

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

31 août 2015

Remerciements

Je tiens à remercier mon encadrant M. Pascal POIZAT pour tout ses conseils pendant la rédaction de ce mémoire.

Je remercie également Ibrahim Boubacar Dicko Chef de projet chez Alstom qui m'a fourni beaucoup d'information par rapport à la gestion des projets chez Alstom, je remercie également toute ma famille et mes collègues pour leur soutien pendant la réalisation de ce mémoire.

Résumé

La gestion des projets informatiques est une tâche assez compliquée. Aujourd'hui de nombreuses sociétés gèrent les projets informatiques avec les moyens qu'ils ont. Soit avec des environnements de génie logiciel centrés procédé, ou avec des fichiers, ou encore avec d'autres moyens. L'utilisation de ces moyens a pour but de faciliter la conduite de leur projet. Mais dans la plus part des cas tout ne se passe pas comme prévu, car on planifie le projet mais pendant la conduite de ce projet, on peut rencontrer des problèmes, ces problèmes sont appelés des variations.

Dans ce mémoire, nous vous proposons deux solutions pour essayer de faire face à ces variations c'est à dire de pouvoir les détecter et proposer si possible un plan de correction. La première solution est basée sur le principe des environnements de génie logiciel centrés procédé, la deuxième solution est l'utilisation d'un fichier Excel pour détecter les variations.

La détection d'une variation consiste à comparer le modèle de procédé défini et le processus réel de développement observé. Une fois qu'une variation est détectée, on propose des guides de correction au chef de projet afin que cette déviation n'ait pas un gros impact sur le projet.

Nous avons appliqué notre deuxième solution sur un cas d'étude car la première solution n'est pas implémenté.

Abstract

IT project management is a rather complicated task. Today, many IT companies manage their project with the resources they have : with process-centered software engineering environments, files, or other means. The main goal of using these means is to facilitate the project management. But in most of the time, everything do not go as planned. Because while the project management, we can have some problems, these problems are called "variations".

In this paper we proposed two solutions to detect these variations and propose if possible a correction plan. The first solution is based on the process-centered software engineering environments principle, and the second one is the use of an excel file.

We detect a deviation by the comparison of the defined process model and the actual observed development process. Once a change is detected, we proposed a correction plan to the project manager, to avoid that the detected variation has more impact on the project.

To finish, we applied our second solution on a case of study.

Table des matières

Remerciements	i
Résumé	ii
Abstract	iii
Contents	iv
Liste des figures	v
Liste des tableaux	vi
1 Introduction	1
1.1 Problématique	1
1.2 Objectif	2
1.3 Structure du document	3
2 État de l’art	4
2.1 Modélisation des procédés	4
2.1.1 Modèle de procédé	4
2.1.2 Langage de modélisation de procédés	6
2.2 Process-Centered Software Engineering Environments (PSEE)	11
3 Gestion des variations	16
3.1 Solution 1	16
3.1.1 Méta-modèle de procédé	17
3.1.2 Technique de détection des variations et guide de correction	20
3.2 Deuxième solution	21
3.2.1 Technique de détection	22
3.2.2 Application de la solution	30
4 Conclusion	35
Bibliographie	37

Liste des figures

2.1	Modèle conceptuel de procédés [1]	6
2.2	Modèle conceptuel de SPEM [30]	8
2.3	Méta-modèle de SPEM 1.1 [30]	9
2.4	Structure du SPEM 2.0 [2]	10
2.5	Rôle d'un PSEE [3]	11
2.6	Architecture de RHODES [4]	13
2.7	Architecture de SPADE [5]	15
3.1	Méta-modèle de SPEM solution	17
3.2	Fichier modèle	22
3.3	Structure des classes et modules	25
3.4	Mapping module	26
3.5	FileChecker module	28
3.6	Vérification des données d'entrées	29
3.7	Common function	30
3.8	Diagramme Temps-Temps des dates de début des tâches	31
3.9	Diagramme Temps-Temps des dates de fins des tâches	32
3.10	Diagramme d'évolution des tâches	33
4.1	Amélioration	36

Liste des tableaux

3.1	Information global sur le projet	22
3.2	Caractéristiques	23

Abréviations

CORBA : Common Object Request Broker Architecture

CWM : Common Warehouse Metamodel

MOF : Meta-Object Facility

OMG : Object Management Group

PML : Process Modeling Language

PSEE : Process-centered Software Execution Environment

SPEM : Software Process Engineering Metamodel

SPM : Software Process Model

SPML : Software Process Modeling Language

UML : Unified Modeling Language

Chapitre 1

Introduction

Dans le cadre de mon Master II MIAE, chaque étudiant devait chercher et traiter un sujet de niveau d'un « Master II MIAE » 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 effectuer 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 considéré 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 Language (PML)*.

Une fois les processus modélisés, les agents pourront suivre les étapes définies pour réaliser le produit (logiciel). L'erreur étant humaine, les agents ne sont pas à 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 mis 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 flexibles pour être adoptés dans le milieu industriel [14].

1.2 Objectif

Les objectifs attendus de ce travail de recherche peuvent être divisés en deux grandes parties.

- **État de l'art :**

dans cette partie nous allons étudier les PSEEs actuels : leurs architectures, leurs techniques 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 composée 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 :

- **L'introduction** : ce chapitre contiendra définir les motivations par rapport à ce sujet ainsi que les objectifs attendus.
- **L'étude de l'existant** : il présente l'architecture des PSEEs actuels ainsi que leurs techniques de détection.
- **La gestion des variations** : dans ce chapitre, nous présenterons nos solutions avec nos techniques de détection et de correction des variations.
- **La conclusion** : dans cette partie nous présenterons un bilan de ce travail par rapport à l'objectif et aborderons les perspectives d'améliorations 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 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 artefacts¹ 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 procédé². 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 artefacts 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.

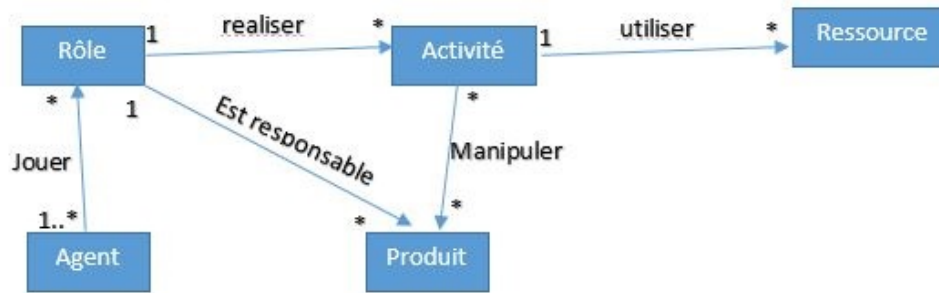


FIGURE 2.1: Modèle conceptuel de procédés [1]

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 : formel³, informel⁴, semi-formel⁵.
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 ⁷ "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 est qu'un procédé de développement de logiciel est une collaboration entre des entités

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

7. société Américaine créée en 1989, il est à la base des standards UML, MOF "Meta-Object Facility", CORBA "Common Object Request Broker Architecture", CWM "Common Warehouse Metamodel"

actives et abstraites (les rôles) qui effectuent des opérations appelés activités, sur des entités concrètes qui sont les produits (Figure 2.2).

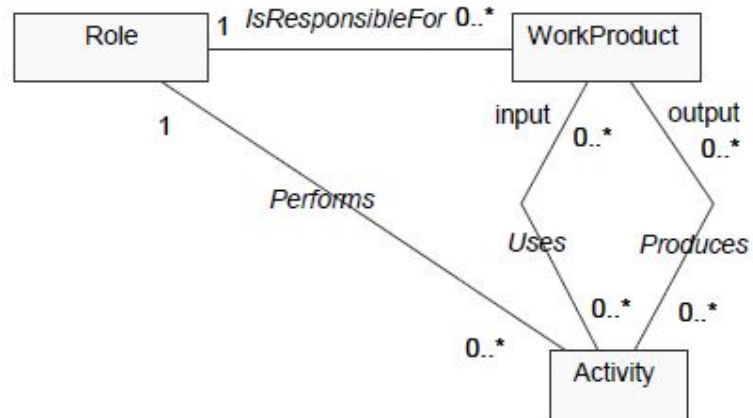


FIGURE 2.2: Modèle conceptuel de SPEM [30]

Aujourd'hui, il existe deux versions de SPEM : le **SPEM 1.1** et le **SPEM 2.0**.

- SPEM 1.1 : c'est la première version de SPEM, adopté en janvier 2005, il est composé de deux groupes : le *SPEM foundation* qui fournit les concepts de base pour définir le méta-modèle SPEM (Figure. 2.3) et le *SPEM-Extension* qui décrit le concept de base pour décrire les processus logiciels.

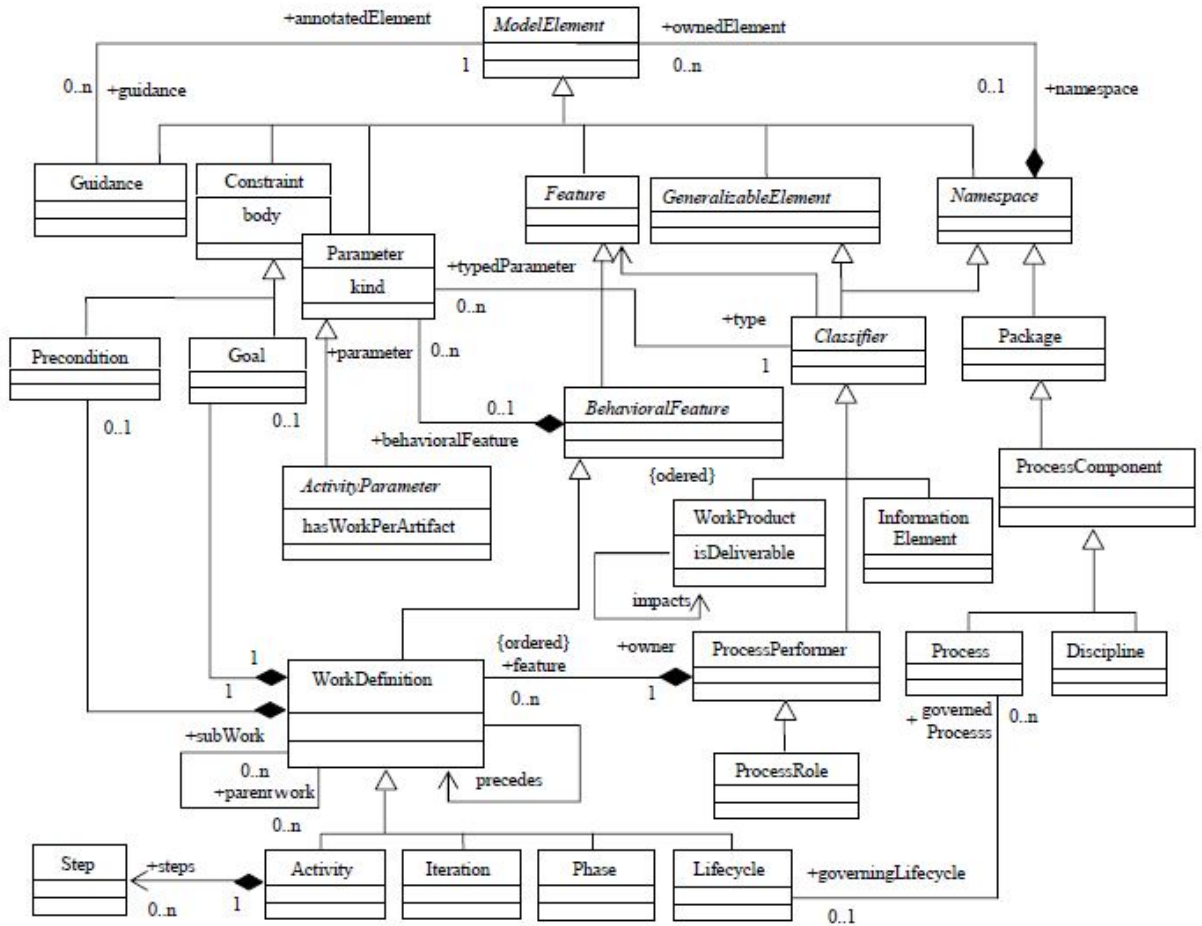


FIGURE 2.3: Méta-modèle de SPEM 1.1 [30]

- SPEM 2.0 : c'est la version la plus récente de SPEM depuis 2008. Comme on peut le voir sur la figure 2.4, SPEM 2.0 est composé de sept paquets.

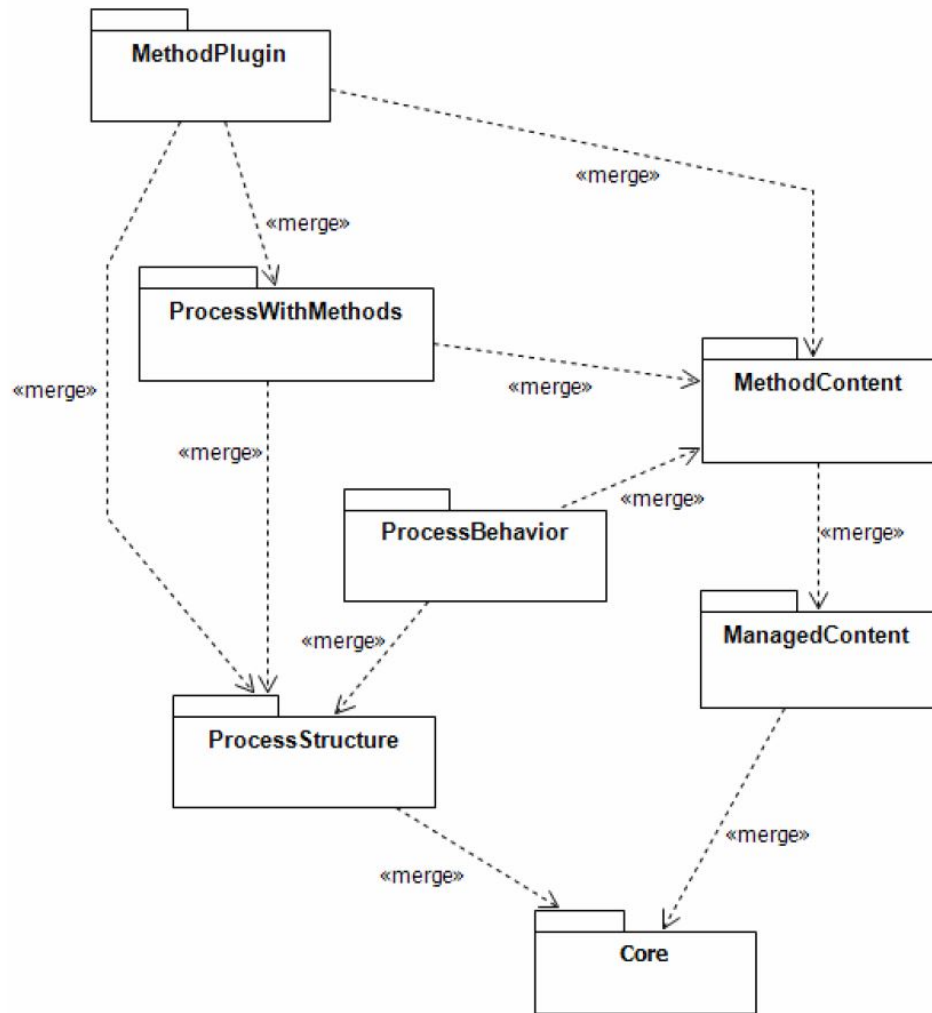


FIGURE 2.4: Structure du SPEM 2.0 [2]

- **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 2.5 résume le rôle d'un PSEE dans le processus de développement d'un logiciel.

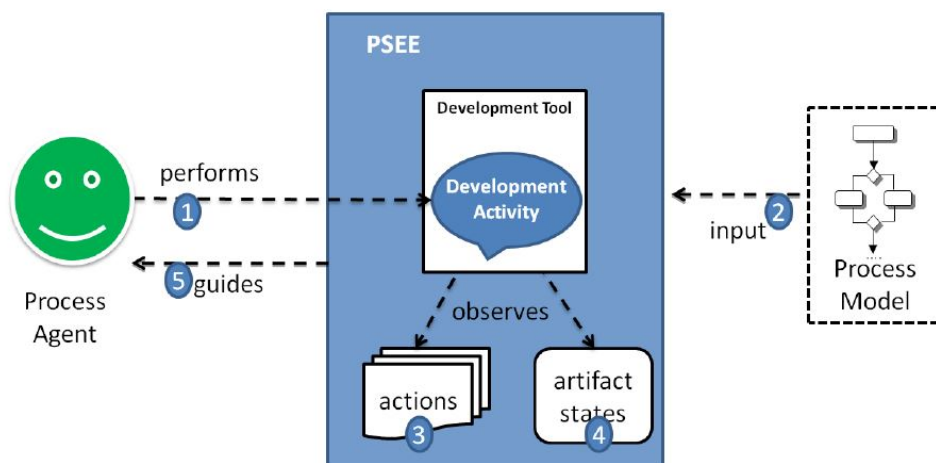


FIGURE 2.5: Rôle d'un PSEE [3]

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 défini, 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 « **dévi**ation ». 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 donnée⁸ 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 réfléchi⁹ 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

8. les informations sont stockées sous forme de collections d'objets persistants

9. 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'ENSEEIH-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 2.6 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

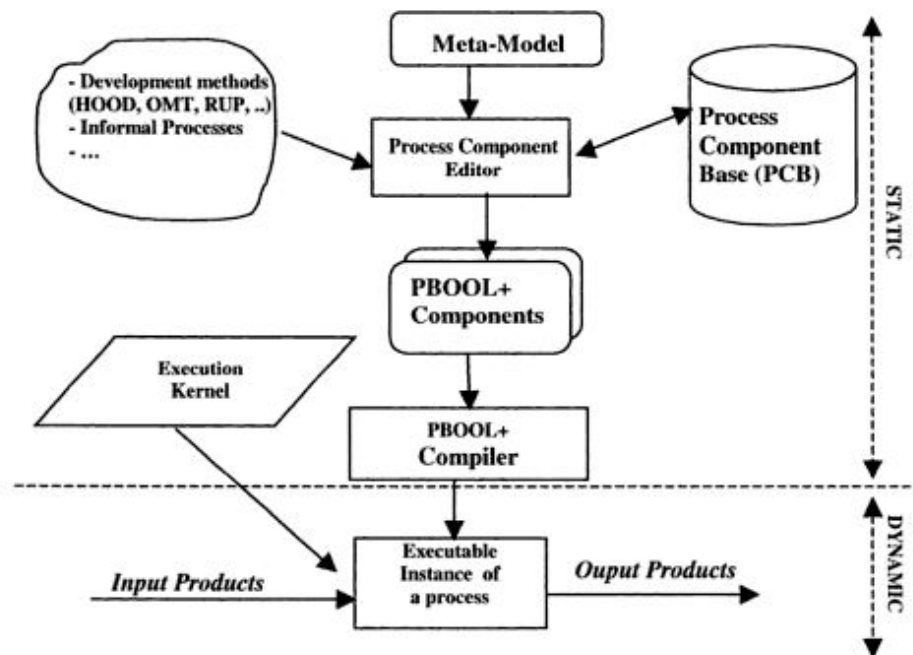


FIGURE 2.6: Architecture de RHODES [4]

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 artefacts créés 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.

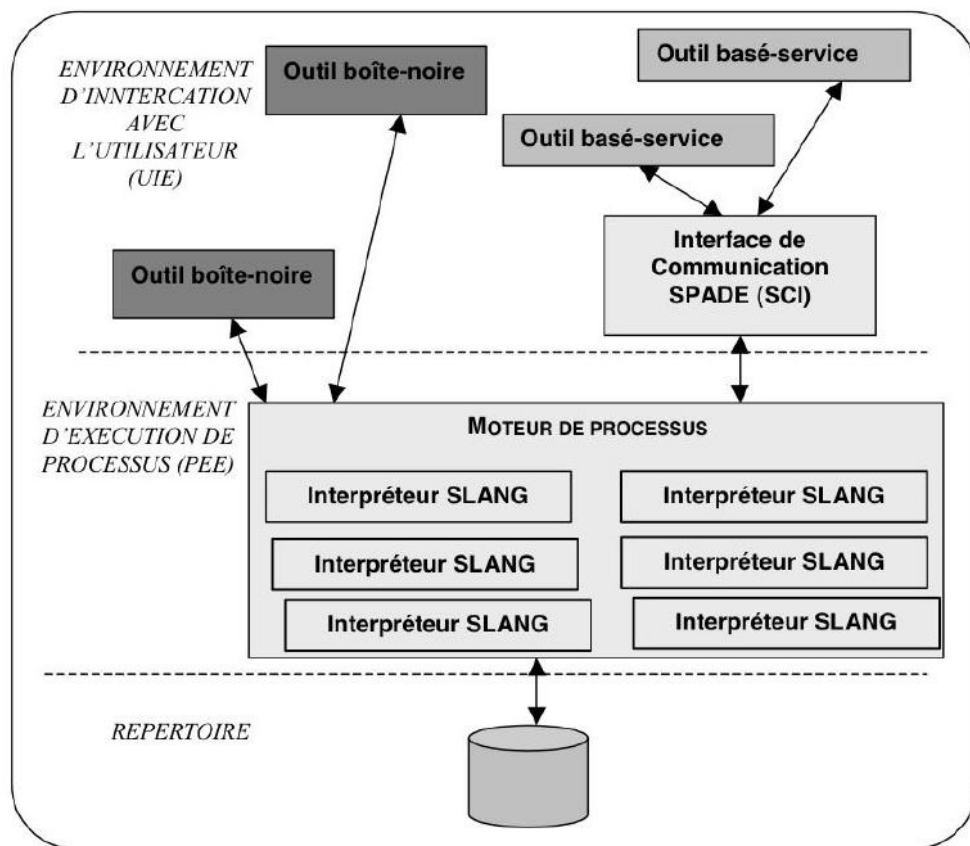


FIGURE 2.7: Architecture de SPADE [5]

Chapitre 3

Gestion des variations

Une variation est considéré comme toute action non définie dans le modèle de procédé initiale ou qui viole certaines contraintes de procédés. Après une étude de l'existant, nous allons proposer deux solutions de gestion de variations. La première solution consiste à traiter les variations grâce à un PSEE ; la seconde solution quand à elle, utilise un macro pour gérer les variations. Dans ce chapitre nous présenteront nos techniques de détection de variations (la technique à l'aide d'un PSEE et la seconde technique à l'aide d'un fichier excel) et nos plans de corrections.

3.1 Solution 1

Comme nous l'avons dit dans le chapitre [2](#), l'OMG a proposé en 2005 le SPEM 1.1, mais ce dernier (SPEM 1.1) présentant quelques lacunes, ils ont proposé en 2008 une nouvelle version (SPEM 2.2). Ces normes de l'OMG sont utilisées dans de nombreux travaux de recherches car elles sont considérées aujourd'hui comme étant des normes assez complètes pour pouvoir décrire les procédés. Alors pour cette première solution, je me suis beaucoup basé sur les solutions proposées par l'OMG dans leurs travaux.

- `stateObjective` : attribut de type *Objective*, il indique l'objectif ou le but du travail.

WorkProduct, WorkProductKind, WorkProductState

L'élément ***WorkProduct*** ou artéfact représente tout ce qui est produit, modifié ou consommé par un *ActivityWorkDefinition*. Ses attributs sont :

- `Name` : chaîne de caractère représentant le nom du produit.
- `isDeliverable` : booléen permettant de savoir si le produit peut être délivré ou pas.
- `Kind` : de type *WorkProductKind*, il permet de connaître le type du produit c'est à dire si c'est un document, model ou code.
- `State` : indique l'état du produit, il est de type *WorkProductState*.
- `isCreated` : booléen permettant de savoir si le produit fut créé ou pas par la tâche.

Un ***WorkProductKind*** représente les catégories de produits que l'on peut avoir. Ces catégories peuvent être du code, des documents, du model etc.

L'élément *WorkProductState* indique les états d'un produit (initial, validé, invalidé etc...)

RoleLevel, Role

RoleLevel est une énumération des niveaux qu'un rôle peut avoir (débutant, junior, confirmé, senior).

Un *Role* représente les rôles ou acteurs responsables d'un produit ou contribuant à la réalisation d'une tâche. Par exemple on peut avoir des analystes, des testeurs etc... Cet élément rôle possède un attribut *name* qui indique le nom du rôle ; un attribut *state* indiquant l'état du rôle c'est à dire si le rôle a terminé d'accomplir la tâche ou si il est en train de le faire etc... et un attribut *level* qui indique le niveau de l'acteur (débutant, junior, confirmé).

Postcondition et Precondition

Les éléments ***Precondition*** et ***Postcondition*** sont des spécialisation de la classe *Constraint*. Ils représentent respectivement les conditions qui doivent être satisfaites avant et après l'exécution de chaque *ActivityWorkDefinition*.

Parameter

L'élément **Parameter** est composé d'un attribut *kind* de type **ParameterDirectionKind** qui est une énumération des types de relations qui relie une tâche à un produit. C'est à dire qu'il permet d'indiquer si un *WorkProduct* est utilisé par une *ActivityWorkDefinition* tant que « entrée », « sortie » ou « entrée/sortie ».

Precedes et PrecedenceKind

Pour expliquer les relations qui lient nos *ActivityWorkDefinition*, nous allons prendre deux tâches t1 et t2, t2 étant le successeur de t1. L'élément **PrecedenceKind** est une énumération des relations qui peuvent exister entre nos *ActivityWorkDefinition* :

- pk_start_start* : ce type de relation permet d'indiquer que t2 ne pourra commencer que si t1 commence.

- pk_finish_finish* : ce type de relation permet d'indiquer que t2 ne pourra terminer que si t1 l'est.

- pk_finish_start* : indique que t2 ne pourra commencer qu'à la fin de t1.

L'élément **Precedes** comporte un attribut *kind* de type *PrecedenceKind*, il indique le type d'ordonnancement qui relie les *ActivityWorkDefinition*. Par exemple supposons que nos deux tâches t1 et t2 sont reliées par une liaison de type *pk_start_start*, t2 ne peut pas commencer tant que t1 n'a pas commencé.

Objective

L'élément **Objective** représente l'objectif ou le but de la réalisation d'une tâche. Si une tâche est terminée, on pourra vérifier si l'objectif a été atteint.

ResourceState, Resources

ResourceState représente les états de disponibilités des ressources. L'élément **Ressources** représente les différentes ressources. Ces ressources sont utilisées par des tâches selon leurs disponibilités.

LifeCycle, Phase, Iteration

L'élément **LifeCycle** représente le cycle de vie de logiciel par exemple « le modèle cascade » ou « le modèle en V ». La **Phase** est une spécialisation de *ActivityWorkDefinition*. C'est une tâche correspondant à une phase d'un cycle de vie de logiciel. L'élément **Iteration** est une *ActivityWorkDefinition* avec des préconditions et objectif bien défini appelé « minor milestone ».

3.1.2 Technique de détection des variations et guide de correction

Cette partie explique la technique de détection des variations et les solutions pour les résoudre.

Technique de détection

Rappelons nous qu'une variation est toute actions qui violent les contraintes définies dans notre procédé. Alors, notre technique de détection des variations est simple, elle se base sur certains attributs de notre modèle. Ces attributs sont tous les attributs représentant des contraintes. On surveille l'exécution des procédés, avant chaque exécution, on vérifie si les conditions requises pour commencer la tâche sont satisfaites c'est à dire si la pré-condition est satisfaite, si les relations de précédence sont également satisfaites. A la fin de chaque exécution, on vérifie si les post-conditions sont remplies, si la tâche est correctement terminée et l'objectif du travail est atteint. Si on voit qu'il y'a une anomalie au niveau de la satisfaction de nos contraintes alors on signale une déviation et on propose un plan de correction au chef de projet.

Guide de correction

Notre technique de correction est basée sur la ré-exécution de la tâche ou peut-être l'ignorer. Si une déviation est détectée, on regarde les causes ensuite on propose au chef de projet les solutions suivantes :

- ignorer la déviation si elle est mineure comme par exemple, on veut lancer une tâche dont toutes les pré-conditions sont satisfaites mais la date supposée être la

date de lancement de la tâche n'est pas arrivé, alors on peut proposer au chef de projet de modifier la date ou de l'ignorer. Car ça ne va pas générer des erreurs négatives (comme un retard dans la livraison).

■ la seconde solution à analysé les causes :

si c'est un dépassement de la date limite du travail, on pourra proposer au chef de projet d'affecter la tâche à un rôle plus compétent si le rôle responsable de la tâche n'est pas assez compétent sinon on modifie la date limite et on laisse le même rôle continuer à exécuter la tâche. Sinon si c'est l'exécution d'une action non défini dans le modèle de procédé initial, on demande au chef de projet de redéfinir le modèle de procédé.

3.2 Deuxième solution

Pour cette deuxième solution, je me suis rapproché d'un chef de projet chez « Alstom » afin de mettre en place cette solution. Elle est plus simple car elle est réalisée à l'aide d'un fichier excel appelé « fichier modèle » ou *template*.

3.2.1 Technique de détection

Dans cette partie nous allons présenter l'implémentation de notre technique de détection grâce au macro.

Fichier modèle (Template)

Le fichier modèle est un fichier Excel qui décrit la relation entre un projet, les tâches et les différentes dates qui sont nécessaires à leur suivi. Il est composé de trois feuilles (Params, Log et ForeCast). La dernière feuille 3.2 est composée de deux blocs principaux :

- Un premier bloc qui donne des informations sur le projet dans sa globalité
- Un second bloc qui décrit les différentes tâches nécessaires pour la réalisation de ce projet

Project Identification	Project1																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
------------------------	----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

FIGURE 3.2: Fichier modèle

Le premier bloc (information global sur le projet) contient les champs suivants (Champs en anglais).

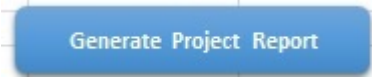
Champs	Description
Project identification	L'identifiant du projet (unique par projet)
Project Name	Le nom du projet
Project Manager	Le responsable du projet (chef de projet)
Project Begin Date	La date de début du projet
Project End Date	La date de fin du projet
Review Date	La date de dernière mise à jour du fichier

TABLE 3.1: Information global sur le projet

Le second bloc (information sur les tâches requises) contient les champs suivants (Champs en anglais) :

Champs	Description
Task ID	L'identifiant d'une tâche
Task	Le nom de la tâche
Task Description	La description de la tâche (champs libre)
Begin Date AT T0	La date de début de la tâche telle que planifiée à T0 (c'est-à-dire au tout début du projet). Ce champ ne doit plus être modifié après le lancement du projet
End Date At T0	La date de fin de la tâche telle que planifiée à T0 (c'est-à-dire au tout début du projet). Ce champ ne doit plus être modifié après le lancement du projet.
Task Priority	La priorité de la tâche. C'est un nombre compris entre 1 et 3
Task Predecessor	Permet de définir le lien entre tâche. Dans ce tableau, on ne vise pas à gérer les types de liens en série ou en parallèle, mais juste à définir le lien de dépendance entre différents jalons.
Real Begin Date	La date réelle de début de la tâche. Ce champ est à modifier en fonction des réalités et de l'avancement du projet.
Real Completion Date	La date réelle de fin de la tâche. Ce champ est à modifier en fonction des réalités et de l'avancement du projet
Comments	Un champ commentaire (champ libre)
Task status	Le statut de la tâche. Permet de mettre un flag pour savoir si la tâche est finie ou non.
Affected user	L'identifiant de la ressource associée à la tâche. Concerne les ressources humaines
User role	Le rôle de la ressource (sa fonction)

TABLE 3.2: Caractéristiques

En plus des différents filtres natifs qu'on peut utiliser dans Excel pour filtrer sur les jalons, ce fichier contient également un bouton  qui permet de générer un rapport de performance sur le projet.

Ce fichier modèle de base, permet de suivre les jalons, les mettre à jour et à chaque instant pouvoir générer des courbes et des indicateurs (à mettre en place en fonction des besoins) sur le suivi des tâches et leur déviation par rapport à la prévision initiale au début du projet.

Dans la section suivante, nous avons détaillé plus en profondeur la partie technique qui nous permet de générer le rapport d'état du projet

Le code de génération du rapport de performance

Le code de génération du rapport de performance est une macro c'est-à-dire qu'il a été développé avec du VBA (Visual Basic For Application) Excel. Architecturalement, ce code est composé des classes et des modules suivants :

Une classe ***C_Task*** : cette classe représente une tâche. Les attributs de cette classe sont principalement les champs du bloc 2 qui décrivent une tâche dans le fichier modèle.

Une classe ***C_Projet*** : Cette classe représente le projet dans sa globalité. Elle contient des attributs du bloc 1 qui décrivent le projet de façon global et aussi une liste (collection) de ***C_Jalon***

Un module ***Checker*** : Ce module a trois fonctions principales :

- la première fonction vise à analyser syntaxiquement le fichier modèle, c'est-à-dire s'assurer que les mots clés du fichier sont à la bonne place (champs des jalons et les champs qui décrivent le projet)
- la deuxième fonction vise à analyser le type de donnée rentré dans chaque champ et à s'assurer que la valeur saisie est la valeur attendue. La vérification syntaxique du fichier et de la qualité des données rentrées dans les champs garantit une cohérence quant aux données qui sont à traiter.
- La troisième fonction vise à vérifier les règles métiers de la gestion du projet. Par exemple, cette partie permettra de vérifier que la date de fin réel d'un jalon ne peut être inférieure à sa date de début etc. C'est vraiment basé sur la cohérence des données qui sont dans le fichier.

Un module **Reader** : après la vérification de syntaxe et de cohérence des données, le module Reader permet de lire le fichier Excel et de charger les données dans la classe *C_Projet*. Ce qui nous permet de gérer des objets de types projets et ainsi bénéficier de la puissance de programmation orientée objet.

Un module **Mapping** : afin de ne pas mettre en dure dans le code certains paramètres comme la colonne excel dans laquelle se trouve une valeur et ainsi garantir plus de flexibilité, le module Mapping permet de définir l'ensemble des paramètres qui pourraient être modifié demain. Il définit la structure du fichier et permet une utilisation plus propre du fichier.

Un module **Module Z11** : ce module contient l'ensemble du code qui permet de générer un rapport de performance.

Un module **CommonFunctions** : ce module contient les fonctions communes qui peuvent être utilisées dans les autres modules. On rencontre en outre, les fonctions de conversion en majuscule, les fonctions d'ouverture de l'explorateur pour aller chercher un fichier etc.

Pour plus de détails sur le fonctionnement du code, se référer au fichier modèle et ouvrir l'éditeur de texte Visual basic.

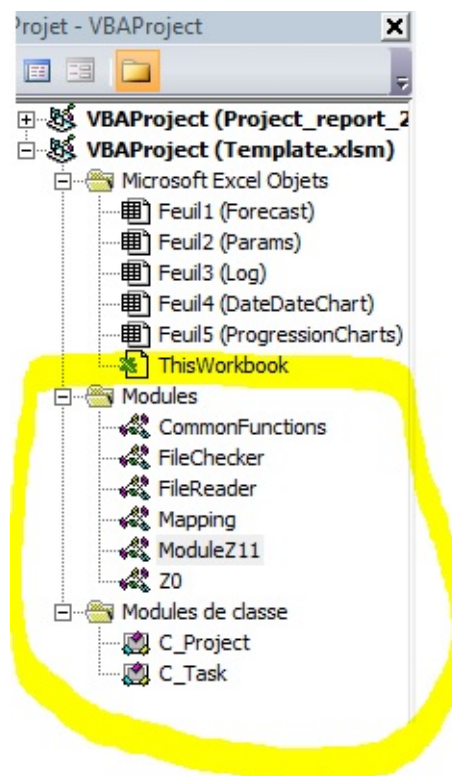


FIGURE 3.3: Structure des classes et modules

Développement des modules

Le module **Mapping** définit la structure du fichier et contient les mots clés et les noms de colonnes. L'image 3.4 nous donne un extrait de ce module.

Le module **Mapping** définit la structure du fichier et contient les mots clés et les noms de colonnes. L'image 3.4 nous donne un extrait de ce module

```
Public Const Keyword_ProjectIdentification As String = "A2"
Public Const Keyword_ProjectName As String = "A3"
Public Const Keyword_ProjectManager As String = "A4"
Public Const Keyword_ProjectBeginDate As String = "A5"
Public Const Keyword_ProjectEndDate As String = "A6"
Public Const Keyword_ReviewDate As String = "A7"

Public Const KeywordValue_ProjectIdentification As String = "B2"
Public Const KeywordValue_ProjectName As String = "B3"
Public Const KeywordValue_ProjectManager As String = "B4"
Public Const KeywordValue_ProjectBeginDate As String = "B5"
Public Const KeywordValue_ProjectEndDate As String = "B6"
Public Const KeywordValue_ReviewDate As String = "B7"

Public Const Col_TaskID As String = "A"
Public Const Col_Task As String = "B"
Public Const Col_TaskDescription As String = "C"
Public Const Col_TaskBeginDateAtT0 As String = "D"
Public Const Col_TaskEndDateAtT0 As String = "E"
Public Const Col_TaskPriority As String = "F"
Public Const Col_TaskPredecessor As String = "G"
Public Const Col_RealBeginDate As String = "H"
Public Const Col_RealCompletionDate As String = "I"
Public Const Col_Comments As String = "J"
Public Const Col_TaskStatus As String = "K"
Public Const Col_UserAffected As String = "L"
Public Const Col_UserRole As String = "M"
```

FIGURE 3.4: Mapping module

Le module ***FileChecker*** contient les fonctions suivantes :

- *VerifSyntaxFile* permet de vérifier la structure du fichier. Retourne vraie si la structure est bonne, et faux dans le cas contraire.
- *o VerifiDataFile* permet de vérifier la qualité des données saisies dans le fichier. Retourne vrai si on a les bonnes valeurs dans les champs, et faux s'il y a des champs avec des valeurs incorrectes.

Ces deux fonctions sont incluses dans une fonction globale appelée *CheckFile*.

Voici une image [3.5](#) de ce module (Se référer au fichier modèle pour plus de détails)

```

Public Sub CheckFile()
    Dim wsForecast As Worksheet
    Dim sText As String

    'Log le nom du fichier
    Call WriteLogMessage(" ")
    Call WriteLogMessage("Filename : " & ActiveWorkbook.Name)

    '=====
    ' 1) OPEN FORECAST SHEET
    '=====
    On Error Resume Next
    Set wsForecast = ActiveWorkbook.Worksheets(ForecastSheetN)
    On Error GoTo 0
    If wsForecast Is Nothing Then
        sText = "ERROR : file must contain 'FORECAST' sheet (
        Call WriteLogMessage(sText)
        'Close file
        ActiveWorkbook.Close
        Exit Sub
    End If

    '=====
    ' 3) VERIF SYNTAX
    '=====
    ' call ckeck syntax
    If VerifSyntaxFile() = False Then
        'Close file
        ActiveWorkbook.Close
        Exit Sub
    End If

    '=====
    ' 4) VERIF DATA
    '=====

```

FIGURE 3.5: FileChecker module

Pour des raisons de temps, ce module de vérification n'est pas complètement fini. Beaucoup de cas de vérification restent à implémenter.

Le module ***FileReader*** permet de lire le fichier Excel, transformer les données en un objet *C_Project* et générer le rapport de performance. Ce module contient les fonctions suivantes :

- ***UpdateDATAFromFile*** : permet de lire le fichier modèle et retourne un objet de type *C_Project*. Voici un extrait de cette fonction [3.6](#)

```

Public Function UpdateDATAFromFile(ByVal wsWorksheet As Worksheet) As C_Project

    Set UpdateDATAFromFile = Nothing

    Dim iCurrentRow As Integer
    Dim iLastRow As Integer
    Dim sTempValue As String
    Dim sProjectCode As String

    Dim oProject As C_Project
    Dim oJalon As C_Task
    Dim sStringKey As String

    Dim sText As String

    iCurrentRow = 0
    iLastRow = 0
    sTempValue = ""

    ' TESTE DONNEE ENTREE
    '=====
    If wsWorksheet Is Nothing Then
        UpdateDATAFromFile = Nothing
        sText = "SYSTEM : bad input parameter : UpdateDATAFromFile(Nothing). File not proce
        Call WriteLogMessage(sText)
        Exit Function
    End If

    Dim ProjectIdentification As String

```

FIGURE 3.6: Vérification des données d'entrées

- *GenerateReportfile* fait appelle à la fonction *UpdateDataFromFile()* et la fonction *SaveReportFile()* pour générer un rapport de performance.

Le module *ModuleZ11* contient les fonctions nécessaires à la génération du rapport. Ces fonctions sont appelées par la fonction *SaveReportingFile*. Ce module contient les fonctionnalités suivantes :

- *GenerateReportingSheets* : permet de créer dans le rapport de performance les feuilles nécessaires et appelle toutes les autres fonctions pour remplir ces feuilles
- *WriteDownForectSheetValues* : recopie dans une feuille excel du rapport, les dates nécessaires des tâches lues depuis le fichier modèle
- *StoreMinAndMaxDate* : permet de détecter et de rajouter la tâche avec la date minimale et la tâche avec la date maximale. Ces dates permettent de définir correctement l'échelle utilisée pour les courbes. Un tri sur les dates est effectué afin de garantir l'ordre
- *CopyChartsTemplate* : permet de copier dans le fichier des rapports les templates des courbes pour pouvoir mieux tracer dessus (Ces templates sont dans le fichier modèle dans des feuilles cachées)
- *drawDayDayChart* : permet de tracer la courbe *date-date chart*
- *drawCumulativeChart* : permet de tracer la courbe *Task Evolution*

Le module **Commonfunctions** : contient les fonctions communes utilisées dans toute l'application. Voici un extrait 3.7 de quelques fonctions utilisées dedans.

```
' MET EN MAJUSCULE ET RETIRE LES ESPACES A DROITE ET A GAUCHE D'UNE CHAINE
'=====
Public Function ChangeCase(ByVal sString As String) As String
' entree : chaine
' sortie : chaine en majuscule sans espace à droite et gauche
sString = UCase(sString)
ChangeCase = Trim(sString)
End Function

'=====
' RETOURNE LE CHEMIN DU DESKTOP (avec separateur)
'=====
Public Function DesktopDir()
DesktopDir = CreateObject("WScript.Shell").SpecialFolders("Desktop") & Application.PathSeparator
End Function

Public Sub addNeededSheets(ByRef wb As Workbook)
Dim w As Worksheet

wb.Worksheets.Add.Name = REPORT_DATASHEET
'Delete unnecessary sheets
Application.DisplayAlerts = False
For Each w In wb.Worksheets
If w.Name <> REPORT_DATASHEET Then
w.Delete
End If
Next
Application.DisplayAlerts = True

wb.Worksheets.Add.Name = REPORT_TASKSCURVESHEET
wb.Worksheets.Add.Name = REPORT_PROGRESSIONCHARTS
End Sub
```

FIGURE 3.7: Common function

3.2.2 Application de la solution

Afin de valider cette deuxième solution, nous l'avons expérimenté sur un projet comportant : une tâche d'analyse, une tâche conception, six tâches de développement des fonctionnalités et six tâches de tests des fonctionnalités développées. Ce projet a une date de début prévue le 31 août 2015 et une date de fin 11 septembre 2015. Les tâches d'analyses, conceptions et développement tâche 1.1 respecte la prévision. Mais à partir de la tâche (tâche 1.2) on commence à avoir une déviation. En appliquant notre solution sur cet exemple nous obtenons les résultats suivants :

Le diagramme Temps-Temps (Date-Date Chart)

Le diagramme temps-temps est un outil de management de projet. Grâce à une représentation en diagramme temps-temps, il est possible de voir graphiquement l'évolution au cours du temps des dates prévisionnelles des tâches par rapport aux dates réelles sur un projet. Ce qui nous permet de détecter assez rapidement et de façon visuelle les dérives des tâches par rapport à ce qui était prévu initialement. Par exemple, sur la figure 3.8, nous

avons un diagramme temps-temps qui présente graphiquement **la variation réelle des dates de début des tâches** d'un projet par rapport à ce qui était prévu à l'état initial c'est-à-dire au début du projet. **Interprétation du diagramme 3.8** Le diagramme

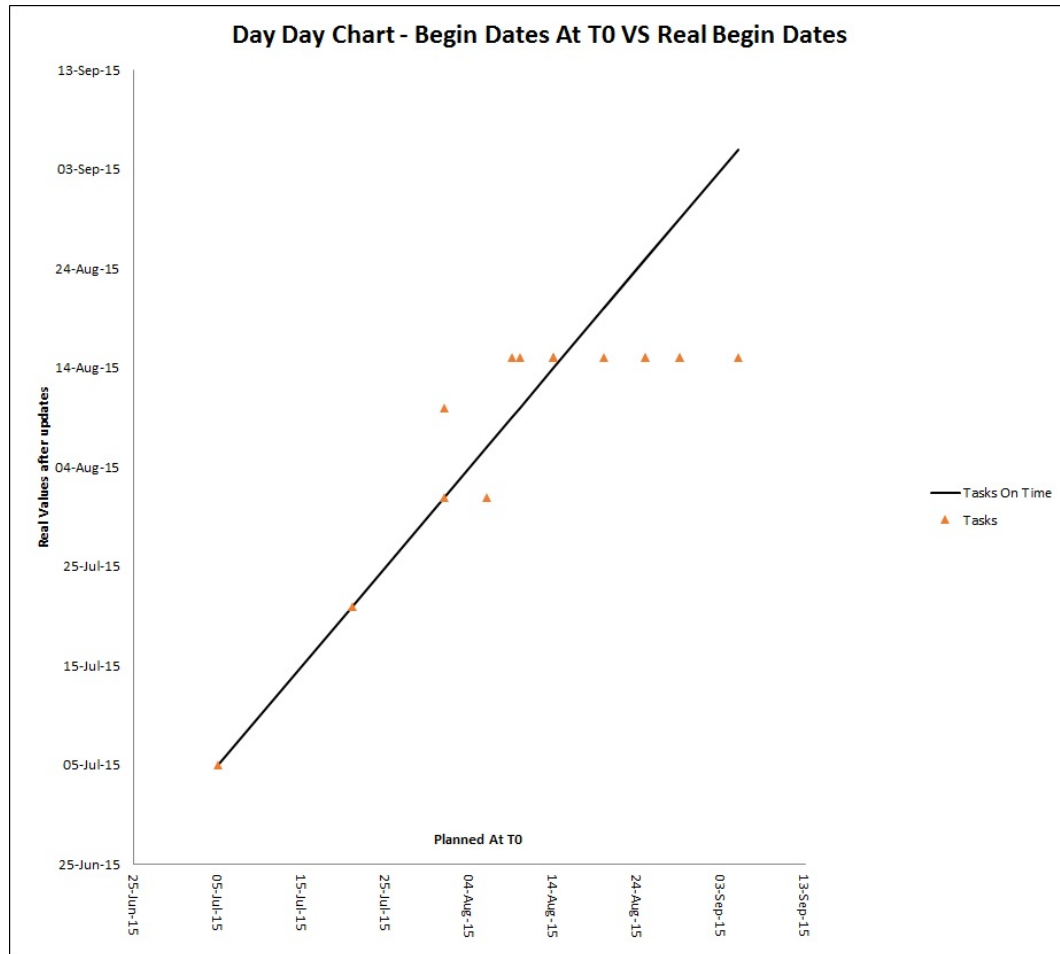


FIGURE 3.8: Diagramme Temps-Temps des dates de début des tâches

présenté se base sur deux indicateurs des jalons : La date initiale prévue de début des tâches (*Begin date planned at T0*) et la date réelle de début des tâches (*Real begin date*) qui est fréquemment mise à jour. Dans cette figure, l'axe des X (Horizontal) représente le calendrier initial (*dates initiales*) des dates de début des tâches. L'axe des Y (Vertical) représente quant à lui le calendrier des dates de début mises à jour à une période donnée (T).

Selon le site [project planning office](#) : « la diagonale d'un tel graphique a toute une signification. Par sa définition mathématique ($Y=X$), elle correspond à des événements dont la date prévisionnelle de réalisation ou d'achèvement (Y) et celle du jour où on en fait la prévision (X) sont identiques ». Dans notre cas, nous pouvons dire que cette diagonale représente la référence (*base line*) des dates de début de nos tâches à la période T0 (Au

début du projet) et nous servira à détecter les déviations de dates de début.

Les petits triangles (▲) quant à eux déterminent la dérive des tâches par rapport à la référence. En gros, toutes les tâches qui sont sur la diagonale commencent à la date de début prévue initialement ; toutes les tâches qui sont situées au-dessus de la diagonale représentent les tâches qui commencent plus tard que leur date initialement prévue ; et toutes les tâches qui sont situées en-dessous de la diagonale représentent les tâches qui commencent plus tôt que leur date initialement prévue. Pour plus d'information sur les diagrammes temps-temps, vous pouvez vous référer au site de [project planning office](http://projectplanningoffice.com). Le même graphique est généré pour les dates de fins des tâches afin de détecter les variations sur les dates de fin (Figure 4.1).

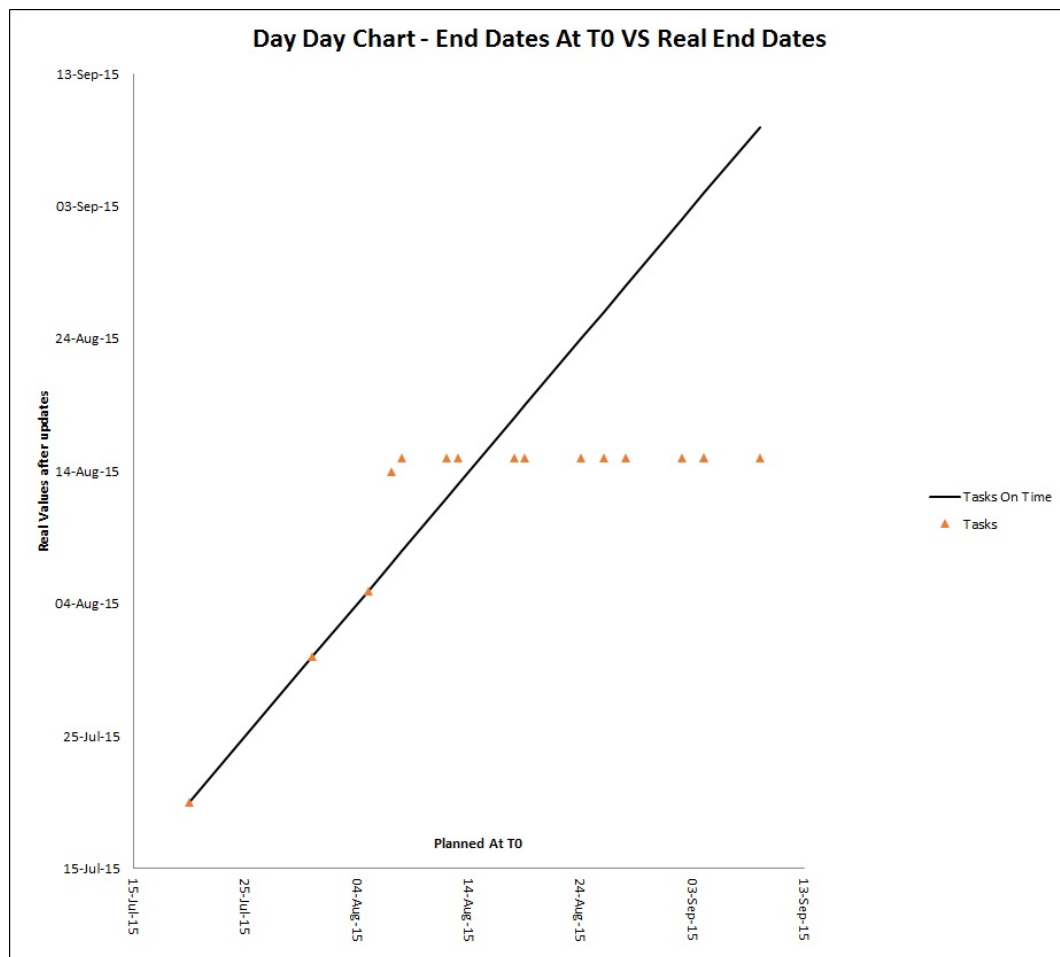


FIGURE 3.9: Diagramme Temps-Temps des dates de fins des tâches

Le diagramme d'évolution des tâches et de comparaison du prévisionnel par rapport au réel

Ce diagramme est un outil de management de projet et permet de comparer l'évolution des tâches telle que prévue initialement, et l'évolution réelle des tâches après une mise à jour des dates de début de tâche et de fin de tâche. **Interprétation du diagramme ??**

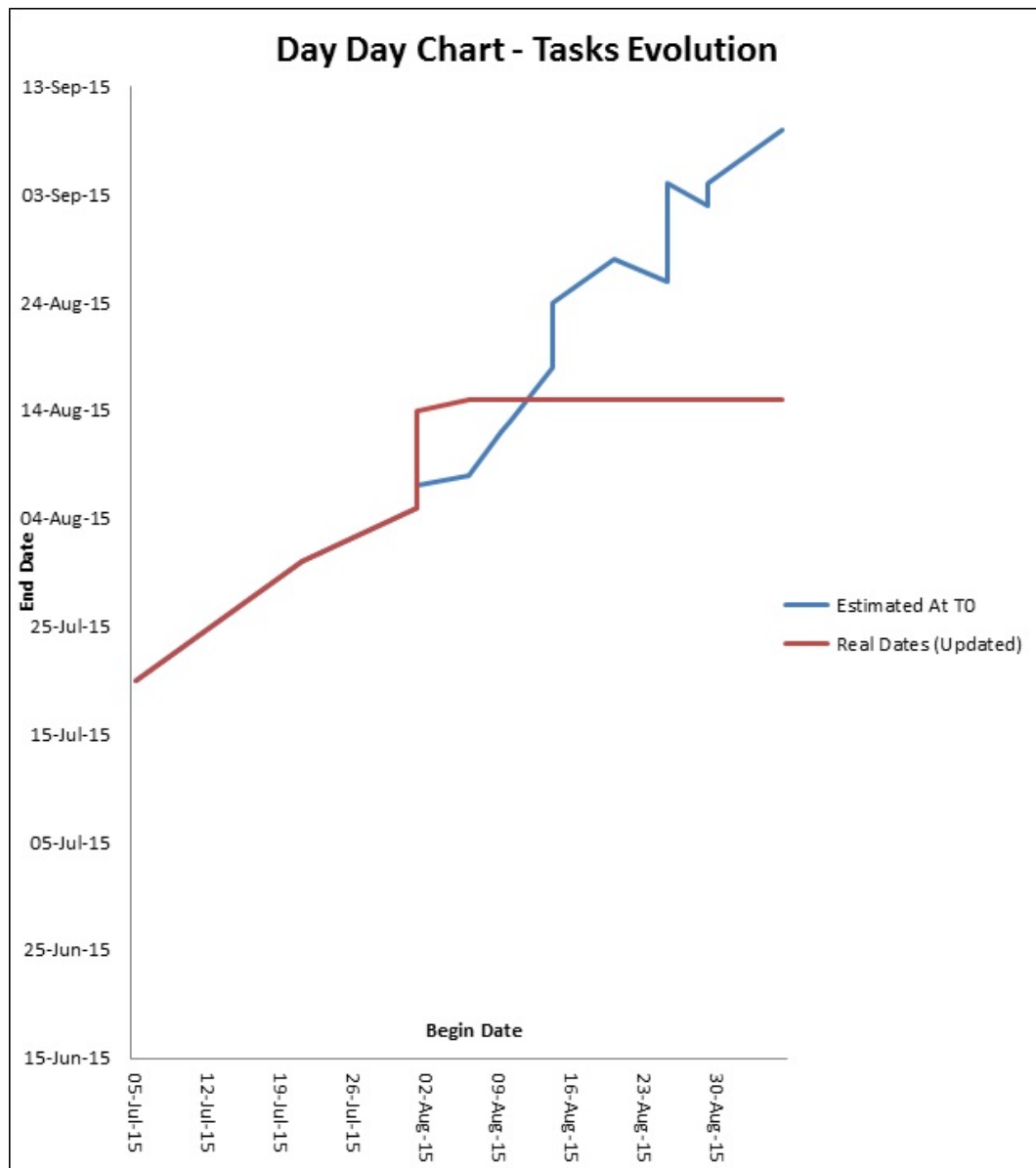


FIGURE 3.10: Diagramme d'évolution des tâches

Tout comme le diagramme présenté auparavant (Figure 3.8), les axes X et Y de ce graphique ?? ont des échelles de dates. Sur l'axe des abscisses, on affecte les dates de début des tâches et sur l'axe des ordonnées, on affecte les dates de fin des tâches.

Les courbes qui résultent des couples (X, Y) sur ce graphique représentent l'évolution

dans le temps des tâches en se basant sur leur date de début et de fin.

Dans notre figure, la courbe en bleue représente l'évolution des tâches telles que initialement prévues (date de début et de fin initiales) ; et la courbe en rouge représente l'évolution réelle des tâches après mises à jour des dates de début et de fin des tâches.

En comparant les deux courbes rouge et bleue, on peut interpréter ainsi :

- A une date de début d (en regardant l'axe des abscisses X), si le point de la courbe en rouge est au-dessus du point de la courbe en bleue, ça veut dire qu'il y a un retard sur la réalisation de la tâche par rapport à ce qui était prévu initialement. Si par contre, le point de la courbe en rouge était en dessous du point de la courbe en bleue, ça veut dire qu'on a de l'avance sur la réalisation de tâche par rapport à ce qui était initialement prévu.
- A une date de fin f (en regardant l'axe des ordonnées Y), si le point de la courbe en rouge est à droite du point de la courbe en bleue, ça veut dire qu'on a du retard sur la date de début de la tâche par rapport à ce qui était prévu initialement. Si par contre, le point de la courbe en rouge était à gauche du point de la courbe en bleue, ça veut dire qu'on a de l'avance sur la date de début de la tâche initialement prévue.

Chapitre 4

Conclusion

Dans ce travail de recherche, nous avons étudié les variations par rapport aux prévisions dans la gestion des projets informatiques. Nous avons pour objectif d'étudier les variations et de proposer un modèle de gestion de ces variations pendant la gestion du projet. Nous avons proposé deux solutions : une première solution basée sur les PSEE et une seconde solution avec un fichier excel.

La première solution est composée de trois parties : le méta-modèle pour décrire les procédés, la technique de détection des variations et le plan de correction pour la variation détectée.

La seconde solution a été mise en place suite à une discussion avec un chef de projet chez Alstom. C'est un macro excel permettant de détecter les variations grâce aux paramètres qu'on lui fournit et nous affiche les résultats sous forme de graphe.

Même si ces solutions permettent de détecter quelques variations, et aident le chef de projet dans la prise de décision ; elles ne peuvent pas détecter tous les types de variations.

Limites et perspectives : les limites et les perspectives d'amélioration de nos deux solutions sont :

Solution 1 :

Notre première solution est assez limitée car elle ne permet pas d'anticiper la détection des variations. La variation n'est détectée que lorsque les tâches sont terminées.

Solution 2 :

Les diagrammes résultant de l'exploitation de notre modèle de fichier Excel initialement mis en place et nous permettent à travers des représentations graphiques de voir

systématiquement la variation sur les dates de début et de réalisation des tâches.

Par rapport aux diagrammes 3.8 et 4.1 on pourra améliorer ces derniers en traçant une droite horizontale qui représente la date de validité (date à laquelle les données ont été mises à jour), on aurait pu séparer les taches précédentes des tâches futures à venir ??.

D'autres types d'indicateurs peuvent également être mis en place en fonction des be-

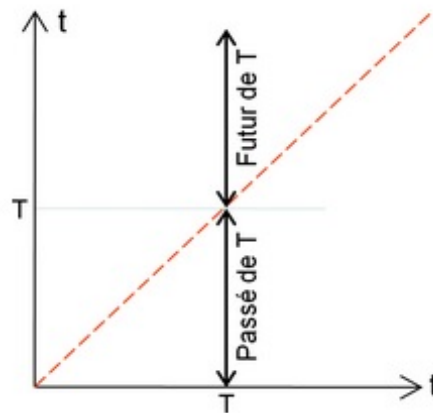


FIGURE 4.1: Amélioration

soins. On pourra par exemple mettre en place un tableau de synthèse de variations qui pourra être importé dans un autre outil etc.

Bibliographie

- [1] Hanh Nhi Tran. *Modélisation de procédés logiciels a base de patrons reutilisable*, pages 10–11. 2010.
- [2] OMG. Software process engineering 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=AvtEC1DmY3cC&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/MACA0/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-plant-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 engineering 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=OCF4Q6AEwBw#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=OCGMQ6AEwCA#v=onepage&q=SPADE&f=false.