Université Paris Ouest Nanterre La Défense

Master II MIAGE

Méthodes Informatiques Appliquées à la Gestion des Entreprise

Étude des variations par rapport à la prévision et son analyse dans la gestion des projets informatiques

Auteur :
KABINE KABA

Encadrant :
M. Pascal Poizat

Sujet de recherche pour l'obtention d'un Master MIAGE à

l'UFR SEGMI

19 juillet 2015

Remerciements

Les remerciements.....

Résumé

Le résumé

Abstract

Partie abstract

Table des matières

Liste des figures

Liste des tableaux

Abréviations

Chapitre 1

Introduction

Dans le cadre de mon Master II MIAGE, chaque étudiant devait trouver et traiter un sujet de niveau « Master II MIAGE » qui présente une problématique réelle. Pour cela je me suis intéressé à un sujet particulier : « La gestion de la variation par rapport à la prévision », car je suis très intéressé par la gestion des projets. L'objectif de ce travail de recherche est de proposer un modèle capable de détecter les variations par rapport à la prévision lors de la réalisation d'un projet de développement informatique. Alors, ce document présente le travail que j'ai pu effectué par rapport à ce sujet : de la problématique jusqu'à la conclusion. Dans ce premier chapitre nous allons expliquer la problématique, détailler les objectifs et présenter la structure du document.

1.1 Problématique

Répondre urgemment au besoin de produire un logiciel de très bonne qualité est l'objectif principal du génie logiciel. Bien qu'il existe plusieurs documents traitant les facteurs de qualités d'un logiciel, l'évaluation de la qualité d'un logiciel n'est pas aussi simple que cela puisse paraître [6]. Il existe une forte corrélation entre la qualité du processus de développement et la qualité des logiciels développés [7]. Par conséquent nous pouvons avoir plus de contrôle sur la qualité des produits en contrôlant le processus logiciel.
Depuis quelques années, nous assistons à une croissance des projets de développements logiciels [8]. Avec une estimation de 9.05 milliards de dollars en 2012, d'après Gartner le marché du développement logiciel devrait atteindre 10.28 milliards de dollars en 2016 [9].

Cependant, nous assistons également à une forte augmentation du taux d'échec dans les projets informatiques.

Selon [10] un projet peut être considérer comme réussi lorsqu'à sa date de mise à disposition au client, les trois critères : performance, coûts et délai sont conformes aux objectifs contractuels de démarrage. Malheureusement cette situation ne se réalise pas toujours. Comme le montre une autre étude effectuée en 2012, 31% des projets sont abandonnés avant leur terme, 88% dépassent les délais, le budget ou les deux et encore la proportion de dépassement de ces délais est très surprenante 189% pour le dépassement de budget et 222% pour celui des délais [11]. La plupart de ces échecs sont souvent dus à des variations pouvant subvenir lors de la réalisation du projet. Une variation peut être définie comme étant une action effectuée durant le processus de développement d'un logiciel mais qui est incohérent avec le processus mise en place.

Pendant la réalisation d'un projet de développement logiciel, on définit les différentes séquences ou phases de développement du logiciel. Ces séquences ou phases souvent appelés cycle de vie du produit est le processus de développement logiciel ou SPM *(Software Process Model)* [12]. Pour décrire les différentes phases, on peut utiliser un langage de modélisation des processus appelé *Process Modeling Langage(PML)*.

Une fois les processus modélisés, les agents pourront suivre les étapes définis pour réaliser le produit (logiciel). L'erreur étant humaine, les agents ne sont pas l'abri d'en commettre durant les processus d'exécutions. Face à ce problème, les entreprises ont jugé nécessaire d'automatiser ces actions avec l'aide des environnements de développement logiciel centrés procédés *(PSEE – Process-centered Software Execution Environment)*. Les PSEE ont pour objectif de s'assurer que les processus misent en place sont bien suivis par les agents [13]. La plupart des PSEE existant ne sont pas à mesure de gérer proprement les variations qui ont lieu durant l'évolution du projet, ils ne sont pas assez flexible pour être adopter dans le milieu industriel [14].

1.2 Objectif

L'objectif de ce travail de recherche peut être divisé en deux grandes parties.

- État de l'art :

  dans cette partie nous allons étudier les PSEEs actuels : leurs architectures,leurs méthodes de détections de variations et les options qu'ils proposent pour gérer ces variations.

- Solutions :

  après l'étude des solutions existantes, cette deuxième partie sera consacrée à la proposition d'une solution pour répondre à la problématique. La solution sera composé d'un modèle capable de supporter les variations, et de la description d'une méthode de détection et de correction des variations détectées.

1.3 Structure du document

Ce document est composé de quatre chapitres :

1. le **chapitre I** est **l'introduction**. Dans ce chapitre, nous allons définir les motivations par rapport à ce sujet ainsi que les objectifs attendus.
2. le **chapitre II** sera consacré à **l'étude de l'existant** : l'architecture, les problématiques qu'ils répondent, etc.
3. le **chapitre III** sera **la gestion des variations**. Nous présenterons notre modèle ainsi que notre technique de détection et de correction des variations.
4. le **chapitre IV** sera **la conclusion**. Nous présenterons un bilan de ce travail par rapport à l'objectif et aborderons les perspectives de ce travail.

Chapitre 2

État de l'art

Plusieurs travaux par rapport à la gestion des variations ont été déjà effectués. Comme par exemple les travaux de [15] [3] [16] [17], etc. Ce qui veut dire que des solutions ont déjà été proposées. Alors dans ce chapitre, nous allons parler de la modélisation des procédés ainsi que les solutions qui traitent les variations au cours de l'évolution du projet.

2.1 Modélisation des procédés

Le développement d'un logiciel suit au moins ce qu'on appelle un procédé. Ce procédé définit toutes les étapes nécessaires pour sortir un produit réussi. Aujourd'hui, la qualité de la modélisation des procédés occupe une place importante dans la réalisation d'un logiciel car la réussite du produit en dépend [18] [19].
On peut alors définir un procédé logiciel comme un ensemble d'activités aussi techniques qu'administratives pour développer et maintenir un produit logiciel [20].

2.1.1 Modèle de procédé

Le modèle de procédé est une abstraction du procédé réel. Il décrit les éléments de procédé et les relations entre eux. Il y a plusieurs discussions par rapport aux éléments

à décrire dans les modèles de procédés. Malgré cette divergence, il existe des points communs qui qui ont été recueillis dans plusieurs études [20] [19] [21] [22].
Han Nhi Tran [1] propose une classification de ces éléments en deux catégories.

1. Éléments primaires

   Ils représentent le cœur des procédés. Ces éléments ne tiennent pas compte des aspects de planification et d'exécution de procédé. Ce sont :

   - les activités : L'ensemble des actions effectuées par les rôles pour accomplir un objectif dans le développement [23].
   - les produits : ce sont des artéfacts [1] utilisés ou élaborés par les activités durant le développement.
   - les rôles : un rôle décrit un ensemble de responsabilités, droits et compétences d'un agent dans le contexte des activités ou il intervient [23].

2. Éléments secondaires

   Ce sont des éléments fournissant des informations supplémentaires pour la mise en œuvre d'un procédé. Ces éléments sont :

   - les agents : un agent est celui qui exécute le procédé [2]. Il peut être un humain ou un outil logiciel. Un agent n'est pas un rôle, mais il est caractérisé par les propriétés de son ou ses rôle(s) [23]
   - les ressources : ce sont des éléments facilitant l'exécution d'une activité [1]
   - les informations qualitatives : ce sont des informations permettant d'évaluer la performance et la qualité de procédés. Ces informations peuvent être des résultats de révision ou test, des métriques etc. [1]
   - les informations organisationnelles : ce sont des informations qui facilitent l'exécution d'un procédé pour un projet spécifique

1. les artéfacts sont des produits créés ou modifiés pendant un procédé
2. tous les éléments impliquées dans le développement et la maintenance d'un produit ou service, c'est à dire artefacts, support de production (outils), activités agents

Le modèle conceptuel de procédés (Fig. 2.1), nous montre les différentes relations existantes entre les éléments de procédés.

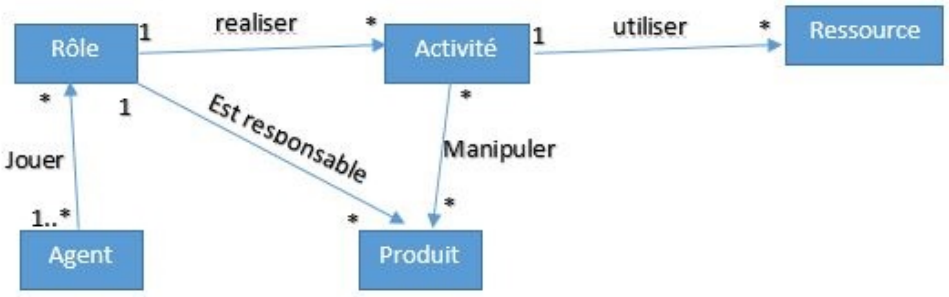

FIGURE 2.1: Modèle conceptuel de procédés [1]

2.1.2 Langage de modélisation de procédés

Pour décrire un modèle de procédé, nous avons besoin de langage, ce langage est appelé « Langage de modélisation de procédés » ou *"Process Modeling Language (PML)"* [21] [24] [25] ou encore *"Software Process Modeling Language (SPML)"* [3].
Ils existent plusieurs langages de modélisation de procédés logiciel, mais d'après les études de [19],[26] [27], les propriétés les plus importantes attendues d'un PML sont :

1. Formalisation : cette propriété représente le degré de formalisation (définition syntaxique) et de la sémantique du PML. Il existe trois degrés de formalisation : formel [3], informel [4], semi-formel [5].
2. Expressivité : cette catégorie reflète la capacité du PML à représenter tous les éléments du procédé.
3. Compréhensibilité : elle peut être textuel ou graphique, cette propriété reflète le degré de facilité à comprendre le modèle décrit à travers la notation du PML.
4. Abstraction et modularité : elles représentent la capacité du PML à proposer des mécanismes d'abstraction et d'agrégation afin de structurer les procédés pour une facilitation de la réutilisation.
5. Exécutabilité : c'est la capacité du PML à représenter des modèles exécutables (opérationnels).

3. un langage formel est un langage dont la sémantique et la syntaxe sont bien définis
4. langage non défini avec des concepts, une syntaxe et une sémantique
5. langage dont la syntaxe est précise mais la sémantique non

6. Évolutivité : c'est la capacité de supporter l'évolution de modèles de procédé.
7. Multi-vue : la capacité à supporter les modèles d'activités, produits, rôles, ressources

En étudiant les PMLs, on s'aperçoit qu'ils peuvent utilisés différentes approches afin de mieux répondre aux besoins de la phase. Nous n'allons pas citer toutes les approches existantes, mais nous présenterons quelques une des plus connues. Parmi ces approches nous avons :

- approche procédurale : proposée dans [28], l'approche procédurale permet de représenter le modèle de procédé sous la forme d'un programme. Ce programme décrit de manière détaillé comment le procédé logiciel doit être réalisé.
- approche déclarative : cette approche utilise des déclarations logiques (règles) pour décrire les procédés logiciels. Ceux-ci sont décrits en termes de résultats attendus par l'utilisateur sans détailler la manière dont ces résultats sont obtenus [29].
- approche fonctionnelle : l'approche fonctionnelle définit le procédé logiciel à travers un ensemble de fonctions mathématiques. Chaque fonction est décrite en termes de relations entre les données d'entrée et les données de sortie.
- approche basée sur les réseaux de pétri : cette approche permet de décrire le procédé logiciel à travers un réseau de pétri [6].
- approche basée sur UML : cette approche utilise des diagrammes UML pour représenter les concepts du procédé et renforce la sémantique de ces diagrammes avec un langage formel pour rendre les modèles de procédé à la fois compréhensifs et exécutables. Plusieurs approches ont été développées avec cette perspective, notamment le standard de l'OMG (société Américaine créée en 1989, il est à la base des standards UML, MOF *"Meta-Oject Facility"*, CORBA *"Common Object Request Broker Architecture"*, CWM *"Common Warehouse Metamodel"*) *"Object Management Group"* SPEM.

***Software Process Engineering Metamodel* (SPEM)**

le méta-modèle d'ingénierie des procédés logiciels est à la fois un méta-modèle (c'est un langage de modélisation permettant d'exprimer les concepts conforme au MOF) et un profil UML (il est basé sur du UML). Le leitmotiv d'un SPEM

6. un réseau de pétri est un moyen de : - modélisation du comportement des systèmes dynamiques à évènements discrets ; - description des relations existantes entre des conditions et des évènements

est qu'un procédé de développement logiciel est une collaboration entre des rôles exécutant des actives sur des produits.

Il existe deux versions de SPEM : **SPEM 1.1** et **SPEM 2.0**.

- SPEM 1.1 : première version de SPEM, adopté en janvier 2005 [30], il était composé de deux groupes : le *SPEM foundation* qui fournit les concepts de base pour définir le méta-model SPEM et le *SPEM-Extension* qui décrit le concept de base pur décrire les processus logiciels.
- SPEM 2.0 : cette version est la plus récente de SPEM. Le SPEM 2.0 est structuré en sept paquetages [2] comme le montre la figure 2.2.

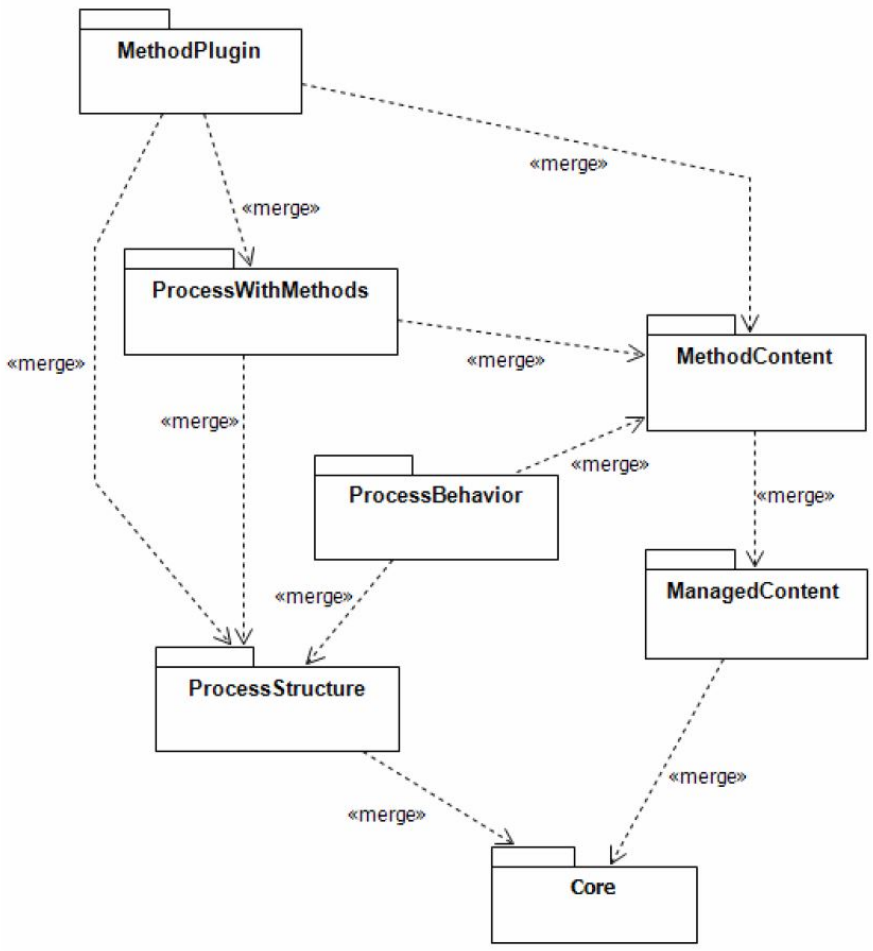

FIGURE 2.2: Structure du SPEM 2.0 [2]

- ***MethodPlugin*** : dans ce paquetage, seront gérés les librairies et les référentiels méthodologiques.
- ***ProcessWithMethods*** : ce paquetage permet de définir ou de redéfinir les éléments pour intégrer les processus définis avec les concepts du paquetage *ProcessStructure*, selon les contenus définis avec les concepts du paquetage *MethodContent* ;
- ***MethodContent*** : ce paquetage fournit les concepts pour que les utilisateurs puissent spécifier le contenu des méthodes comme les rôles *(Roles)*, les tâches *(Tasks)*, etc. ;
- ***ProcessBehavior*** : il offre la possibilité de lier un élément de procédé SPEM 2.0 avec un comportement externe comme un modèle BPMN *(Business Process Modeling Notation)* ou un modèle UML 2.
- ***ManagedContent*** : ce paquetage fournit les concepts pour gérer les descriptions textuelles des éléments de procédés.

- ***ProcessStructure*** : ce paquetage définit la base pour représenter tous les modèles de procédés.
- ***Core*** : il contient les éléments de base (classes et abstractions) utilisés dans les autres paquetages.

Après la présentation des modèles de procédés et des langages de modélisation, nous allons passer aux environnements de génie logiciel centrés procédé.

2.2 Process-Centered Software Engineering Environments (PSEE)

Un PSEE est un environnement de travail permettant de fournir divers services aux développeurs par l'exécution des modèles de processus. Ces services sont multiples, parmi lesquels nous avons : l'aide aux développeurs logiciel, l'automatisation de certaines tâches, l'invocation et le contrôle des outils de développement logiciel, le contrôle de l'application des règles obligatoires, etc. [26].

La figure 2.3 résume le rôle d'un PSEE dans le processus de développement d'un logiciel.

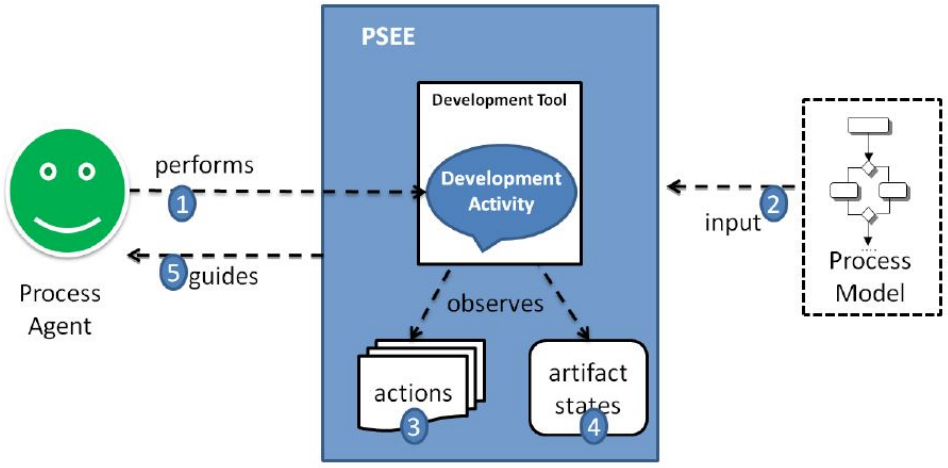

FIGURE 2.3: Rôle d'un PSEE [3]

Le PSEE prend en entrée le modèle de procédé et permet à l'agent de réaliser l'activité de « développement de logiciel ». Dans le but de s'assurer que les activités en cours respectent le modèle de processus définit, le PSEE observe les actions effectuées par les agents. Si les agents ne suivent pas les processus définis, le PSEE doit les guider dans la bonne direction. Le fait d'exécuter une action ne respectant pas les règles décrit dans le modèle de procédé est appelée « **variation** » ou « **déviation** ». Il existe plusieurs PSEE, mais la plupart ne supportent pas la notion de la gestion de variation dans le processus de développement du logiciel. Dans cette partie, nous allons faire une présentation de certains PSEE et également la manière dont ils gèrent les variations par rapport au prévisionnel.

EPOS

Développé à l'université norvégienne de sciences et de technologie, EPOS est un PSEE assez complet pour la gestion des configurations et la modélisation des procédés logiciels. Il comporte un ensemble d'outils de procédés et utilise SPELL (Software Process Evolutionary Language) comme PML. Dans EPOS, l'élément principal est un système de base de données « EPOSDB » dans laquelle sont stockées les objets et les instances de procédé [31]. Le modèle de procédé logiciel dans EPOS est décrit comme étant un réseau de description de tâches ou d'activités lié entre elles.

L'architecture d'EPOS est composée de plusieurs couches [32] qui sont :

1. EPOSDB, celle ci constituant le système de base de données, offre un modèle orientée objet de donnée [7] pour définir les entités et la relation entre les classes. Les relations entre chaque entité avec leurs classes et méta-classes sont stockées dans la base de données.
2. Un PML réfléchi [8] et orienté objet, cette couche permet de supporter l'analyse, la conception, la personnalisation et l'évolution des méta-activités.
3. Une couche pour gérer la concurrence entre les modèles de procédés logiciel.
4. Une dernière couche pour les modèles de procédé spécifique à l'application. C'est les domaines spécifiques avec les schémas.

A chaque tâche est associée des pré et post-conditions, les pré-conditions sont eux même divisées en deux parties : une partie statique, utilisé à l'instanciation de la tâche ; et

7. les informations sont stockées sous forme de collections d'objets persistants
8. La réflexion est la capacité à un programme d'examiner et de contrôler son propre implémentation

une partie dynamique qui est testée par le gestionnaire d'exécution au moment de son exécution [31]. EPOS est capable de détecter quelques variations liées à la structure et à l'organisation, mais il est incapable de faire une détection le plus rapidement possible et ne propose pas non plus de risque associé à la variation détectée [33].

RHODES

Développé à l'ENSEEIHT-IRIT à Toulouse, RHODES est un PSEE utilisant le langage PBOOL+ comme PML ; Le PBOOL+ permet de définir les composants (activités, rôles, produits et stratégies) des processus et les comportements attendus [4]. Comme on le voit sur la figure 2.4 l'architecture de RHODES est composée de deux niveaux : statique et dynamique. Le niveau statique se compose : du méta-modèle, de la base de composant de procédé, du composant d'éditeur de procédé, du noyau d'exécution et du compilateur PBOOL+. Le niveau dynamique est constitué d'une instance exécutable de procédé. Le

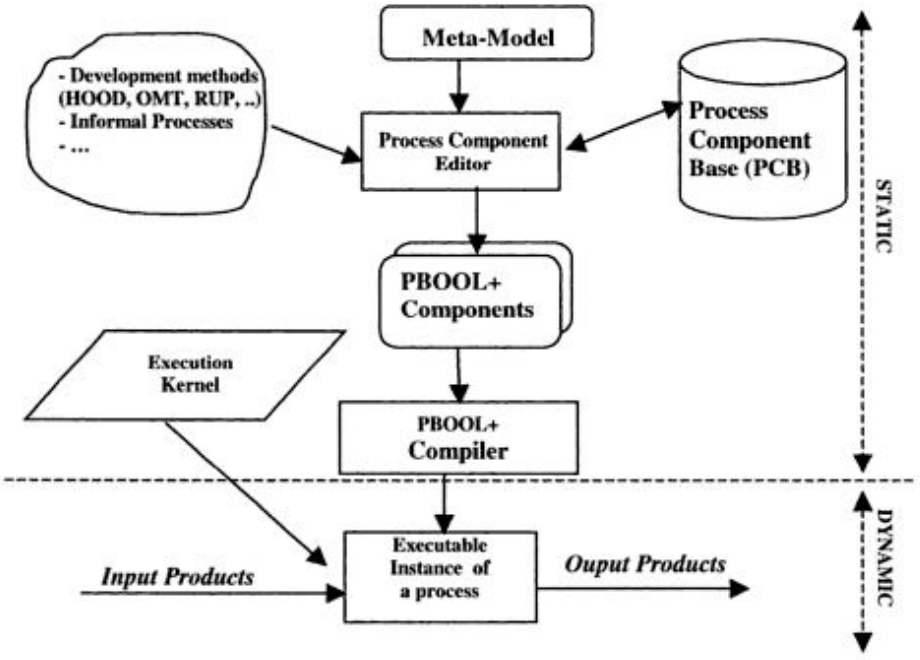

FIGURE 2.4: Architecture de RHODES [4]

fait de pouvoir définir comment l'activité est supposée être exécutée grâce à PBOOL+, permet à RHODES de détecter la plupart des variations car si on est capable de définir la manière d'exécution ou le comportement attendu d'une activité, il suffit après de faire une comparaison pour savoir si oui ou non nos conditions ont été respectées. En revanche

il ne présente pas de mécanisme capable d'estimer ou d'évaluer le risque associé à une variation [34].

SPADE

L'acronyme de *Software Process Analysis, Design and Enactment*, SPADE est un PSEE développé à Politecnico di Milano et CEFRIEL. Son PML appelé SLANG est basé sur une extension des réseaux de pétri : ER nets [35] [36]. L'architecture de SPADE est centrée sur le principe de séparation entre l'interprétation des modèles de procédé et l'interaction utilisateur.

SPADE est composé de trois couches [35] :

1. l'environnement d'exécution des procédés, dans cet environnement sont exécutés nos procédés décrits avec SLANG. Les artéfacts crées ou modifiés sont stockés dans une base de données orientées objet DBMS.
2. l'environnement d'interaction avec l'utilisateur, dans cet environnement est géré tout ce qui est interaction entre l'utilisateur et les outils dans l'environnement.
3. Filtre, il est l'intermédiaire entre l'environnement d'exécution des procédés et celui d'interaction utilisateur. Son rôle est de convertir les messages générés depuis l'environnement d'interaction utilisateur, les transformer en jetons pour qu'ils puissent être exécutés dans l'environnement d'exécution. Et inversement il transforme les résultats des travaux effectués par l'interpréteur SLANG et les convertissent en message à destination de l'utilisateur.

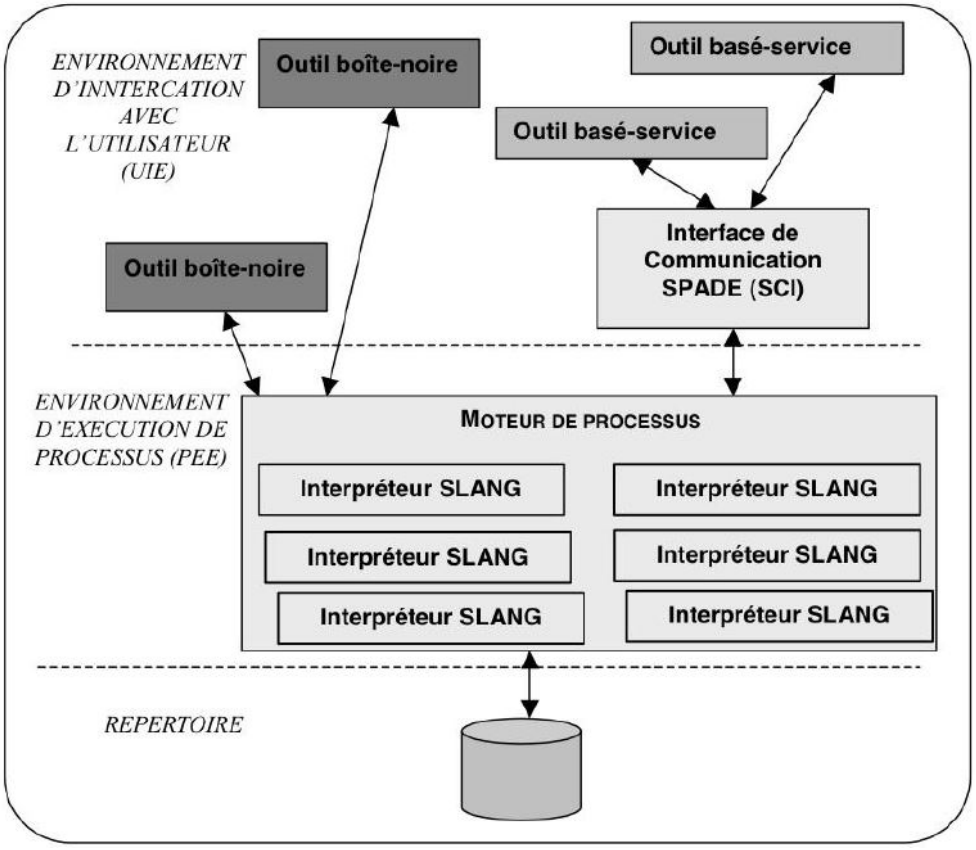

FIGURE 2.5: Architecture de SPADE [5]

Chapitre 3

Gestion des variations

Dans cette partie on décrira notre modèle de détection et de correction par rapport à la variation.

Chapitre 4

Conclusion

Conclusion

Bibliographie

[1] Hanh Nhi Tran. *Modélisation de procédés logiciels a base de patrons reutilisable*, pages 10–11. 2010.

[2] OMG. Software process engeneering metamodel specification spem 2.0. 2005. URL http://doc.omg.org/formal/08-04-02.pdf.

[3] Marcos Aurélio Almeida da Silva. *Detection and Handling of Deviations in Process-Centered Software Engineering Environments*. April 2012.

[4] S. Bernadette F. Joaquim and M. Paula. *Enterprise Information Systems III*, page 181. Kluwer Academic Publishers, 2002. URL https://books.google.fr/books?id=AvtEClDmY3cC&pg=PA181&lpg=PA181&dq=coulette+et+al+rhodes&source=bl&ots=wuPasAF7tx&sig=OoHYaL3SaLaHXndmQh9kOUd3AX8&hl=fr&sa=X&ei=XUaWVfaSKYfkyAPt9YMg&ved=0CCsQ6AEwAQ#v=onepage&q=coulette%20et%20al%20rhodes&f=false.

[5] MOHAMMED ISSAM KABBAJ. *Gestion des déviations dans la mise en oeuvre des procédés logiciel*, pages 57–58. Octobre 2009. URL ftp://www.irit.fr/IRIT/MACAO/These_Kabbaj_finale_25-10-2009.pdf.

[6] Wikipedia. Assurance qualité logicielle. Avril 2015. URL https://fr.wikipedia.org/wiki/Assurance_qualit%C3%A9_logicielle.

[7] Wikipedia. Architecture logicielle. URL http://fr.wikipedia.org/wiki/Architecture_logicielle.

[8] Journal du net. Marché des logiciels pro en 2013 : dépenses et chiffre d'affaires. 2013. URL http://www.journaldunet.com/solutions/saas-logiciel/marche-du-logiciel.shtml.

[9] Alain Clapaud. Logiciels de développement : une industrie de 9 milliards de dollars. 2012. URL http://pro.01net.com/editorial/571595/logiciels-de-developpement-une-industrie-de-9-milliards-de-dollars/.

[10] Roger AIM. La gestion de projet. pages 7–9, 2014.

[11] CHOB. Gestion de projet : pourquoi ça plante ? 2012. URL http://www.choblab.com/gestion-projets/gestion-de-projet-pourquoi-ca-plante-6244.html.

[12] The software experts. What is software process model ? URL http://www.the-software-experts.com/e_dta-sw-process-model.php.

[13] Marcos Aurélio Almeida da Silva. *Detection and Handling of Deviations in Process-Centered Software Engineering Environments*, page 3. 2012.

[14] MOHAMMED ISSAM KABBAJ. *Gestion des déviations dans la mise en oeuvre des procédés logiciel*, pages 20–21. Octobre 2009. URL ftp://www.irit.fr/IRIT/MACAO/These_Kabbaj_finale_25-10-2009.pdf.

[15] C Ghezzi S Bandinelli, A Fuggetta and L Lavazza. *SPADE : An Environment for Software Process Analysis, Design and Enactment*. 1994.

[16] MOHAMMED ISSAM KABBAJ. *Gestion des déviations dans la mise en oeuvre des procédés logiciel*. Octobre 2009. URL ftp://www.irit.fr/IRIT/MACAO/These_Kabbaj_finale_25-10-2009.pdf.

[17] Gianpaolo Cugola. *Tolerating deviations in process support systems via flexible enactment of process models*. 1998.

[18] Watts S. Humphrey. *Characterizing the software process : a maturity framework*, pages 73–79. 1988.

[19] C Ghezzi P Armenise, S Bandinelli and A Morzenti. *A Survey and assesment the process representing formalisms*. 1993.

[20] Jacques Lonchamp. *A structured conceptual and terminological framework for software process engineering*, pages 41–53. 1993.

[21] Silvia T. Acuña and Xavier Ferre. *Software Process Modelling*. 2001.

[22] J Kramer A Finkelsteiin and B Nuseibeh. *Software Process Modelling and Technology*. 1994.

[23] Lizbeth Gallardo. *Une approche à base de composants pour la modélisation des procédés logiciels*, pages 10–11. 2000.

[24] Reidar Conradi and Chnnian Liu. *Revised PMLs and PSEE for Industrial SPI.* 1997.

[25] SM Sutton and LJ Osterweil. *The Design of a Next-Generation Process Language.* 1997.

[26] R Conradi V Ambriola and A Fuggetta. *Assessing process-centered software engineering environments.* 1997.

[27] A. B. Kaba JC Derniame and Wastell. *Software Process : Principles, Methodology, Technology.* 1999.

[28] Leon Osterweil. *Software Processes are Software Too.* 1987.

[29] LIoyd G. Williams. *Software process modeling : a behavioral approach.* 1988.

[30] OMG. Software process engeneering metamodel specification spem 1.1. 2005. URL http://doc.omg.org/formal/05-01-06.pdf.

[31] L. Jens-Otto J. Letizia and C. Reidar. Software process modeling and evolution in epos. pages 1–4, December 1999. URL http://www.idi.ntnu.no/grupper/su/publ/pdf/capri-final.pdf.

[32] J. Letizia and C. Reidar. Technique for process model evolution in epos. pages 5–8, May 1993. URL https://static.aminer.org/pdf/PDF/001/128/476/techniques_for_process_model_evolution_in_epos.pdf.

[33] Marcos Aurelio Almeida da Silva. Detection and handling of deviations in process-centered software engineering environments. pages 22–24, April 2012.

[34] Marcos Aurélio Almeida da Silva. *Detection and Handling of Deviations in Process-Centered Software Engineering Environments.* April 2012.

[35] Brian C. Warboys. *Software Process Technology*, pages 15–29. Springer, 1994. URL https://books.google.fr/books?id=XNFG2Y7TVQgC&pg=PA16&lpg=PA16&dq=SPADE:+An+Environment+for+Software+Process+Analysis,+Design+and+Enactment&source=bl&ots=Opl2GXZwSe&sig=b3dfPvn7BDkQc_LaaJi2XZfFZuU&

hl=fr&sa=X&ei=Q_acVZuwB-GeygPizJPoCg&ved=0CF4Q6AEwBw#v=onepage&q=RHODES&f=false.

[36] A. B. Kaba JC Derniame and D. Wastell. *Software Process : Principles, Methodology, Technology*, page 42. Springer, 1999. URL https://books.google.fr/books?id=YH5oXL8CK28C&pg=PA285&lpg=PA285&dq=SPADE:+An+Environment+for+Software+Process+Analysis,+Design+and+Enactment&source=bl&ots=X4XXxVPYfy&sig=uEi2LoURPzzLeFtnrLc-1xruVb0&hl=fr&sa=X&ei=Q_acVZuwB-GeygPizJPoCg&ved=0CGMQ6AEwCA#v=onepage&q=SPADE&f=false.