programming, La Référence*, CampusPress, 180 p.

Beck, K. & Fowler, M., 2000. *Planning Extreme Programming* 1st ed., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Bell, T.E., 1981. Structured Life-cycle Assumptions. In *Proceedings of the 1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality*. New York, NY, USA: ACM, p. 1–3.

Bell, T.E. & Thayer, T.A., 1976. Software Requirements: Are They Really a Problem? In *Proceedings of the 2nd International Conference on Software Engineering*. ICSE '76. Los Alamitos, CA, USA: IEEE Computer Society Press, p. 61–68..

Bénard, J.-L. et al., 2002. *Gestion de projet eXtreme Programming*, Eyrolles, 300 p.

Bendraou, R. et al., 2007. Software Process Modeling and Execution: The UML4SPM to WS-BPEL Approach. In *EUROMICRO-SEAA*. p. 314–321.

Bennington, H.D., 1983. Production of Large Computer Programs. *Annals of the History of Computing*, 5(4), p.350–361.

Boehm, B., 1986. A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*, vol. 11, p. 14–24.

Boehm, B., 1976. Software Engineering. *IEEE Transactions on Computers*, vol. 25 (n°. 12), p. 1226–1241.

Boehm, B., 1989. Software risk management. In C. Ghezzi & J. McDermid, eds. *ESEC '89*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 1–19.

Boehm, B. et al., 1998. Using the WinWin Spiral Model: A Case Study. *Computer*, vol. 31 (n°7), p.33–44.

Boehm, B.W., 1979. Software Engineering as it is. In *Proceedings of the 4th International Conference on Software Engineering*. ICSE '79. Piscataway, NJ, USA: IEEE Press, p. 11–21.

Boehm, B.W. & Hansen, W.J., 2001. The Spiral Model as a Tool for Evolutionary Acquisition. *The Journal of Defense Software Engineering*, vol 4, n°11, p.4–10.

Boehm, B.W., McClean, R.K. & Urfrig, D.B., 1975. Some experience with automated aids to the design of large-scale reliable software. In *Proceedings of the international conference on Reliable software*. New York, NY, USA: ACM, p. 105–113.

Booch, G., 1993. *Object-Oriented Analysis and Design with Applications*, Addison-Wesley Professional, 2nd ed., 534 p.

Breedvelt-Schouten, I.M., Paternó, F.D. & Severijns, C.A., 1997. Reusable Structures in Task Models. In M. Harrison & J. Torres, eds. *Design, Specification and Verification of Interactive Systems '97*. Eurographics. Springer Vienna, p. 225–239.

Brinkkemper, S., Saeki, M. & Harmsen, F., 1998. Assembly Techniques for Method Engineering. In *Proceedings of CAISE'98*. Pisa, Italy, p. 381-400.

Bucher, T. et al., 2007. Situational Method Engineering. In J. Ralyté, S. Brinkkemper, & B. Henderson-Sellers eds. *Situational Method Engineering: Fundamentals and Experiences*. IFIP, The International Federation for Information Processing. Springer US, p. 33–48.

Calvary, G. et al., 2003. A Unifying Reference Framework for multi-target user interfaces. *Interacting with Computers*, vol. 15, n° 3, p.289 – 308.

Calvary, G., Coutaz, J. & Thevenin, D., 2001. A Unifying Reference Framework for the Development of Plastic User Interfaces. In M. Little & L. Nigay, eds. *Engineering for Human-Computer Interaction*. Springer Berlin / Heidelberg, p. 173–192.

Céret, E., Dupuy-Chessa, S., Calvary G., Front A., Rieu D., 2013. A taxonomy of design methods process models. *Information and Software Technology, Elsevier*, vol. 55, n°. 5, p. 795–821.

Céret, E., Dupuy-Chessa, S., Laurillau, Y., Calvary, G. La plasticité des interfaces homme-machine pour la performance des usines. *Génie Logiciel*, Hors série, p. 47–55.

Céret, E., Calvary, G. & Dupuy-Chessa, S., 2013. Flexibility in MDE for scaling up from simple applications to real case studies: illustration on a Nuclear Power Plant. In *25ème conférence francophone sur l'Interaction Homme-Machine, IHM'13*. Bordeaux, France: ACM, p. 33-42.

Coad, P., Lefebvre, E. & DeLuca, E., 1999. *Java Modeling in Color with UML: Enterprise Components and Process*, Upper Saddle River, NJ: Prentice Hall PTR.

Cockburn, A., 1995. Unraveling Incremental Development. *Object Magazine*, p. 49–51.

Cockburn, A. & Highsmith, J., 2001. Agile software development, the people factor. *Computer*, vol. 34, n° 11, p.131–133.

Constantine, L.L., Biddle, R. & Noble, J., 2003. Usage-Centered Design and Software Engineering: Models for Integration. In *ICSE Workshop on SE-HCI'03*. p. 106–113.

Cook, J.E. & Wolf, A.L., 1999. Software process validation: quantitatively measuring the correspondence of a process to a model. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 8, p.147–176.

Coutaz, J., 2007. Meta-User Interfaces for Ambient Spaces. In K. Coninx, K. Luyten, & K. Schneider, eds. *Task Models and Diagrams for Users Interface Design*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 1–15.

Coyette, A., Vanderdonckt, J. & Limbourg, Q., 2007. SketchiXML: A Design Tool for Informal User Interface Rapid Prototyping. In N. Guelfi & D. Buchs eds. *Rapid Integration of Software Engineering Techniques*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 160–176.

Crampes, J.-B., 2008. Méthode MACAO, démarches alternatives. *Jean-Bernard Crampes Consultants*. En ligne: http://www.jbcc.fr/ [Accès le 02/03/2011].

Crampes, J.-B., 2002. *Méthode Orientée-objet intégrale MACAO : démarche participative pour l'analyse, la conception et la réalisation de logiciels*, Paris, France, 308 p: Ellipses.

Credoc, 2011. *La diffusion des technologies de l'information et de la communication dans la société française*, Paris, France.

Curtis, W. et al., 1987. On Building Software Process Models Under the Lamppost. In *Proceedings of the 9th International Conference on Software Engineering*. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press, p. 96–103.

Cusumano, M.A. & Selby, R.W., 1997. How Microsoft builds software. *Commun. ACM*, vol. 40, n°6, p.53–61.

Dardenne, A., van Lamsweerde, A., Fickas, S., 1993. Goal-directed requirements acquisition. *Science of Computer Programming*, vol. 20, p.3 – 50.

Debjit, 2010. Sharp J-SH04: World's First Ever Phone With Integrated Camera. En ligne: http://gadgetizor.com/sharp-j-sh04-worlds-first-ever-phone-with-integrated-camera-pictures-2001/5482/ [Accès le 16/05/2014].

Demuynck, M. & Meyer, B., 1979. Les Langages de Spécification. *Bulletin de la Direction des Etudes et Recherches d'Electricité de France, Série C (Informatique), n°. 1*, p. 39–60.

Deneckère, R. et al., 2009. From Method Fragments to Method Services. In *Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'08)*.

Django Software Foundation, 2005. Django, the Web framework for perfectionists with deadlines. En ligne : http://www.djangoproject.com/ [Accès le 29/02/2012].

Dowson, M., 1987. Iteration in the software process; review of the 3rd International Software Process Workshop. In *Proceedings of the 9th international conference on Software Engineering*. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press, p. 36–41.

EMC, 2011. The 2011 IDC Digital Universe Study, « Extracting Value from Chaos», *EMC*. En ligne : http://www.emc.com/leadership/programs/digital-universe.htm.

Evans, E., 2004. Domain-driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, Wesford, MA, USA, 529 p.

Farago, P., 2012. iOS and Android Adoption Explodes Internationally, *Flurry*. En ligne: http://www.flurry.com/bid/88867/iOS-and-Android-Adoption-Explodes-Internationally.

Feiler, P.H. & Humphrey, W.S., 1993. Software process development and enactment: concepts and definitions. In *Software Process, 1993. Second International Conference on the Continuous Software Process Improvement*. p. 28–40.

Ferry, N., 2008. *Formalisation des modèles de la méthode MACAO et réalisation d'un outil de génie logiciel pour la création d'interfaces homme-machine*. Thèse de l'Université de Toulouse, France. 266 p.

Fichman, R.G. & Kemerer, C.F., 1993. Adoption of Software Engineering Process Innovations: The Case of Object-orientation. *Sloan Management Review*, vol. 34, n° 2, p. 7–22.

Fitzgerald, B., 1998. An empirical investigation into the adoption of systems development methodologies. *Information & Management*, vol. 34, n°. 6, p. 317 – 328.

Fitzgerald, B., Russo, N.L. & O'Kane, T., 2003. Software Development Method Tailoring at Motorola. *Commun. ACM*, vol. 46, n°4, p.64–70.

Fowler, M. & Highsmith, J., 2001. The Agile Manifesto Miller Freeman, Inc., ed. *Software Development*, vol. 9, n° 8, p. 28–32.

Fox, S., 2002. *Search Engines*, PewReseach Internet Project. En ligne : http://www.pewinternet.org/2002/07/03/search-engines/.

France, R. & Rumpe, B., 2007. Model-driven Development of Complex Software: A Research Roadmap. In *2007 Future of Software Engineering*. FOSE '07. Washington, DC, USA: IEEE Computer Society, p. 37–54.

Gabillon, Y. et al., 2011. Automated planning for user interface composition. In *Proceedings of the 2nd International Workshop on Semantic Models for Adaptive Interactive Systems: SEMAIS'11 at IUI 2011 conference*. Springer HCI.

Gamma, E. et al., 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*,

García Frey, A. et al., 2012. UsiComp: an extensible model-driven composer. In *proceedings*. ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS). p. 263–268.

Garzotto, F. & Perrone, V., 2007. Industrial Acceptability of Web Design Methods: an Empirical Study. *Journal of Web Engineering*, vol. 6, n°. 1, p. 73–96.

Glass, R.L., 1969. An Elementary Discussion of Compiler/Interpreter Writing. *ACM Computer Surveys*, vol. 1, n°. 1, p. 55–77.

Goldberg, R.P., 1974. Survey of virtual machine research. *Computer*, vol 7, n° 6, p.34–45.

Golra, F.R., 2014. *A Refinement based methodology for software process modeling*. Thèse réalisée à Télécom Bretagne, Brest, France.

Gonzalez-Perez, C. & Henderson-Sellers, B., 2008. A work product pool approach to methodology specification and enactment. *Journal of Systems and Software*, vol. 81, n°. 8, p. 1288–1305.

Griffiths, T. et al., 2001. Teallach: a model-based user interface development environment for object databases. *Interacting with Computers*, vol. 14, n°. 1, p. 31 – 68.

Guzelian, G., 2007. *Conception de systèmes d'information, une approche orientée service*. Thèse réalisé à l'Université Paul Cézanne, Marseille, France.

Guzman, A.R. et al., 2012. Assessing a Web Engineering Method in Practice: a Preliminary Analysis for Personal Genomics Portals. In *CIbSE*. p. 250–263.

Hardy, C.J., Thompson, J.B. & Edwards, H.M., 1995. The use, limitations and customization of structured systems development methods in the United Kingdom. *Information and Software Technology*, vol. 37, n° 9, p.467 – 477.

Harmsen, F., 1997. *Situational Method Engineering*. Thèse de l'Université de Twente et de Moret, Ernst & Young Management Consultants. 351 p.

Harmsen, F., Brinkkemper, S. & Oei, J.L.H., 1994. Situational method engineering for informational system project approaches. In *Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle*. Elsevier Science Inc., New York, NY, USA, p. 169–194.

Harrison, N.B. & Coplien, J.O., 1996. Patterns of Productive Software Organizations. *Bell Labs Technical Journal, 1(1)*, p.138–145.

Hassine, I. et al., 2002. Symphony: a conceptual model based on business components. In *SMC'02, IEEE International Conference*. Systems, Man and Cybernetics. Hammamet, Tunisie.

Heitmeyer, C., 1998. On the need for practical formal methods. In A. Ravn & H. Rischel eds. *Formal Techniques in Real-Time and Fault-Tolerant Systems*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 18–26.

Helms, J.W., 2001. *Developing and Evaluating the (LUCID/Star)*Usability Engineering Process Model*, Virginia Polytechnic Institute and State University. En ligne : http://scholar.lib.vt.edu/theses/available/etd-05102001-190814/unrestricted/jhelmsthesisnew.pdf.

Henderson-Sellers, B. & Edwards, J.M., 1990. The object-oriented systems life cycle. *Communications of the ACM*, vol. 33, p. 142–159.

Henderson-Sellers, B., Gonzalez-Perez, C. & Ralyté, J., 2008. Comparison of Method Chunks and Method Fragments for Situational Method Engineering. In *Australian Software Engineering Conference*. p. 479–488.

Henderson-Sellers, B. & Ralyté, J., 2010. Situational Method Engineering: State-of-the-Art Review. *Journal of Universal Computer Science*, vol. 16, n° 3, p.424–478.

Hess, G.D., 1980. *A Software Design System*. Thèse de l'Institut de Technologie de Californie. Pasadena, CA, USA. 193 p.

Hix, D. & Hartson, H.R., 1993. *Developing User Interfaces: Ensuring Usability Through Product &Amp; Process*, New York, NY, USA: John Wiley & Sons, Inc.

Hug, C., 2009. *Méthode, modèles et outil pour la méta-modélisation des processus d'ingénierie de systèmes d'information*. Thèse de l'Université de Grenoble, 222 p.

Hug, C., Front, A. & Rieu, D., 2008. A Process Engineering Method based on a Process Domain Model and Patterns. In *Proceedings of the International Workshop on Model Driven Information Systems Engineering: Enterprise, User and System Models (MoDISE-EUS'08) held in conjunction with the CAiSE'08 Conference*. Montpellier, France, p. 126–137. En ligne : http://hal.archives-ouvertes.fr/hal-00450717.

Humphrey, W.S. & Kellner, M.I., 1989. Software process modeling: principles of entity process models. In *Proceedings of the 11th international conference on Software engineering*. ICSE '89. New York, NY, USA: ACM, p. 331–342.

INSEE, 2011. Deux ménages sur trois disposent d'internet chez eux. *Insee Première*, p.1340.

International Organization for Standardization, 1999. Human-centred design processes for interactive systems. ISO 13407:1999.

International Organization for Standardization, 2007. ISO/IEC 24744 - Software Engineering - Metamodel for Development Methodologies.

Introna, L.D. & Whitley, E.A., 1997. Against method-ism: Exploring the limits of method. *IT & People*, vol. 10, n°. 1, p. 31–45.

Jaimes, N., 2013. La France compte 24,1 millions de possesseurs de smartphones. *Journal du Net*. En ligne : http://www.journaldunet.com/ebusiness/internet-mobile/equipement-et-usages-des-smartphones-0613.shtml.

Jarke, M. et al., 1993. Theories underlying requirements engineering: an overview of NATURE at Genesis. In *Proceedings of the IEEE International Symposium on Requirements Engineering*. San Diego, CA, USA, p. 19–31.

Jefferson, D.K., 1979. Analysis of requirements for a large-scale information system. In *Computer Software and Applications Conference, 1979. Proceedings. COMPSAC 79. The IEEE Computer Society's Third International*. p. 282–282.

Jerónimo, J.C. et al., 2009. Design of a Model of Human Interaction in Virtual Environments. In V. Lopez Jaquero et al., eds. *Computer-Aided Design of User Interfaces VI*. Springer London, p. 89–101.

Kabbaj, M., Lbath, R. & Coulette, B., 2008. A Deviation Management System for Handling Software Process Enactment Evolution. In Q. Wang, D. Pfahl, & D. Raffo, eds. *Making Globally Distributed Software Development a Success Story*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 186–197.

Kalay, Y.E., 2006. The impact of information technology on design methods, products and practices. *Design Studies*, vol. 27, n°. 3, p. 357 – 380.

Karlsson, F. & Ågerfalk, P.J., 2004. Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, vol. 46, n° 9, p.619 – 633.

Kolb, D.A., 1976. *Learning Style Inventory: Self Scoring Test and Interpretation*, Boston, MA, USA: McBer and Company.

Kolodner, J.L., Simpson, R.L. & Sycara-Cyranski, K., 1985. A Process Model of Cased-Based Reasoning in Problem Solving. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*. Los Angeles, California, USA, p. 284–290.

Kroeger, T. & Davidson, N., 2009. A Perspective-Based Model of Quality for Software Engineering Processes. In *Software Engineering Conference, 2009. ASWEC '09. Australian*. p. 152–161.

Kroll, P. & Kruchten, P., 2003. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison Wesley.

Kruchten, P., 2003. *The Rational Unified Process: An Introduction* 3rd ed., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Krueger, C.W., 1992. Software Reuse. *ACM Comput. Surv.*, vol. 24, n° 2, p.131–183.

Kumar, K. & Welke, R.J., 1992. Method engineering: a proposal for situation-specific methodology construction. In W. W. Cotterman & J. A. Senn, eds. *Challenges and Strategies for Research in Systems Development*. New York, NY, USA: John Wiley & Sons, Inc., p. 257–269.

Laanti, M., 2008. Implementing Program Model with Agile Principles in a Large Software Development Organization. In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*. Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International. p. 1383–1391.

Lanza, M. et al., 2005. CodeCrawler: an information visualization tool for program comprehension. In *ICSE*. p. 672–673.

Lee, J.C., 2010. *Integrating scenario-based usability engineering and agile software development*. Virginia Polytechnic Institute and State University.

Lekkos, A.A., 1979. Some Conclusions on a Top-down Functional Specifications Definition Experiment. *SIGDA Newsletter*, vol. 9, n°. 1, p. 19–23.

Limbourg, Q. et al., 2005. USIXML: A Language Supporting Multi-path Development of User Interfaces. In R. Bastide, P. Palanque, & J. Roth, eds. *Engineering Human Computer Interaction and Interactive Systems*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 134–135.

Limbourg, Q. & Vanderdonckt, J., 2009. Multipath Transformational Development of User Interfaces with Graph Transformations. In A. Seffah, J. Vanderdonckt, & M. Desmarais, eds. *Human-Centered Software Engineering*. Human-Computer Interaction Series. Springer London, p. 107–138.

Limbourg, Q. & Vanderdonckt, J., 2004. USIXML: A User Interface Description Language Supporting Multiple Levels of Independence. In *ICWE Workshops*. p. 325–338.

Lindgaard, G. et al., 2006. User Needs Analysis and requirements engineering: Theory and practice. *Interacting with Computers*, vol. 18, n°1, p.47 – 70.

Mannaro, K., Melis, M. & Marchesi, M., 2004. Empirical Analysis on the Satisfaction of IT Employees Comparing XP Practices with Other Software Development Methodologies. In J. Eckstein & H. Baumeister, eds. *Extreme Programming and Agile Processes in Software Engineering*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 166–174.

Martin, J., 1991. *Rapid application development*, Macmillan Publishing Co. Indianapolis, IN, USA, 788 p.

Martínez-Ruiz, T. *et al.*, 2013. Process variability management in global software development: a case study. In *proceedings of the International Conference on Software and System Process (ICSSP).* p. 46–55.

McDermid, J. & Ripken, K., 1984. *Life Cycle Support in the ADA Environment*, Cambridge University Press, New York, NY, USA.

Meeker, M. & Wu, L., 2013. *2013 Internet Trends*, Kleiner, Perkins, Caufield & Byers. En ligne : http://www.kpcb.com/insights/2013-internet-trends.

Messager Rota, V., 2009. *Gestion de projet: vers les méthodes agiles*, Eyrolles, 252 p.

Mills, H.D., 1971. Top-down programming in large systems. *Debugging Techniques in Large Systems*, p. 41–55.

Mirbel, I. & Ralyté, J., 2006. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, vol. 11, n°1, p.58–78.

Mulyar, N.A., van der Aalst, W.M.P. & Russell, N.C., 2007. *Process Flexibility Patterns*, Technische Universiteit Eindhoven, Eindhoven, Pays-Bas.

Myers, B.A., Hudson, S.E. & Pausch, R.F., 2000. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction*, vol. 7, n°1, p.3–28.

Neighbors, J.M., 1989. Software Reusability: Vol. 1, Concepts and Models. In T. J. Biggerstaff & A. J. Perlis, eds. New York, NY, USA: ACM, p. 295–319.

Newkirk, J. & Martin, R.C., 2001. *Extreme programming in practice*, Addison-Wesley.

Nguyen Van, Y., 2013. Le taux d'équipement de PC et de tablettes continue de grimper en France. *PC Expert*. En ligne: http://www.pcexpertlemag.fr/Actualites/Annonces/Le-taux-d-equipement-de-PC-et-de-tablettes-continue-de-grimper-en-France.

Nielsen, J., 1994. Usability Inspection Methods. In *proceedings of the ACM CHI Conference Companion on Human Factors in Computing Systems (CHI'94)*, New York, NY, USA, p. 413–414.

Nóbrega, L., Nunes, N. & Coelho, H., 2006. Mapping ConcurTaskTrees into UML 2.0. In S. Gilroy & M. Harrison eds. *Interactive Systems. Design, Specification, and Verification*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 237–248.

Noll, J., 2003. Flexible process enactment using low-fidelity models. In *Proceedings of the International Conference on Software Engineering and Applications (SEA 03)*. Marina del Rey, USA, M.H. Hamza eds, 6 p.

Norman, D.A. & Draper, S.W., 1986. *User centered system design: new perspectives on human-computer interaction*, Lawrence Erlbaum Associates, 526 p.

Normand, V., 1992. *Le modèle SIROCO : de la spécification conceptuelle des interfaces utilisateur à leur réalisation*. Thèse de l'Université Joseph Fourier-Grenoble I, Spécialité Informatique.

O'Reilly, M., 2003. *Extreme Programming Pocket Guide* 1st edition, Sebastopol, CA, USA: O'Reilly Media Inc., 108 p.

Object Management Group, 2011. OMG Unified Modeling Language (UML), Infrastructure, Version 2.4.1.

Object Management Group, 2008. *Software & Systems Process Engineering Metamodel (SPEM)*, OMG. 236 p.

OSGI Alliance, 2010. OSGi Service Platform Release 4. En ligne: http://www.osgi.org/Specifications/HomePage.

Osterweil, L.J., 1986. *Software Process Interpretation and Software Environments*, Technical report of the University of Colorado at Boulder, Department of Computer Science. 65 p.

Palmer, S.R. & Felsing, J.M., 2002. *A Practical Guide to Feature-Driven Development*, Prentice Hall.

Parnas, D.L., 1978. Designing Software for Ease of Extension and Contraction. In *Proceedings of the 3rd International Conference on Software Engineering*. ICSE '78. Piscataway, NJ, USA: IEEE Press, p. 264–277.

Paternò, F., 2000. Model-Based Approaches. In *Model-Based Design and Evaluation of Interactive Applications*. Applied Computing. Springer London, p. 11–30.

Paternò, F., Mancini, C. & Meniconi, S., 1997. ConcurTaskTrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*. p. 362–369.

Paternò, F. & Santoro, C., 2002. One Model, Many Interfaces. In *CADUI*. Valenciennes, France: Kluwer Academics Pub., p. 143–154.

Paulk, M.C. et al., 1993. Capability Maturity Model, Version 1.1. *IEEE Softw.*, 10(4), p.18–27.

Pérez, G., El Emam, K. & Madhavji, N., 1995. Customising software process models. In W. Schäfer, ed. *Software Process Technology*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 70–78.

Phalp, K. & Shepperd, M., 2000. Quantitative analysis of static models of processes. *Journal of Systems and Software*, vol. 52, n°2-3, p.105 – 112.

Pohl, K. et al., 1994. Applying AI Techniques to Requirements Engineering: The NATURE Prototype. In *Proceedings of the Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence*. ICSE.

Pohl, K., 2010. *Requirements Engineering: Fundamentals, Principles, and Techniques* 1st ed., Springer Publishing Company, Incorporated.

Poppendieck, M. & Poppendieck, T., 2003. *Lean Software Development: An Agile Toolkit*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Potts, C., 1989. A generic model for representing design methods. In *Proceedings of the 11th international conference on Software engineering*. ICSE '89. New York, NY, USA: ACM, p. 217–226.

Prakash, N., 1997. Towards a formal definition of methods. *Requirements Engineering*, vol. 2, n°. 1, p. 23–50.

Przesmycki, H., 2008. *La pédagogie différenciée*, Hachette Éducation. 160 p.

Ralyté, J., Deneckère, R. & Rolland, C., 2003. Towards a Generic Model for Situational Method Engineering. In J. Eder & M. Missikoff, eds. *Advanced Information Systems Engineering*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, p. 1029–1029.

Resnick, M. et al., 2005. Design Principles for Tools to Support Creative Thinking. *A Workshop Sponsored by the National Science Foundation*, p.25–35.

Rising, L. & Janoff, N.S., 2000. The Scrum software development process for small teams. *Software, IEEE*, vol. 17, n°4, p.26–32.

Rolland, C., 1998. A Comprehensive View of Process Engineering. In *Proceedings of the 10th International Conference on Advanced Information Systems Engineering*. London, UK: Springer-Verlag, p. 1–24.

Rolland, C., 2007. Capturing System Intentionality with Maps. In J. Krogstie, A. Opdahl, & S. Brinkkemper, eds. *Conceptual Modelling in Information Systems Engineering*. Springer Berlin Heidelberg, p. 141–158.

Rolland, C., 2005. L'ingénierie des méthodes : une visite guidée. *E-revue en Technologies de l'Information (e-TI)*, (1). En ligne : http://www.revue-eti.net/document.php?id=726.

Rolland, C., 1996. L'ingénierie des processus de développement de systèmes: un cadre de référence. *Ingénierie des systèmes d'information*, vol. 4, n°. 6, p. 705–744.

Rolland, C., Prakash, N. & Benjamen, A., 1999. A Multi-Model View of Process Modelling. *Requirements Engineering*, vol. 4, n°4, p.169–187.

Roques, P. & Vallée, F., 2002. *UML en action*, Paris, France: Eyrolles.

Rossi, M. et al., 2004. Managing Evolutionary Method Engineering by Method Rationale. *Journal of the Association for Information Systems*, vol. 5, n°. 9, p. 356–391.

Royce, W., 1990. Pragmatic quality metrics for evolutionary software development models. In *Proceedings of the conference on TRI-ADA '90*. TRI-Ada '90. New York, NY, USA: ACM, p. 551–565.

Royce, W.W., 1970. Managing the development of large software systems: concepts and techniques. *Proc. IEEE WESTCON, Reprinted in the Proceedings of the Ninth International Conference on Software Engineering, March 1987*, p. 328–338.

Schwaber, K., 1995. SCRUM Development Process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*. p. 117–134.

Selby, R.W., 2005. Enabling reuse-based software development of large-scale systems. *Software Engineering, IEEE Transactions on*, vol. 31, n°6, p.495–510.

Sharon, I. et al., 2010. A Decision Framework for Selecting a Suitable Software Development Process. In *ICEIS'10*. 12th International Conference on Enterprise Information Systems. Funchal, Madeira, Portugal, p. 34–43.

Sharp, H. et al., 2006. Agile Development: Opportunity or Fad? In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '06. New York, NY, USA: ACM, p. 32–35.

Smith, A. & Dunckley, L., 1998. Using the LUCID method to optimize the acceptability of shared interfaces. *Interacting with Computers*, vol. 9, n° 3, p.335–345.

Smith, D.P., 1971. *An approach to data description and conversion*, DTIC Document.

Sommerville, I., 1996. Software process models. *ACM Computing Surveys (CSUR)*, 28, p.269–271.

Sony-Ericsson, 2005. Communiqué de Presse. En ligne : www.sony.ru/res/attachment/file/45/1109586841545.pdf [Accès le 16/03/2014].

Sottet, J.-S. et al., 2007. A language perspective on the development of plastic multimodal user interfaces. *Journal on Multimodal User Interfaces*, vol. 1 (n°2), p.1–12.

Stapleton, J., 1999. DSDM: Dynamic Systems Development Method. In *Proceedings of the Technology of Object-Oriented Languages and Systems*. Washington, DC, USA: IEEE Computer Society, p. 406

Stolterman, E., 1992. How system designers think about design and methods. *Scandinavian Journal of Information Systems*, vol. 4, p. 137–150.

Stolterman, E., 1994. The « transfer of rationality »: Acceptability, adaptability and transparency of methods. In *Proceedings of the Second European Conference on Information Systems (ECIS 1994)*. Nijenrode University, The Netherlands, p. 533–542.

Szekely, P., 1996. Retrospective and Challenges for Model-Based Interface Development. In F. Bodart & J. Vanderdonckt, eds. *Design, Specification and Verification of Interactive Systems '96*. Eurographics. Springer Vienna, p. 1–27.

Tarby, J.-C. & Barthet, M.-F., 1996. The DIANE+ Method. In *Computer-Aided Design of User Interfaces, 2nd International Workshop on Computer-Aided Design of User Interfaces - CADUI'96*. p. 95–119.

Tardieu, H., Rochfeld, A. & Colletti, R., 1986. *La Méthode Merise, tome 1 : Principes et outils*, Paris, France: Les éditions d'organisation.

Tech-IS, 2012. Oxygen Code Generator. En ligne : http://www.oxygencode.com/ [Accès le 27/10/2012].

Thevenin, D., 2001. *Adaptation en Interaction Homme-Machine : le cas de la Plasticité*. Thèse de l'Université Joseph Fourier - Grenoble I. En ligne : http://iihm.imag.fr/thevenin/these/index.fr.html.

Thevenin, D. & Coutaz, J., 1999. Plasticity of User Interfaces : Framework and Research Agenda. In *Proceedings*. Interact99. Edinburgh: A. Sasse & C. Johnson Eds, IFIP IOS Press Publ., p. 110–117.

UsiXML Consortium, 2011. *UsiXML Method Specification*, 2011, 46 p.

UsiXML Consortium, 2012. *UsiXML Reference Manual*, 2012, 103 p.

Vanderdonckt, J., 2008. Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges. *Proceedings of ROCHI08*, p.1–10.

Williams, R.D., 1975. Managing the Development of Reliable Software. In *Proceedings of the International Conference on Reliable Software*. Los Angeles, California, p. 3–8.

Willig, C., 2001. *Introducing Qualitative Research in Psychology: Adventures in Theory and Method*, McGraw-Hill Companies,Incorporated.

Wistrand, K. & Karlsson, F., 2004. Method Components – Rationale Revealed. In A. Persson & J. Stirna, eds. *Advanced Information Systems Engineering*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 189–201.

Yourdon, E., 1999. *Death March: The Complete Software Developer's Guide to Surviving « Mission Impossible » Projects*, Prentice Hall PTR.

Yu, E., 1995. *Modelling Strategic Relationships for Process Reengineering*. Toronto: Department of Computer Science University of Toronto.

ANNEXES

ANNEXE 1 - PROCEDURE DE VALIDATION DU MODELE DE PROCESSUS

Dans le cas que nous présentons ici, les modifications ne concernent qu'une exécution du modèle de processus (on ne se préoccupe pas des autres projets en cours). A l'issue de ces modifications, le modèle de processus mis en œuvre sur le projet devient donc différent de celui des autres projets. L'outillage devra imposer de créer une nouvelle version du modèle de processus.

Ces procédures, qui concernent l'ajout et la suppression de buts, de stratégies, d'artefacts et d'activités, sont utilisées lors de l'exécution du modèle de processus : elles portent donc à la fois sur les éléments du modèle et sur leurs instances et il devient nécessaire de vérifier la cohérence globale du processus en prenant en compte les instances, et donc les états de ces instances. Par exemple, la possibilité de modifier une activité (et ses instances) n'est pas la même si l'instance d'activité n'est pas encore démarrée ou si elle est terminée : dans le cas où l'instance est terminée, l'activité ne peut pas être modifiée dans le processus concerné. M2Flex ne définit pas les statuts des instances d'activités : ces statuts sont spécifiés au niveau du modèle de processus. Dans les procédures que nous avons définies, nous avons considéré que les stratégies et les activités peuvent être « *proposée* », « *sélectionnée* », « *rejetée* », « *en cours* » ou « *terminée* ».

Par ailleurs, les procédures diffèrent selon que l'on veut modifier le modèle de processus pour l'une de ses instances (par exemple pour un projet donné) ou pour toutes ses instances : dans ce dernier cas, les contrôles décrits ci-dessous doivent prendre en compte toutes les instances de tous les éléments de toutes les instances du modèle de processus, autrement dit prendre en compte toutes les exécutions du modèle de processus.

- **Ajout d'un but**
  - Créer le but (et son instance)
  - Créer au moins une stratégie ayant ce but comme but source (voir *Ajout d'une stratégie*)
  - Créer au moins une stratégie ayant ce but comme but cible(voir *Ajout d'une stratégie*)
- **Suppression d'un but**
  - Vérifier qu'aucune instance de stratégie ayant ce but comme cible ou comme source n'est au statut « *en cours* » ou « *terminée* »
  - Vérifier qu'il reste un chemin allant du but de départ au but de fin sans les stratégies ayant ce but comme source ou comme cible
  - Supprimer toutes les stratégies (et leurs instances) ayant ce but comme cible (voir *Suppression d'une stratégie*)

    - Supprimer toutes les stratégies (et leurs instances) ayant ce but comme source (voir *Suppression d'une stratégie*)
- **Ajout d'une stratégie**
    - Créer la stratégie
    - Créer au moins une activité qui concrétise la stratégie (voir *Créer une activité*)
    - S'il existe une stratégie ayant le même but source ou le même but cible que la nouvelle stratégie, et dont le statut est « *sélectionnée* », « *en cours* » ou « *terminée* », mettre l'instance de la nouvelle stratégie au statut « *rejetée* », sinon mettre l'instance de la stratégie au statut « *proposée* »
- **Suppression d'une stratégie**
    - Vérifier que l'instance de la stratégie n'est pas au statut « *en cours* » ou « *terminée* »
    - Vérifier qu'il reste un chemin allant du but de départ au but de fin sans cette stratégie
    - Si l'instance de la stratégie est au statut « *sélectionnée* » : mettre au statut « *proposée* » toutes les instances de stratégies qui ont le même but cible ou le même but source que la stratégie à supprimer
    - Supprimer tous les artefacts qui ne sont utilisés que par des activités de la stratégie (voir *Supprimer un artefact*)
    - Supprimer tous les rôles qui ne sont associés qu'à des activités de la stratégie (voir *Supprimer un rôle*)
    - Supprimer toutes les activités de la stratégie (voir *Supprimer une activité*)
    - Supprimer la stratégie
- **Ajout d'un artefact**
    - Créer l'artefact
    - Associer l'artefact comme produit d'au moins une activité dont l'instance a un statut différent de « *terminée* »
    - Associer l'artefact comme entrée d'au moins une activité dont l'instance a un statut différent de « *terminée* »
- **Suppression d'un artefact**
    - Vérifier qu'il n'existe pas d'instance d'activité produisant cet artefact et ayant un statut « *en cours* » ou « *terminée* »
    - Vérifier qu'il reste un chemin possible allant du but de départ au but d'arrivée sans les activités qui utilisent l'un des statuts de l'artefact
    - Vérifier qu'il reste un chemin possible allant du but de départ au but d'arrivée sans les activités qui ne produisent que cet artefact
    - Supprimer les activités (et leurs instances) qui ne consomment que cet artefact
    - Supprimer l'artefact de la liste des produits des activités, et si c'est le seul produit d'une activité, supprimer l'activité et son instance

    - Supprimer l'artefact
- **Ajout d'une activité**
    - Créer l'activité
    - L'associer
        - soit comme concrétisation d'une stratégie ; mettre l'instance de l'activité au même statut que l'instance de la stratégie ;
        - soit comme raffinage d'une autre activité ; mettre l'instance de la nouvelle activité au même statut que l'instance de l'activité qu'elle raffine ;
        - soit comme reformulation d'une autre activité mettre l'instance de la nouvelle activité au même statut que l'instance de l'activité qu'elle reformule ;
        - soit comme cible d'un opérateur de choix ; si l'opérateur est au statut « *terminé* », l'instance de l'activité est au statut « *rejetée* », sinon elle est au statut « *proposée* »
        - soit comme cible d'un opérateur d'activation ; l'instance de l'activité est alors au statut « *proposée* »
        - soit comme cible d'un opérateur de désactivation ; l'instance de l'activité est alors au statut « *proposée* » si l'instance de l'activité source est « *rejetée* » ou « *proposée* », et « *rejetée* » si l'instance de l'activité source est « *terminée* »
        - soit comme cible d'un opérateur de parallélisation ; l'instance de l'activité est au statut « *proposée* »
        - soit comme consommateur et/ou producteur d'artefacts, en vérifiant qu'elle ne produise pas d'artefact dont le statut est déjà « *terminé* »
- **Suppression d'une activité**
    - Vérifier que l'instance de l'activité n'est pas au statut « *en cours* » ou « *terminée* »
    - Pour chacun des artefacts produit par l'activité :
        - si l'artefact est utilisé en entrée par une autre activité non-optionnelle
            - Si aucune une autre activité ne produit le même artefact au même statut, rejeter la demande de suppression
            - Si au moins une autre activité produit le même artefact au même statut, et que son instance est au statut « *rejetée* », rejeter la demande de suppression
        - si aucune autre activité ne produit cet artefact à ce statut, supprimer le statut de l'artefact, ainsi que son lien avec l'activité à supprimer

        - si une autre activité produit le même artefact au même statut, supprimer uniquement le lien entre le statut et l'activité à supprimer
    - Supprimer l'activité

RÉSUMÉ

La diversité des dispositifs et les exigences des utilisateurs en termes de disponibilité et de continuité de service complexifient l'ingénierie de l'interaction homme-machine : il devient nécessaire de créer des IHM douées d'adaptation dynamique à leur contexte d'usage. L'ingénierie de ces IHM, dites plastiques, peut suivre une approche dirigée par les modèles mais ces approches sont encore peu pratiquées et souffrent d'un coût d'apprentissage important. Il est donc impératif d'accompagner les concepteurs et développeurs par un guidage, mais ce guidage doit être suffisamment flexible pour intégrer des compétences variées et des pratiques diverses en constante évolution.

L'ingénierie des méthodes de développement logiciel s'est depuis longtemps préoccupée de la flexibilité des modèles de processus pendant leur conception, mais très peu de travaux se sont préoccupés de la flexibilité à l'exécution. Pourtant, plusieurs études montrent que les concepteurs et les développeurs, qui sont les principaux utilisateurs des méthodes, en expriment le besoin. Ils souhaitent par exemple disposer de modèles de processus exprimés dans les langages qu'ils maîtrisent, qui les laissent maîtres des choix de conception ou de réalisation et les aident dans l'apprentissage de la démarche. La flexibilité des modèles de processus à l'exécution, telle que nous la proposons, permet de répondre à ces attentes et ouvre donc la possibilité de fournir un guidage adéquat pour le développement d'IHM plastiques.

Nous nous sommes focalisés dans un premier temps sur la conceptualisation de la propriété de flexibilité. Cette étude nous a conduits à proposer une taxonomie des modèles de processus, Promote, qui définit et gradue la flexibilité selon six dimensions. Nous avons ensuite transcrit cette définition de la flexibilité dans un métamodèle de processus flexible, M2Flex, et l'avons implémenté dans deux outils : D2Flex (D pour Design time), un outil collaboratif de conception de modèles de processus, et R2Flex (R pour Runtime), un environnement d'exécution des modèles définis dans D2Flex. Nous avons appliqué notre approche aux modèles de processus de développement d'IHM plastiques en rendant flexible la méthode UsiXML. L'environnement logiciel est en maturation technologique pour un transfert vers l'industrie. Ces différentes contributions ont fait l'objet de validations, en particulier auprès de concepteurs novices, en ingénierie de l'interaction homme-machine et des systèmes d'information.

MOTS CLÉS : Méthodes, Modèles de processus, Flexibilité, IHM plastiques, Ingénierie Dirigée par les Modèles