ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DU
DOCTORAT EN GÉNIE
Ph. D.

PAR
PALZA VARGAS, Edgardo

RÉFÉRENTIEL DES PROCESSUS ET D'UN ENTREPÔT GÉNÉRIQUE DES MESURES POUR LA VÉRIFICATION & VALIDATION (V&V) DES LOGICIELS CRITIQUES

MONTRÉAL, LE 18 DÉCEMBRE

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE :

M. Alain Abran, directeur de thèse
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. Christopher Fuhrman, co-directeur de thèse
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. Roger Champagne, président du jury
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. Eric Lefebvre, membre du jury
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. Yann-Gaël Guéhéneuc, examinateur externe
Département d'informatique et recherche opérationnelle à l'Université de Montréal

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 12 DÉCEMBRE

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je ne pourrais pas présenter ce travail de recherche aujourd'hui si un certain nombre de personnes ne m'avaient pas soutenu tout au long de mes études de doctorat.

Je tiens tout d'abord à remercier à mes directeurs de thèse, les professeurs Alain Abran et Christopher Fuhrman, qui tout au long de ma recherche ont fait preuve de patience et ont su m'encadrer de manière rigoureuse. Je voudrais les remercier particulièrement de leurs généreux conseils, encouragements, supports et efforts sans lesquels cette thèse n'aurait vu la lumière.

J'aimerais également remercier ma précieuse famille : mes enfants Gabriel et Edgardo Jr. et ma mère Maria qui ont su croire en moi et qui m'ont soutenu pendant mes études. Une pensée toute particulière pour ma chère épouse Lucia, pour sa patience, son soutien et son écoute sans qui ce travail n'aurait pas abouti non plus.

En fin un grand merci à mes amis et collègues étudiants et professeurs du laboratoire de recherche de génie logiciel de l'École de technologie supérieure, pour leur support et encouragement pendant mes études.

RÉFÉRENTIEL DES PROCESSUS ET D'UN ENTREPÔT GÉNÉRIQUE DES MESURES POUR LA VÉRIFICATION & VALIDATION (V&V) DES LOGICIELS CRITIQUES

PALZA VARGAS, Edgardo

RÉSUMÉ

Pour les secteurs tel que l'aviation, le médical, le nucléaire et l'aérospatial, la qualité du logiciel est un besoin capital. Le logiciel embarqué dans ces types des systèmes est reconnu comme étant critique car leurs échec peut causer blessures (ou mort) aux êtres humains ainsi que la perte d'investissements financiers considérables.

Dans cette thèse nous proposons un référentiel des processus de vérification et validation (V&V) et d'un entrepôt générique des mesures pour permettre, d'une part, de mieux définir et implémenter les objectifs, les processus, les activités, les tâches, les rôles, les produits de travail de la V&V dans les projets et, d'autre part, de faciliter la prise des décisions dans les projets de V&V par le biais d'un entrepôt des données des mesures pour mieux identifier, faire le suivi et l'évaluation des processus et produits (ex. des coûts, du temps, des ressources, etc.) des activités de V&V.

Le référentiel sert à la définition des pratiques de V&V du développement du logiciel critique et en concordance avec les exigences de certification du guide DO-178B dans le contexte de l'aéronautique. Le contenu de ce référentiel de V&V est basé sur les pratiques du développement logiciel recommandées par les Processus Unifiés (PU). Nous avons utilisé la version « Open Source » des processus unifiés appelé « OpenUP » pour construire le référentiel.

Le référentiel de pratiques de V&V est exprimé par le biais d'un artefact logiciel appelé «Plug-in» ou «plugiciel». Cet artefact a été construit avec l'outil « Eclipse Process Framewrok Composer (EPFC) », qui est une extension du « OpenUP ». EPFC est basé sur les principes de SPEM-OMG (Software Process Engineering Metamodel – Object Management Group). L'EPFC donne la capacité au référentiel des pratiques de V&V de pouvoir être personnalisé ou adapté selon les particularités de chaque projet du logiciel critique.

La proposition des mesures pour les activités de V&V est fondée sur la proposition d'un entrepôt des mesures basé sur les principes décrits dans la norme ISO 15939 pour la construction des processus mesures dans le logiciel.

L'entrepôt des mesures est basé également sur un modèle générique des mesures ainsi que sur la technologie OLAP (Online Analytical Processing) pour l'analyse multidimensionnelle des données des mesures. L'entrepôt des mesures proposé est basé sur un modèle générique qui ne présuppose pas un ensemble des mesures définies mais une structure basée sur un modèle orienté-objets qui permet de définir divers types de mesures basées sur les mêmes données stockées dans l'entrepôt. La technologie OLAP confère la flexibilité, la capacité de

stoker, la capacité de représenter ainsi que la capacité d'analyser des mesures dans plusieurs perspectives.

Cette thèse, au cours de son élaboration, a bénéficié de la collaboration de différentes partenaires industriels tels que : NASA IV&V Facility, CMC Electronics, Verocel Inc. et Ericsson Research Canada.

A REPOSITORY OF PROCESSES AND A GENERIC DATA WAREHOUSE OF MEASURES FOR VERICATION & VALIDATION (V&V) IN SAFETY CRITICAL SOFTWARE

PALZA VARGAS, Edgardo

ABSTRACT

For sectors such as avionics, medical, nuclear and aerospace, software quality is a challenge. The software embedded in these systems is recognized as safety critical because its failure can cause injury (or death) to human beings and/or the loss of considerable financial investments.

In this thesis we propose a repository of Verification and Validation (V&V) processes and a Generic Data warehouse of Measures. The repository of V&V process aims to facilitate the definition and implementation of goals, processes, tasks, roles, work products for V&V in safety critical projects. The Data warehouse aims as well to facilitate the decision-making in the V&V projects, through a Generic Data warehouse to better identify, monitor and evaluate processes and products of V&V (i.e., costs, time, resources, etc.).

The repository of V&V processes is consistent with the DO-178B requirements for Software Certification. The content of this repository of V&V is based on the software development practices recommended by the Unified Process (UP). We used the "Open Source" version of the Unified Process called OpenUP to develop the proposed V&V repository.

The repository of V&V is expressed through a software application called "Plug-in". This software application was developed using the tool "Eclipse Process Framework Composer (EPFC)", which is an extension of the OpenUP. The EPFC is based on the principles of SPEM-OMG (Software Process Engineering Metamodel - Object Managment Group). The EPFC enables the repository of V&V to be customized to the particularities of each safety critical project.

The measurement perspective in the repository of V&V is based on the proposal of a Data warehouse of measures aligned with the principles described in ISO 15939 for the construction of the measurement process for software.

The V&V repository is based on a generic model of measures and the OLAP (Online Analytical Processing) Technology for the multidimensional analysis of measures. The Measures Repository is based on a generic model so that the measures do not presuppose particular predefined measures but a structure based on a object-oriented model that can define various types of measures on the same data in the Data warehouse. The OLAP technology enables the possibility of storing, representing and analysing measures from several perspectives.

This thesis, during its development, has been benefited from collaboration with the following industrial partners: NASA IV&V Facility, CMC Electronics Inc., Verocel Inc. and Ericsson Research Canada.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
CMP	Configuration Management Plan
COTS	Commercial Off-the-Shelf
DAF	Document Approval Form
DER	Designated Engineering Representative
FAA	Federal Aviation Administration
FAA- iCMM	FAA Integrated Capability Maturity Model Version 2.0 (iCMM)
GÉLOG	Laboratoire de Génie logiciel de l'ÉTS
IDEF0	Integrated Definition Method 0
IEC	International Electrotechnical Commission
IV&V	Independent Verification and Validation
NASA	National Aeronautics and Space Administration
OLAP	Online Analytical Process
OMG	Object Management Group
PSAC	Plan for Software Aspects of Certification
PSM	Practical Software Measurement
RTCA	Requirements and Technical Concepts for Aviation
RUP	Rational Unified Process
SAS	Software Accomplishment Summary
SCDP	Software Certification Data Package
SCI	Software Configuration Index
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SCstd	Software Coding Standard
SDD	Software Design Document

SDF	Software Development Folder
SDL	Software Development Library
SDP	Software Development Plan
SDstd	Software Design Standard
SECI	Software Life Cycle Environment Configuration Index
SEE	Software Engineering Environment
SPEM	Software Engineering Process Metamodel
STP	Software Test Plan
STRATS	System Requirements Allocated To Software
SVA	Software Vulnerability Analysis
SVP	Software Verification Plan
SWEBOK	Software Engineering Body of Knowledge
UML	Unified Modeling Language
UP	Unified Process
V&V	Verification and Validation

GLOSSAIRE

Architecture de logiciel (« Software Architecture ») : la structure du logiciel déterminée pour implémenter les exigences du logiciel (voir le concept de structure dans ce glossaire) (RTCA/DO-178B, 1992).

Architecture du système (« System Architecture ») : la structure du matériel déterminée pour implémenter les exigences du système (RTCA/DO-178B, 1992).

Certification : identification légale par une autorité de certification qu'un produit, un service, une organisation ou une personne est conforme aux certains critères. En particulier, la certification d'un produit implique : (a) le processus d'évaluer la conception d'un produit pour s'assurer qu'il est conforme à un ensemble de normes applicables à ce type de produit afin de démontrer un niveau acceptable de sûreté ; (b) le processus d'évaluer un produit individuel pour s'assurer qu'il se conforme aux exigences d'un standard préétabli; (c) l'émission d'un certificat qui déclare la conformité à un standard préétabli et en concordance aux articles (a) ou (b) mentionnés dans ce paragraphe (RTCA/DO-178B, 1992).

Critères de transition (« Transition Criteria ») : les conditions minimales devant être satisfaites pour le passage d'un processus à autre. Les critères de passage sont définis par le processus de planification de logiciel (RTCA/DO-178B, 1992).

État hasardeux (« Hazardous state ») : possibilité du dysfonctionnement d'une composante soit du logicielle ou de système (Leveson, 1986).

Exigences dérivées (« Derived Requirements ») : exigences additionnelles dérivées du processus de développement du logiciel. Elles ne sont pas traçables directement aux exigences de haut niveau (RTCA/DO-178B, 1992).

Exigences de haut niveau (« High Level Requirements ») : exigences du logiciel développées à partir de l'analyse des exigences du système, des exigences de sûreté et de l'architecture du système (RTCA/DO-178B, 1992).

Exigences de bas niveau (« Low Level Requirements ») : exigences du logiciel dérivées des exigences de haut niveau, à partir desquelles le code source peut être directement implémenté sans informations supplémentaires (RTCA/DO-178B, 1992).

Fiabilité (« Reliability ») : probabilité qu'un logiciel effectue la tâche qui lui est assignée dans une période donnée et dans un environnement donné (IEEE Std 982.2, 1988).

Structure (« Structure ») : organisation spécifique ou une interrelation des parties d'un tout (RTCA/DO-178B, 1992).

Sûreté (« Safety ») : mesures à prendre afin d'assurer que des désastres, catastrophes ou des dommages ne se produisent pas lors d'un état hasardeux qui affecte un système (Leveson, 1986).

Sûreté du système (« System Safety ») : l'état de sûreté de tout les composants que comprend le système (inclus le logiciel).

Sûreté du logiciel (« Softwate Safety ») : logiciel qui est exécuté sans créer un état hasardeux dans le contexte de système dont il fait partie (Leveson, 1986).

Sûreté versus fiabilité (« Safety vs. Reliability ») : la fiabilité est concernée par la production d'un système sans erreur tandis que la sûreté est concernée par la production d'un système sans état hasardeux, autrement dit sans danger (Singh, 1999).

Sûreté versus Sécurité (« Safety vs. Security ») : la sûreté est concernée par les risques et les menaces à la vie humaine et par les dommages et catastrophes matérielles. La sécurité est

concernée par les risques et les menaces à la confidentialité, l'intégrité de la vie privée. Il faut noter que les deux termes ont des points en commun (Singh, 1999).

INTRODUCTION

L'informatique et les télécommunications influencent notre vie et les logiciels sont omniprésents dans notre quotidien (par exemple : les voitures, les avions, les appareils médicaux, les centrales électriques et nucléaires, etc.). Dowson (Dowson, 1993) et Mayadoux (Mayadoux, 1994) expliquent que certains logiciels sont de plus reconnus comme étant « critiques » car leur mauvais fonctionnement pourraient occasionner des conséquences catastrophiques pour la vie humaine (ces logiciels sont appelés aussi de « sûreté critique »). Il est donc clair que des projets informatiques sont devenus progressivement plus sophistiqués, plus complexes et avec des exigences plus grandes de fiabilité et de sûreté (Dettmer, 1988).

Dans ce travail de recherche nous sommes préoccupés spécifiquement par certaines caractéristiques importantes du logiciel critique. En particulier nous sommes intéressés par les caractéristiques qui établissent leur processus de Vérification et Validation (V&V), par le contrôle des coûts associés pour l'implantation de ces processus de V&V, ainsi que par les exigences de certification nécessaires pour qu'il soit apte à être déployé dans contexte particulier.

Ils existent plusieurs critères importants associés à la qualité du logiciel critique et en particulier il est important de noter le critère de fiabilité. Kornecki et al. (Kornecki, 1999) raisonnent de la façon suivante par rapport à la fiabilité du logiciel : pour arriver à obtenir la haute fiabilité dans le logiciel, il est nécessaire d'implémenter la fiabilité des exigences, pour implémenter la fiabilité des exigences il faut les spécifier et les documenter, pour garantir que les exigences ont été implémentées de façon appropriée il faut vérifier, valider et certifier autant les produits que les processus.

Une autre composante importante pour arriver à améliorer la fiabilité du logiciel, selon certains auteurs, c'est la rigueur dans l'implémentation des processus de construction du logiciel critique. Benediktsson, Hunter et al (Benediktsson, Hunter et al. 2001) dans leur article « Processes for Software in Safety Critical Systems » mentionnent qu'il est possible

que dans beaucoup de cas où la défaillance du logiciel dans les systèmes est la cause des désastres, ceux-ci pourraient avoir été évités si des processus plus rigoureux avaient été employés dans le développement du logiciel pour de tels systèmes. Ceci s'appliquerait à toutes les parties du processus de développement de logiciel.

Eastaughffe, Cant et al (Eastaughffe, Cant et al. 1999) signalent que, dans la pratique, la sûreté totale des systèmes critiques n'existe pas mais que les organisations en charge de développer ce type de systèmes ont l'obligation d'établir et suivre des processus et activités avec le but d'améliorer la sûreté des systèmes pour garantir leur fonctionnement correct.

Dunn et Corliss (Dunn et Corliss, 1990) de la « NASA Ames Research center » ont signalé que les méthodes classiques pour assurer la sûreté des logiciels que l'on retrouve souvent dans l'industrie (comme par exemple : les pratiques dans le cycle de vie du logiciel, les tests, entre autres) ne remplissent pas toujours les caractéristiques nécessaires pour combler les besoins de qualité à un degré rigoureux et à un coût acceptable pour les organisations. Donc une application intensive de la V&V devient nécessaire pour ces projets.

Cette thèse porte un intérêt spécial sur l'implantation des activités de la V&V dans les projets. Feather (Feather et Markosian, 2006) du « NASA Jet Propulsion Laboratory » mentionnent qu'étant donné la nature évolutive de développement du logiciel, la V&V impose des défis en termes de l'adaptation des activités de V&V pour les projets. Boehm (Boehm, 1984) explique que les objectifs spécifiques de la V&V pour chaque projet doivent faire l'objet d'une analyse particulière. Ces objectifs dépendront de la criticité du logiciel, de ses contraintes et de sa complexité. Les activités de V&V constituent un élément clé du processus de développement du logiciel critique car ces activités vont impacter directement la maîtrise des coûts, de la qualité des livrables et des délais. Les activités de V&V proposent aussi des enjeux en termes de la rigueur dans leur définition et de leur implantation dans les projets (Jenkins, Deshpande et al. 1998).

Zelinski (Zelinski, 1995) explique que la V&V, et particulièrement lorsqu'elle est exécutée par un groupe indépendant (i.e. la vérification et validation indépendante - V&VI), constitue un accélérateur pour améliorer la qualité du logiciel et les coûts. La V&VI doit être exercée comme un processus formalisé et répétitif afin de maximiser son efficacité. Zelinski mentionne qu'une façon d'arriver à ces résultats est d'employer une méthodologie qui établirait formellement une correspondance entre les activités de V&V et les processus du développement. Cette méthode devrait inclure des mesures pour faire le suivi et l'évaluation des résultats.

Dans cette recherche également nous portons principalement l'attention sur le contrôle et le suivi des coûts et des efforts de V&V dans la construction du logiciel critique. Thomas et Dowling (Thomas et Dowling, 1982) expliquent qu'il existe une préoccupation dans l'industrie et dans la recherche par rapport aux coûts de la V&V dans le développement des logiciels critiques. Dépendamment de la rigueur des objectifs de certification requise par le système pour le logiciel, les coûts de la V&V pourraient représenter jusqu'à 50% des coûts globaux du projet (Kit, 1995). Robillard, Labrèche *et al* (Robillard, Labrèche et al. 2007), signalent le défi pour les organisations en charge de développement du logiciel critique et particulièrement pour le secteur aéronautique, de l'utilisation adéquate de la V&V afin de minimiser les coûts de son implantation et combler les objectifs du guide DO-178B (RTCA/DO-178B, 1992) .

Un volet considéré important dans le logiciel critique est leur certification, c'est-à-dire leur autorisation par une autorité compétente à être déployé sans danger. Bertrand et Fuhrman (Bertrand et Fuhrman, 2008) ont remarqué que les compagnies dédiées à la production du logiciel critique expérimentent difficultés pour adapter les objectifs de la vérification et validation pour ces projets. Et concrètement c'est le cas des organisations qui ont besoin de la certification avec le guide RTCA DO-178B (RTCA DO-178B, 1992). Ce guide est imposé par le FAA (Federal Aviation Administration - USA) pour la construction des logiciels qui seront embarqués dans les équipements des aéronefs. Il faut noter que le guide DO-178B prescrit des objectifs précis pour attendre la certification du logiciel mais il est flexible au

niveau des pratiques concrètes, en termes par exemple des cycles de vie, ou activités que les organisations devront suivre pour combler ces objectifs (RTCA/DO-178B).

Notre recherche traite également de la possibilité d'implanter des pratiques de V&V dans un contexte de la construction du logiciel critique avec une approche itérative et incrémentale, telle que présentée dans les Processus Unifiés (Jacobson et al, 1999). Bertrand et Fuhrman (Bertrand et Fuhrman, 2008), expliquent que la version libre des Processus Unifies appelée «OpenUP» (Balduino, 2006) offre une alternative pour combler les besoins concrets des pratiques du développement pour le logiciel critique tel que demandé par la DO-178B. Cette version des processus unifiés présente un ensemble des processus, procédures, rôles, activités, et guides pour la construction du logiciel en général. Ces pratiques pourraient être adaptées pour les exigences de la DO-178B. L'OpenUP, explique Balduino (Balduino, 2006), présente aussi la flexibilité d'être adaptable selon les besoins des projets. Cette flexibilité d'adaptation est fournie par le composant EPFC (Eclipse Process Framework Composer). EPFC permet d'ajouter, modifier, éliminer des processus, rôles, activités, tâches et les produits (« workproducts ») qui sont définis dans OpenUP.

Concrètement nous avons identifié certaines lacunes pour la construction du logiciel critique dans le contexte aéronautique selon le guide DO-178B, à savoir : dans l'implantation des processus de la V&V, dans le contrôle et le suivi des activités de V&V et dans la certification.

Dans ce travail de recherche nous proposons combler les lacunes par rapport à l'implantation des processus de la V&V et la certification du logiciel critique, par le biais de la proposition d'un référentiel des processus de V&V basé sur les Processus Unifies (identifiés dans la version libre appelée l'OpenUP) et conforme aux exigences de certification du guide DO-178B pour le secteur aéronautique.

Pour combler les lacunes identifiées par rapport au contrôle et au suivi des activités de V&V dans les projets, nous proposons un entrepôt de mesures basé sur un modèle générique pour construire les données des mesures.

Notre proposition de recherche sera évaluée dans plusieurs contextes. Le référentiel des processus de V&V sera comparé par rapport aux livrables de nos partenaires industriels comme CMC Electronics Inc., Verocel Inc. et l'entrepôt des données de mesures sera testé en prototype avec des données de projets d'Ericsson Research Canada.

Il faut préciser que l'identification de la problématique et des objectifs de recherche de notre proposition de thèse sera raffinée sur une base itérative et incrémentale : itérative car nous allons revenir questionner et mieux repositionner nos résultats de recherche dans chaque itération, et incrémentale car dans chaque itération nous allons améliorer notre compréhension du domaine de connaissances faisant ainsi évoluer notre problématique et nos objectifs de recherche.

Organisation de la thèse

Cette thèse est composée d'une introduction et de 10 chapitres. L'introduction présente une vue d'ensemble de quelques aspects reliés à la V&V et au logiciel critique. La revue de la littérature est présentée dans les chapitres de 1 à 4.

Le premier chapitre présente des concepts utilisés dans le cadre du logiciel critique, la vérification et validation. Le chapitre introduit également la vérification et validation indépendante. Une révision des standards du logiciel critique est également présentée et la norme IEEE 1012 (IEEE Std 1012, 2004) est présentée avec plus de détail.

Le deuxième chapitre est consacré à la présentation du guide RTCA DO-178B. Il explique entre autres les concepts relatifs aux niveaux de défaillance, niveaux de criticité, traçabilité, aspects de la certification, méthodes de vérification, etc.

Le troisième chapitre présente les processus logiciels, leur amélioration et leur formalisation avec SPEM (Software Process Engineering Metamodel), (OMG 2008). Les processus unifiés et d'autres sont aussi présentés. Des outils pour la formalisation des processus sont également cités comme par exemple : RUP XDE (Bencomo, 2005), Eclipse Process Framewrok Composer (EPFC) (Haumer Peter, 2007), etc.

Le quatrième chapitre introduit les concepts de mesure et l'importance des mesures pour les activités de V&V dans les projets du logiciel critique. Il présente une étude et une analyse du standard ISO 15939 (ISO/IEC Std 15939) ainsi que certaines technologies pour la création de l'entrepôt dans le contexte OLAP (On-Line Analytical Processing).

Le cinquième chapitre explique l'approche identifiée pour atteindre les objectifs qui guident ce travail de recherche. La problématique de la thèse ainsi que les objectifs de recherche sont décrits dans ce chapitre. Le chapitre explique également les différentes activités du travail de recherche et présente des diagrammes à différents niveaux utilisés pour illustrer les intrants, les processus et les extrants de chacune des phases de cette recherche. Le chapitre explique aussi la participation des partenaires industriels.

Le sixième chapitre présente des analyses préliminaires de ce travail de recherche. Nous présentons différents types de modélisations dans différents contextes et avec différents outils. Ce chapitre comporte une analyse des différents aspects pour le développement de logiciels critique selon le guide DO-178B et aussi la modélisation de la norme IEEE 1012 (IEEE Std 1012, 2004) avec le RUP XDE (Bencomo, 2005).

Le septième chapitre analyse et propose des pratiques de vérification pour les exigences du logiciel selon les objectifs de la DO-178B. Cette analyse servira pour établir les spécifications pour la construction du plugiciel (« Plug-in ») des pratiques de V&V cohérent avec OpenUP et le guide DO-178B. Ce chapitre présente l'analyse et la proposition des pratiques de V&V pour les exigences du logiciel critique.

Le huitième chapitre présente la modélisation des pratiques de V&V avec l'outil «Eclipse Process Framework Composer». La modélisation sera basée sur des spécifications (pratiques) déterminées dans le chapitre 7. La construction et l'implémentation du plugiciel sont aussi présentes dans ce chapitre.

Le neuvième chapitre présente l'entrepôt générique des mesures pour les projets de V&V. Il explique la conception du modèle ainsi que sa construction en utilisant la technologie OLAP. L'entrepôt devrait faciliter le contrôle et le suivi des processus et produits décrits dans le plugiciel pour la V&V

Le dixième chapitre présente l'évaluation des résultats de recherche, incluant les différentes méthodes utilisées pour vérifier et valider les résultats de recherche du référentiel des processus de V&V et leurs mesures pour les projets de logiciels critiques.

La conclusion générale présente une synthèse des contributions de recherche et signale les impacts des contributions dans le cadre des projets d'implantation de la V&V pour les logiciels critiques.

Publications de recherche

Nous avons publié au niveau international un ensemble d'articles dans le cadre de cette recherche. Ces publications portent sur la vérification et validation, les mesures et l'entrepôt des mesures, ainsi que l'amélioration des processus du logiciel. La liste des publications est présentée dans l'annexe I de cette thèse.

CHAPITRE 1

LE LOGICIEL CRITIQUE ET LA VÉRIFICATION & VALIDATION

1.1 Introduction

Ce chapitre introduit les concepts du logiciel critique et l'importance de la V&V dans le contexte des systèmes critiques en général et du logiciel critique en particulier. Nous expliquons les diverses interprétations ainsi que les différences entre la V&V et la V&VI. Nous allons également présenter les perspectives de la V&V en concordance avec les standards IEEE 1012, AFNOR, RTCA/DO-178B entre autres. Nous analysons le cycle de vie de la V&V proposé par la norme IEEE 1012.

1.2 Démarche pour ce chapitre

Dans cette section nous présentons les activités à accomplir afin d'identifier notre problématique et objectifs de recherche par rapport à la création d'un référentiel des processus de V&V.

Le contenu détaillé de la revue de la littérature pour le référentiel des processus de V&V est présenté dans les chapitres 1 à 3. Dans les paragraphes suivants nous présentons un aperçu des activés de la revue de la littérature. Voir la figure 1.1.

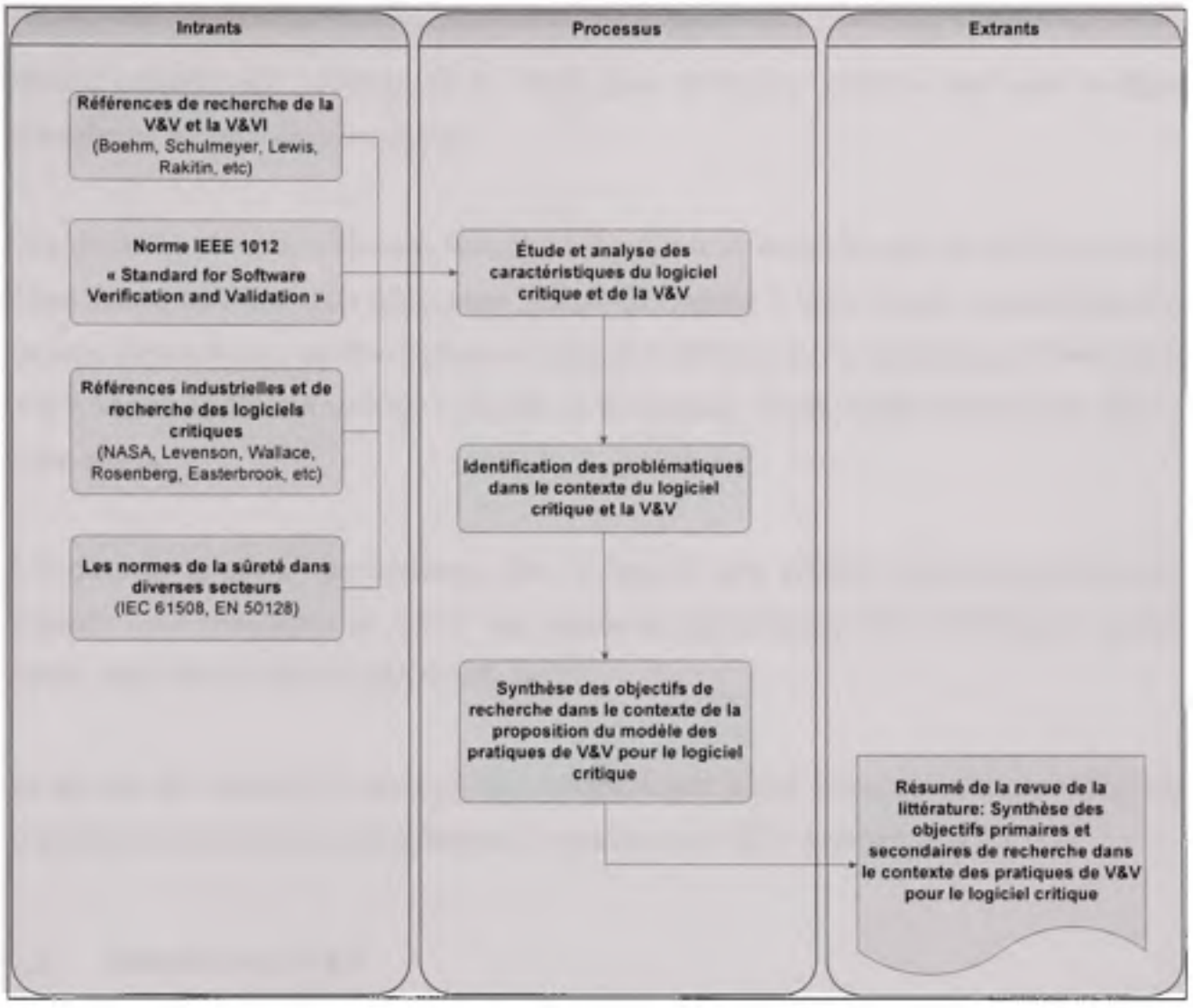

Figure 1.1 ***Revue de la littérature : Le logiciel critique et la V&V.***

Nos activités de revue de la littérature débutent avec l'étude des publications (livres, articles et autres références), produits par divers auteurs qui ont travaillé, soit par le biais de la recherche ou de l'expérience pratique, sur la V&V et la V&V indépendante (V&VI). Nous explorons les divers concepts et interprétations de la V&V adaptés selon les phases du développement du logiciel critique. Nous consulterons entre autres les auteurs suivants: (Schulmeyer et MacKenzie, 1999), (Walters, 2000), (Rakitin, 2001), (Wallace et Fujii, 1989a), (Lewis, 1992).

Une section est réservée à une analyse de la norme IEEE 1012 (IEEE Std 1012, 2004). Cette section présente les processus de la V&V pour le logiciel critique ainsi que la façon d'implanter la V&V dans les projets.

Une étude des diverses références sera présentée en vue de traiter le sujet du logiciel critique. Nous ferons référence aux cas comme Therac 25, Ariane 5 entre autres. Les opinions des experts sur les risques du développement logiciel et la façon de les atténuer par le biais de la V&V seront également incluses : (Vanek et Rosenberg, 2000), (Zelkowitz et Rus, 2001), entre autres.

L'impact de la sûreté des systèmes dans le logiciel sera présenté dans une section dans laquelle nous présentons un survol des standards internationaux IEC 61508 (IEC 61508, 2005), AFNOR EN 50128 (EN-50128, 2001).

Le résultat de l'analyse (le résumé) de ce chapitre sera utilisé comme intrant pour bâtir nos objectifs de recherche pour le référentiel de pratiques de V&V pour le logiciel critique.

1.3 Définitions de V&V

Pressman (Pressman, 1997) indique que la vérification est l'ensemble d'activités qui assurent que le logiciel implémente correctement une fonction spécifique, et la validation est l'ensemble d'activités qui assurent que le logiciel qui a été construit correspond aux attentes du futur utilisateur.

Boehm (Boehm, 1981) mentionne succinctement la différence *entre vérification et validation* en ces termes: Vérification: « Construisons-nous bien le produit? et la Validation: « Construisons-nous le bon produit? ».
Quelques définitions dans la littérature par rapport à la vérification et validation sont mentionnées dans les publications suivantes : IEEE (IEEE Std 610.12, 1990), US Department of Defense (DoD, January 4, 1994.), O'Keefe *et al.* (Robert M. O'Keefe, 1987),

Adrion *et al.* (W. R. Adrion, 1982), Pierce (Pierce, 1996), (Schulmeyer et MacKenzie, 1999). En général ces définitions coïncident dans le sens que la vérification et la validation sont des activités complémentaires et qu'elles ont comme but de stimuler la qualité du logiciel pendant le cycle de vie de développement.

Lewis (Lewis, 1992) ajoute ce sens :

> "Verification is an iterative process aimed at determining whether the product of each step in the development cycle (a) fulfills all the requirements levied on it by the previous step and (b) is internally complete, consistent, and correct enough to support the next phase." (Lewis, 1992).
>
> "Validation is the process of executing the software to exercise the hardware and comparing the test results to the required performance." (Lewis, 1992).

Autrement dit, la vérification est une activité qui évalue les produits de chacune des phases du cycle de développement et la validation permet de s'assurer que les exigences logicielles telles qu'exprimées par le client sont adéquatement testées.

Rakitin (Rakitin, 2001) dans son livre « Software Verification and Validation : A Practitioner's Guide» mentionne que la vérification implique typiquement des revues pour l'évaluation des documents, des plans, le code, des exigences et des spécifications. Cette vérification peut être faite avec des listes de contrôle, des walkthroughs et des réunions d'inspection. La validation implique typiquement l'essai réel. La validation a lieu après que des vérifications soient accomplies. On évalue par exemple les fonctionnalités de l'application, si l'exigence du client est incorporée ou pas dans le module ou l'application, etc.

La V&V emploie différents types d'analyse afin d'assurer que les efforts du développement logiciel ont été un succès (Robillard, Labrèche et Gendreau, 2007). L'analyse se fait par le

biais des différentes techniques que la V&V utilise tout au loing du cycle de vie du développement logiciel (Wallace et Fujii, 1989b).

1.4 La V&V et le logiciel critique

Les logiciels deviennent de plus en plus complexes et les facteurs qui compliquent leur réalisation sont nombreux: la composition hétérogène sur une diversité de plateformes, l'exécution distribuée, la complexité dans des algorithmes de calcul, la multiplicité d'entrepreneurs avec des méthodologies diverses de développement, etc. La croissance de sophistication et de complexité du logiciel augmente les risques, incrémentant les coûts de développement ainsi que les coûts de maintenance. Certains logiciels dits critiques sont plus susceptibles, s'ils ne sont pas d'un certain niveau de qualité, de mettre en péril la vie humaine et/ou mettent à risques des grands investissements financiers.

Le type de logiciel qui, directement ou indirectement, assure la sûreté de la vie humaine ou des investissements significatifs, est désigné sous le nom de « logiciel critique » (ou de sauvegarde critique ou de sûreté-critique). Ce type de logiciel doit atteindre des niveaux très élevés de la sûreté et de la fiabilité et répondre à des standards exigeants de qualité. Par conséquent, le procédé de développement pour ces logiciels doit être étroitement contrôlé, en raison du niveau très élevé de la qualité exigée (Wallace et Fujii, 1989a). En effet, beaucoup d'accidents provoqués par une qualité déficiente du logiciel critique ont été rapportés dans la littérature : Therac 25 (Leveson et Turner, 1993), Ariane 5 (Kunzig, 1997), et autres (Williamson, 1997).

Pour le logiciel critique, la réduction de risque est aussi très importante. Beaucoup de projets de logiciel, par exemple chez la NASA (Zelkowitz et Rus, 2001), (Vanek et Rosenberg, 2000) et ailleurs, comportent des activités de V&V du logiciel pour atténuer certains risques de développement de logiciel (Singh et al., 1995).

Le développement du logiciel dans des domaines critiques comme le contrôle en temps réel (ex. dans l'avionique, les transports, le contrôle dans les centrales nucléaires, etc.), pose évidemment les problèmes de la vérification et de la validation (V&V) de ces systèmes (Bruner, 2002). La V&V peut représenter jusqu'à cinquante pour cent du budget dans les projets critiques du logiciel (Kit, 1995), d'où l'intérêt en général dans l'industrie et la recherche d'améliorer les pratiques et l'impact sur le coût d'implantation de la V&V dans les projets.

Les objectifs spécifiques de la V&V pour chaque projet doivent faire l'objet d'une analyse particulière. Ces objectifs spécifiques dépendront de la criticité du logiciel, de ses contraintes et de sa complexité. Les objectifs de la V&V vont déterminer le type de technique de V&V qui serait implémentée par phase dans le projet afin de s'assurer que le produit satisfait aux exigences des besoins de l'utilisateur. L'étendue de la V&V doit aussi permettre de s'assurer que le produit correspond aux attentes du client dans les domaines des exigences non-fonctionnelles telles que: performance, sécurité, portabilité, maintenance, sûreté, entre autres (ISO/IEC Std 9126-1 to 4, 2001).

Kornecki (Kornecki, 1999) mentionne qu'il est impératif pour le logiciel critique de passer par un rigoureux processus de V&V. Il est clair dans ce cas que l'exécution de la V&V peu être mieux exercée dans un contexte d'indépendance entre des équipes de V&V et les développeurs. L'intention est d'établir une interaction et un feedback entre les groupes de V&V et les groupes de développement qui définissent des changements au niveau de processus de développement, des décisions des produits, et des priorités.

Une équipe de V&V indépendante (V&VI) est à l'extérieur du chemin critique de l'effort du développement logiciel. Ces groupes ont plus de possibilités pour l'expérimentation des techniques que les développeurs (Easterbrook, 1999).

1.4.1 La vérification et validation indépendante (V&VI)

La V&VI est implémentée quand une organisation estime qu'il est nécessaire de disposer d'un groupe spécialisé pour l'analyse de la V&V. Ce groupe devrait travailler sur l'analyse des produits et processus du développement logiciel. L'analyse est exécutée en parallèle au processus du développement logiciel de l'organisation. Cette analyse de caractère indépendant pourrait être faite pour un sous-traitant externe à l'organisation. Il faut préciser que cette analyse indépendante de V&V ne remplace pas l'analyse de V&V qui devrait être faite par des groupes internes à l'organisation (Easterbrook et Callahan, 1996).

Arthur (Arthur et Nance, 2000) mentionne parmi les bénéfices de la V&VI la détection précoce des erreurs, ce qui entraîne une réduction des coûts pour les enlever. Donc, la V&VI encouragerait un processus de développement plus contrôlé.

La vérification et validation indépendante est adoptée lorsque le logiciel critique a besoin d'un niveau d'évaluation et de contrôle plus autonome des autres équipes du projet. L'idée d'appliquer la V&V indépendante est de réduire le risque du développement. Les équipes de V&V indépendante devraient fournir des informations sur le projet d'un point de vue impartial et faire des évaluations entre autres des coûts et des échéanciers des produits pendant le processus du développement.

Le guide DO-178B et le standard IEEE 1012 exige l'introduction de la V&VI pour certains projets où la défaillance du logiciel pourrait occasionner une catastrophe.

Selon Lewis la V&VI :

> *"Independent Verification and Validation (NASA Software IV&V Facility) is a series of technical and management activities performed by someone other than the developer of a system to improve the quality and reliability of that system and to assure that the delivered product satisfies the user's operational needs."* (Lewis, 1992).

Lewis introduit la notion d'indépendance. Il mentionne que pour avoir une vraie V&VI, Il faut de plus distinguer trois principales approches d'indépendance : indépendance technique, managériale et financière. Ces concepts sont aussi mentionnés dans le standard IEEE 1012 (IEEE Std 1012, 2004), à savoir :

- **Indépendance technique** implique que les activités de V&V ne sont pas exécutées par le même personnel impliqué dans le développement du produit. L'équipe de V&VI doit développer sa propre compréhension du problème ainsi que ses propres solutions pour résoudre le problème.
- **Indépendance managériale** requiert que les activités de V&VI ne dépendent pas de la même organisation que l'organisation responsable de produire le système.
- **Indépendance financière** exige que les budgets de V&VI soient séparés des budgets de développement. Ceci pour prévenir que des fonds alloués à la V&VI ne soient détournés vers d'autres fins à cause des pressions externes ou financières.

Dans l'annexe C du standard IEEE 1012, il est aussi mentionné d'autres formes d'indépendance :

- **V&VI classique,** lorsque les trois paramètres sont respectés (indépendance technique, managériale et financière).
- **V&VI modifié,** lorsque l'indépendance est technique et financière.
- **V&VI intégré,** lorsque l'indépendance est financière et managériale.
- **V&VI interne,** lorsque l'indépendance (financière, technique et managériale) n'est pas établie de façon explicite dans le projet. La V&V est exercée par un groupe interne à l'organisation. La norme IEEE 1012 recommande que le groupe de V&V doit être différent du groupe du développement.
- **V&VI embarqué**, similaire à la V&VI interne. Ses livrables sont les mêmes que aux du processus du développement. L'indépendance est compromise à trois niveaux, à savoir: technique, financier et managérial.

1.5 La V&V dans le standard IEEE 1012

La norme IEEE 1012 donne les définitions suivantes :

Vérification : (a) Le processus d'évaluer un système ou un composant pour déterminer si les produits d'une phase de développement donnée remplissent les conditions imposées au début de cette phase. (b) Le processus de fournir l'évidence objective que le logiciel et ses produits associés répondent aux exigences (par exemple, pour l'exactitude, la complétude, la consistance, l'exactitude) pour toutes les activités et processus du cycle de vie et satisfaites également les normes, les pratiques, et les conventions pendant les processus.

Validation : (a) Le processus d'évaluer un système ou un composant pendant ou à la fin du processus de développement pour déterminer s'il répond à des exigences définies. (b) Le processus de fournir l'évidence que le logiciel et ses produits associés répondent à des exigences de système alloués au logiciel[1] à la fin de chaque activité de cycle de vie. Résoudre le problème correctement et satisfaire les exigences de l'utilisateur.

En général, les processus de V&V évaluent le logiciel dans le cadre du système, y compris l'environnement opérationnel, le matériel, l'interface logiciel-système, les opérateurs et les utilisateurs. Les processus de V&V déterminent si le logiciel répond aux exigences du logiciel et du système. Cette détermination peut inclure : l'analyse, l'évaluation, la revue, l'inspection, et les tests.

La norme IEEE 1012 est assez flexible pour permettre l'adaptation (« tailoring ») des processus de V&V pour les divers cycles de vie définis dans les projets. Cette norme décrit des processus, des activités, des tâches et des artefacts (intrants et extrants) à accomplir dans

[1] En anglais « System Requirements Allocated To Software (SRATS) »

le contexte du cycle de vie de la V&V. Elle spécifie de façon détaillée des artefacts de V&V à livrer tôt dans le cycle de vie du logiciel : ceci est illustré dans figure 1.2.

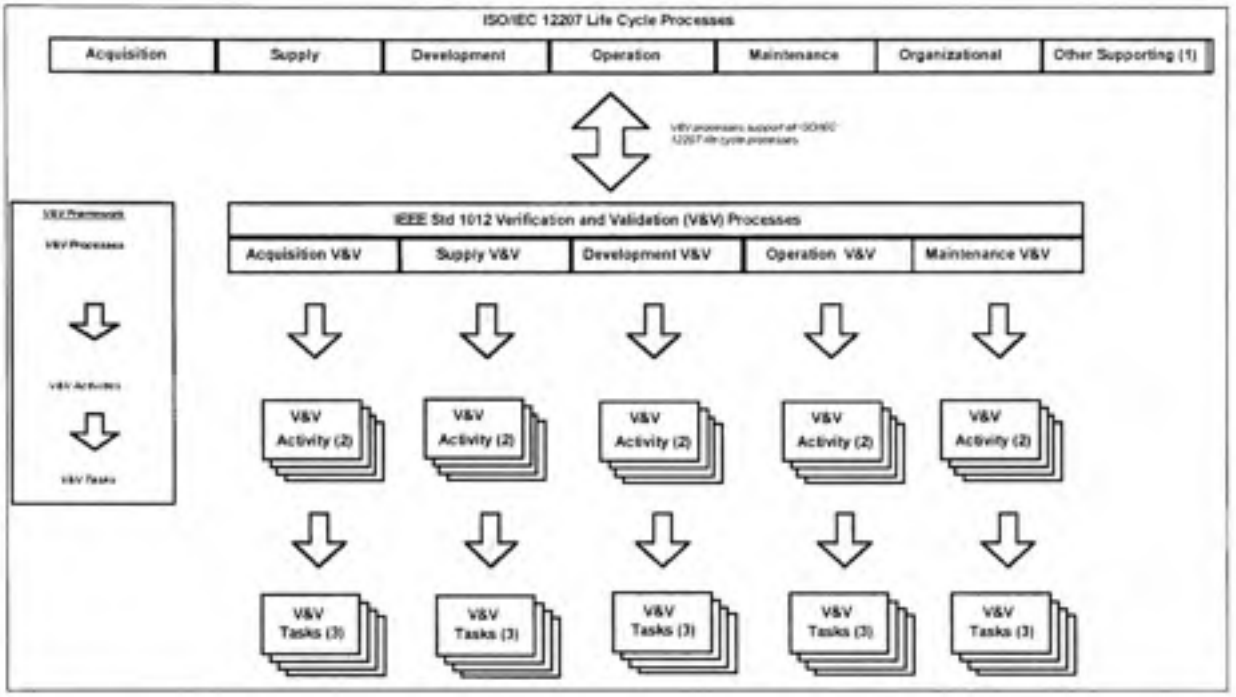

Figure 1.2 ***Organisation des processus de V&V selon la norme IEEE 1012.***

Les processus de V&V décrits dans la norme IEEE 1012 supportent les 6 processus primaires de la norme IEEE 12207 (IEEE/EIA 12207, 1998) pour le cycle de vie du logiciel, à savoir :

1. Processus de Gestion;
2. Processus d'Acquisition;
3. Processus d'Approvisionnement;
4. Processus de Développement;
5. Processus d'Opération;
6. Processus de Maintenance.

Il est spécifié dans l'IEEE 1012 que les activités et les artefacts pour les projets de V&V doivent être définis dans le cadre d'un plan de vérification et validation pour le projet. La norme IEEE 1012 propose une guide pour l'élaboration d'un plan de vérification et validation du logiciel, le standard IEEE 1059 (IEEE Std 1059, 1993), (« SVVP - Software

Verification and Validation Plan »). C'est dans le SVVP que les particularités des processus, activités, tâches, artefacts ainsi que des rôles doivent être indiqués pour chaque projet de V&V spécifique (Walters, 2000).

1.6 Le logiciel critique et les standards dans diverses secteurs

Étant donné que les logiciels critiques sont embarqués dans les systèmes qui demandent une haute fiabilité, ils imposent une rigueur particulière pour leur construction. L'industrie aéronautique a réussi à définir des normes pour améliorer la qualité de la réalisation de ces types du logiciel. Dans cette section nous présentons quelques unes des normes pour le logiciel critique.

1.6.1 Le contexte ferroviaire (Norme AFNOR EN - 50128)

L'industrie des trains a adopté le standard « EN 50128 Railway applications – Communications, signaling and processing systems – » (50128:2001, 2001). Ce standard fournit un cadre cohérent et logique pour le développement des systèmes critiques. Les caractéristiques principales de la norme sont :

- Comporte toutes les phases essentielles du cycle de vie des systèmes critiques (depuis la conception jusqu'à la maintenance), dans un contexte soit électrique, électronique, et électronique programmable. Introduit une culture de sûreté dans les systèmes.
- Conçue avec un concept du développement technologique rapide (« rapidly developing technology »).
- Fournit une méthode pour le développement des spécifications des exigences de sûreté nécessaires pour implémenter la sûreté fonctionnelle requise pour la sûreté des systèmes.
- Utilise les niveaux d'intégrité pour définir les spécifications de sûreté des systèmes.
- Comporte une approche statistique pour la définition des risques pour la détermination des niveaux d'intégrité des systèmes.

- Établit des différences entre modes de défaillance et demande des prévisions pour les défaillances non-détectées. Ces modes de défaillance comportent un impact direct sur les niveaux d'intégrité des systèmes.

1.6.2 Le contexte générale : Norme IEC 61508 (IEC 61508, 2005)

Cette norme est de caractère générique dans le contexte de la sûreté des systèmes. Elle est composée de directives par rapport aux exigences de sûreté. Cette norme est considérée comme une référence des bonnes pratiques de sûreté des systèmes électroniques. La norme précise aussi un cycle de vie de seize étapes qui intègrent le logiciel au système. Elle présente des directives pour classifier les systèmes en fonction des quatre niveaux d'intégrité de la sûreté (A à D). La norme peut être adaptée dans divers domaines avec diverses composants, à savoir : électromécaniques, électroniques, électrotechnique, systèmes informatisés, entre autres.

La norme IEC 61508 comporte sept parties, à savoir :

- Partie 1 : Les exigences générales;
- Partie 2 : Les exigences pour les systèmes de sûreté électriques programmables, électroniques et électriques;
- Partie 3 : Les exigences logicielles;
- Partie 4 : Définition et abréviations;
- Partie 5 : Exemples de méthodes pour la détermination des niveaux d'intégrité de sûreté;
- Partie 6 : Directives sur l'application du IEC 61508-2 et du IEC 61508-3;
- Partie 7 : Vue d'ensemble des technique et des mesures.

1.7 Sommaire

Ce chapitre a fait un survol des concepts du logiciel critique et de la sûreté dans les systèmes alloués au logiciel. Nous avons expliqué que les logiciels critiques jouent un rôle très

important dans le fonctionnement des systèmes appelés critiques. Nous avons montré les caractéristiques de V&V et de V&VI ainsi que des processus qui sont reliés pour son implantation. Nous avons présenté la norme IEEE 1012 et les divers concepts de cycle de vie de la V&V. Nous avons également mentionné le besoin de la planification de la V&V.

CHAPITRE 2

LA CONSTRUCTION ET CERTIFICATION DU LOGICIEL CRITIQUE SELON LE GUIDE DO-178B

2.1 Introduction

Ce chapitre introduit le guide DO-178B et explique entre autres les concepts relatifs aux niveaux de défaillance, niveaux de criticité, traçabilité, aspects de la certification, méthodes de vérification, etc. Il présente également une analyse par rapport aux processus du développement du logiciel critique décrits dans le guide DO-178B. Cette analyse des processus facilitera l'identification des différents produits de travail (artefacts, livrables, résultats) et associations impliqués dans chaque processus du guide. Les résultats de l'analyse permettront une définition appropriée des activités qui seront à inclure dans notre référentiel pour la V&V du logiciel critique.

2.2 Démarche pour ce chapitre

Cette section présente les activités à accomplir afin d'identifier notre problématique et les objectifs de recherche par rapport à notre référentiel des processus de V&V.

La figure 2.1 montre qu'au début nous allons étudier et analyser les caractéristiques et l'identification de la structure et recommandations du guide DO-178B (RTCA/DO-178B, 1992) ainsi que le document explicatif DO-248B (RTCA/DO-248B, 2001). Ces guides décrivent les exigences ainsi que des recommandations pour certifier le logiciel critique. Nous allons étudier et analyser les divers objectifs, processus, activités de la V&V décrits dans la norme. Nous allons aussi produire des résumés qui vont aider pour la construction de la problématique et objectifs de recherche pour le modèle des processus que nous allons proposer dans cette thèse. A la fin de cette démarche il est introduit une synthèse d'ensemble des objectifs primaires et secondaires de la proposition.

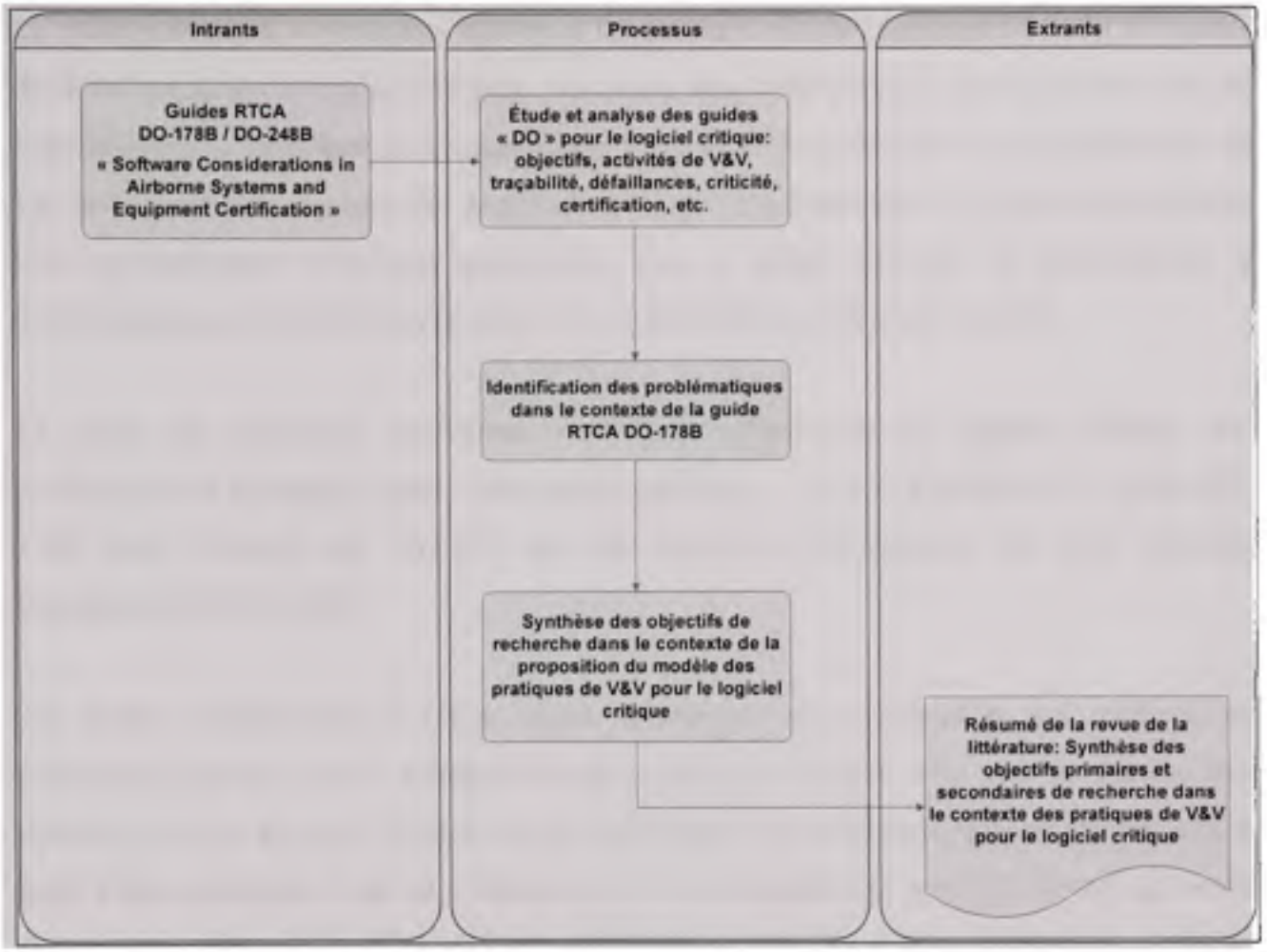

Figure 2.1 ***Revue de la littérature : Construction et certification selon le DO-178B.***

2.3 L'objectif du guide DO-178B

Le guide DO-178B (RTCA/DO-178B, 1992), intitulé "Software Considerations in Airborne Systems and Equipment Certification," fournit les directives pour le logiciel en voie de développement pour les systèmes aéronautiques. Le but du guide DO-178B est de fournir les directives détaillées pour la production du logiciel pour les systèmes aéroportés.

Le guide DO-178B est un standard pour le développement des logiciels critiques dans le secteur aéronautique (Carolyn Salmon et Clive Lee, 2006). Le guide est focalisé sur les aspects de la sûreté d'un point de vue des processus du logiciel. Les directives du guide sont la base de la certification des logiciels embarqués dans les équipements aéronautiques.

Le guide présente les principaux acteurs et les artefacts qui sont impliqués dans le processus de développement du logiciel critique. Le guide DO-178B propose que la construction du logiciel doive se faire dans un contexte des systèmes ; il n'impose pas un cycle particulier de vie du logiciel mais propose des processus qui sont communs dans la plupart des cycles de vie. Les principaux processus mentionnés dans le guide incluent : la planification, le développement, la vérification, la gestion de configuration, l'assurance qualité.

Le guide est applicable également à d'autres applications du logiciel critique (ex. environnement industriel, santé, automoteur, nucléaire, etc.). L'équivalent du guide DO-178B pour l'Europe est : ED-12B est "the European Organization for Civil Aviation Equipment (EUROCAE)".

Au niveau international il existe divers organismes de certification qui auditent les entreprises chargées de la construction des avions et vérifient s'ils sont conformes aux normes. Ce n'est qu'avec l'accord de ces organismes de certification qu'un système aura le droit d'être embarqué et qu'un avion pourra être commercialisé (certificat de navigabilité). Au niveau des É.U. l'organisme certificateur est le FAA (« Federal Aviation Administration »). Dans le contexte du logiciel la certification FAA est faite en utilisant le guide RTCA DO-178B (RTCA/DO-178B, 1992).

2.4 Condition de défaillance et niveaux de criticité

Les organisations qui sont dédiées à la construction des logiciels critiques dans le secteur aéronautique ont besoin d'être conformes aux exigences du guide DO-178B. Ce guide définit des niveaux de criticité du logiciel associé aux équipements aéronautiques. Chaque niveau de criticité est associé à une série d'exigences qui doivent être remplies pour que l'équipement soit considéré comme conforme au guide.

Le guide DO-178B impose aux développeurs du logiciel considéré comme critique certains requis pour que le logiciel soit considéré sécuritaire ou « certifié ». Ces requis sont exprimés

en termes des niveaux de criticité. Le guide présente 5 niveaux de criticité (A à E). Si le niveau de criticité augmente, la rigueur de la vérification augmente aussi, c'est-à-dire plus de contraintes de vérification sont demandées sur :

- les exigences de bas niveau;
- l'architecture du logiciel;
- le degré du test de couverture;
- le niveau d'indépendance de la vérification;
- la robustesse des tests;
- l'utilisation des activités redondantes de vérification.

Les niveaux de criticité sont déterminés dans un contexte des systèmes où le logiciel critique serait embarqué. Des exemples de ces systèmes pour le logiciel critique sont : le « GPS différentiel » et les « systèmes de senseurs de proximité ». Dans ces deux composants une certification de niveau « A » serait proposée car elle devrait assurer l'élimination des conditions de défaillance entraînant le compromis de la sécurité du vol ou atterrissage et qui pourraient amener à l'écrasement de l'avion (voir tableau 2.1).

Le guide DO-178B définit pour chaque niveau de criticité une série d'exigences qui doivent être remplies pour que l'équipement soit considéré comme conforme. Ces exigences sont traduites en procédures internes de développement plus aisément applicables, à savoir :
Plus le niveau de criticité est proche du niveau A, plus le nombre d'objectifs à atteindre est élevé (voir tableau 2.2).

Tableau 2.1

Concepts de défaillance et niveaux de criticité, traduit de (Rierson, 1999)

Conditions de défaillance	**Description**	**Niveau du logiciel**
Catastrophique	Conditions de défaillance entraînant le compromis de la sécurité du vol ou atterrissage - Crash de l'avion.	A
Hasardeuses/ Sévère-Majeur	Conditions de défaillance entraînant une réduction de la capacité de l'aéronef ou la réduction de la capacité de l'équipage à gérer l'opération de l'aéronef. Il s'agit d'un problème majeur entraînant des dégâts sérieux, voire la mort de quelques occupants.	B
Majeur	Conditions de défaillance entraînant une réduction de la capacité de l'aéronef ou la réduction de la capacité de l'équipage à gérer l'opération de l'aéronef. Problème sérieux entraînant un dysfonctionnement des équipements vitaux de l'appareil.	C
Mineur	Conditions de défaillance sans impliquer une réduction de la sécurité de l'aéronef. Les membres de l'équipage auront la capacité de prendre des actions correctives. Problème pouvant perturber la sécurité du vol.	D
Sans effet	Conditions de défaillance sans entraîner la capacité opérationnelle de l'aéronef ou augmenter la charge de travail de l'équipage. Problème sans effet sur la sécurité du vol.	E

Dans les niveaux supérieurs, comme par exemple le niveau A, les objectifs doivent être vérifiés par des groupes indépendants. Aucun « code mort » n'est permis dans le logiciel et toutes les exigences incluant le code et l'information des tests sont susceptibles d'être auditées et doivent être traçables.

2.5 La traçabilité dans le contexte du guide DO-178B

Le guide DO-178B spécifie que toutes les exigences du logiciel critique (incluant les exigences des systèmes allouées au logiciel ou STRATS), le code et l'information de tests devront être audités et tracés. Chaque morceau du code doit satisfaire les objectifs

particuliers du niveau correspondant pour le logiciel. Le guide exige une traçabilité entre les divers artefacts associés en concordance avec le niveau de criticité du logiciel.

Tableau 2.2

Objectifs de défaillance. Adapté et traduit de (Rierson, 1998)

Niveau du logiciel	Description	Objectifs des processus
A	• La couverture de code source au niveau "condition"/"décision" est requise. • En plus des contraintes du niveau B	66
B	• La couverture de code au niveau "condition" est requise. • Les activités de développement et de vérification doivent être confiées à des équipes indépendantes. • En plus des contraintes du niveau C	65
C	• Des règles de développement doivent être fixées au préalable (sur les phases de spécification, conception et codage) • Les contraintes de traçabilité et de vérification s'appliquent également aux phases de conception et de codage du logiciel. • Les exigences de bas niveau (ou exigences de conception) doivent être formellement vérifiées. • La couverture structurelle, ou couverture de code, doit être vérifiée et analysée : toutes les instructions du code doivent avoir été exécutées et testées, tous les écarts doivent être justifiés. Ces contraintes amènent souvent à devoir passer des tests unitaires. • En plus des contraintes du niveau D	58
D	• Le logiciel doit être documenté ; la liste de documents à fournir est fixée par la norme (voir plus bas). • Préalablement au développement, des plans doivent être établis pour fixer les méthodes de développement, de vérification, de gestion de configuration, d'assurance qualité. • Il faut assurer et vérifier la traçabilité entre les spécifications du système, les spécifications de haut niveau du logiciel, et les vérifications. • Tout ce qui est spécifié doit être formellement vérifié : la couverture fonctionnelle doit être assurée. Les documents doivent aussi être formellement vérifiés. • Le logiciel doit être géré en configuration, par exemple toutes les évolutions du code source doivent être justifiées • Un service qualité indépendant doit assurer le respect de la norme en inspectant les sorties du cycle de vie du logiciel.	28
E	• Le développement logiciel n'est soumis à aucune contrainte particulière.	0

La traçabilité dans le guide (RTCA/DO-178B, 1992) est décrite de la façon suivante :

- Les exigences de haut niveau sont développées;
- Les exigences de bas niveau se conforment aux exigences de haut niveau;
- Le code source est conforme aux exigences de bas niveau;
- Le code source est traçable aux exigences de bas niveau;
- Les tests de couverture de haut et de bas niveau ont été complétés.

Chilenski (Chilenski, 2002) a résumé en quelques mots que la traçabilité facilite d'une façon correcte et objective l'évaluation et la mesure de la qualité des exigences du système et du logiciel en termes par exemple de complétude, consistance, testabilité, etc.

Selon la DO-178B la traçabilité montre à quel point les exigences ont été implémentées. La couverture des exigences se fait entre les exigences et les tests de ces exigences. Ceci se fait avec la méthode de couverture des exigences (entre les exigences et les tests) et la couverture structurale (entre les tests et le code source). Le guide établit une définition claire des processus de test (du logiciel et du système). Elle demande de déterminer les tests nécessaires pour établir que :

- toutes les exigences du logiciel et système doivent être testées et vérifiées,
- tout le code source soit implémenté,
- tout le code objet soit implémenté et vérifié (pour le niveau A de criticité).

Il est aussi exigé par la norme d'établir une évaluation objective de la pertinence des processus de test dans le système, à savoir :

- toutes les exigences (système et logiciel) doivent être implémentées,
- tout le code source doit implémenter les exigences du logiciel,
- tout le code objet doit implémenter le code source (juste pour le logiciel catalogué niveau A),
- l'implémentation des exigences (système et logiciel) a été vérifiée.

Ce processus de vérification et d'analyse de la traçabilité permet d'identifier le code mort qui n'est pas testable, le code désactivé, le code qui a été produit accidentellement sans avoir des fonctions spécifiques dans le logiciel, ainsi que les fonctions qui n'ont pas été vérifiées.
Dorsey (Dorsey, 2002) explique qu'il existe des concepts de « Forward traceability » et « Backward traceability » dans le guide DO-178B. Le « Forward traceability » indiquerait que le logiciel critique réunirait toutes les exigences définies pour le logiciel et le « Backward traceability » indiquerait que le logiciel réunirait juste les exigences définies pour le logiciel.

2.6 Les activités de vérification dans le guide DO-178B

Afin de remplir les objectifs au niveau de la vérification du logiciel, le guide DO-178B spécifie les méthodes de vérification suivantes:

- Revues : ces méthodes fournissent une évaluation qualitative de l'exactitude des processus de développement et de vérification
- Analyses : ces méthodes fournissent une évidence répétable de l'exactitude des processus de développement et de vérification
- Les Tests :
  - Fournissent l'évidence que le logiciel satisfait les exigences de haut et bas niveau.
  - Fournissent la confiance que les erreurs qui pourraient conduire à une condition de défaillance ont été éliminées.
  - La DO-178B mentionne que la préparation des tests peut être aussi effective que l'exécution de test.

En général, le guide DO-178B déclare que les activités de vérification procurent une évaluation des exigences, ainsi que de l'architecture et du code source du logiciel au niveau de l'exactitude, complétude et vérifiabilité. Il est noté que le développement des cas de tests procure une évaluation supplémentaire des exigences au niveau de la cohérence interne et complétude; d'ailleurs l'exécution des procédures des tests procure une preuve de conformité avec les exigences.

2.7 La vérification et validation indépendant dans le guide DO-178B

Le concept d'indépendance dans le guide DO-178B traite de la séparation des responsabilités afin d'assurer l'accomplissement de l'évaluation du logiciel d'une façon objective. Il faut préciser que :

(1) pour des activités de processus de vérification de logiciel l'indépendance est réalisée quand l'activité de vérification est exercée par une personne autre que le réalisateur du composant à être vérifié. Un outil peut être utilisé pour réaliser cette activité.

(2) pour le processus de garantie de qualité de logiciel, l'indépendance inclut également l'autorité pour assurer des actions correctives.

Il faut préciser que la plus grande difficulté pour l'autorité de la vérification et validation indépendant (V&VI) est d'évaluer le niveau approprié de V&VI requis. En plus des contraintes d'échéance et de budget, le gestionnaire V&VI doit tenir compte du niveau de complexité et de criticité du programme, de la méthodologie de développement, de l'historique et du niveau d'expérience de l'équipe de développement.

2.8 La certification du logiciel critique selon le guide DO-178B

Le guide DO-178B présente les principaux acteurs et les artefacts qui sont impliqués dans le processus de certification du logiciel critique.

La guide définit pour chaque niveau une série d'exigences qui doivent être remplies pour que l'équipement soit considéré comme conforme. Ces exigences sont traduites en procédures internes de développement plus aisément applicables.

Par exemple : pour le test logiciel, la DO-178B définit des exigences mais n'impose pas de méthodes pour les remplir. Ceci est laissé à la discrétion de chaque organisation chargée du développement. Si une exigence est remplie en test unitaire, test d'intégration ou sur banc

d'essai, cela importe peu du moment qu'elle est validée et, bien entendu, que la méthode est fiable.

Pour éviter toute dérive par rapport à la DO-178B, les organismes de certifications valident l'ensemble des procédures et méthodes. Les organisations qui développement des systèmes critiques pour l'aéronautique ont l'obligation de ce conformer à la DO-178B sous peine de ne pouvoir faire embarquer leurs systèmes et commercialiser leurs avions.

Comme nous avons mentionné au niveau des États Unis, la « Federal Aviation Administration » (FAA) emploie le guide DO-178B pour évaluer la fiabilité des systèmes aéronautiques qui serons embarqués dans les avions qui vont survoler leur territoire.

2.9 Livrables pour la certification du logiciel critique selon le guide DO-178B

Le chapitre 11 du guide DO-178B identifie un ensemble de livrables que tout cycle de vie du logiciel critique devrait produire pour la planification, la direction, l'explication, la définition, l'enregistrement ou pour juste fournir l'évidence que certains activités clés ont été accomplies dans le projet. Ces livrables ou données (« Data » dans le langage du guide) facilitent la définition du processus du logiciel, la certification et la post-certification du produit logiciel.

Les livrables ou données (« Data ») produits pendant le cycle de vie peuvent être classifiés dans une ou deux catégories associées à la gestion de la configuration du logiciel: « Control Category 1 » (CC1) et « Control Category 2» (CC2). Ces catégories sont associées à chaque donnée ou livrable qui serait évalué lors de la certification. Les spécifications pour chaque donnée ou livrable est décrite dans l'annexe A du guide DO-178B.
Voici la liste des livrables :

- Plan for Software Aspects of Certification (PSAC);
- Software Development Plan (SDP);
- Software Verification Plan (SVP);

- Software Configuration Management Plan (SCMP);
- Software Quality Assurance Plan (SQAP);
- Software Requirements Standard (SRS);
- Software Design Standard (SDS);
- Software Code Standard (SCS);
- Software Requirements Data (SRD);
- Software Design Description (SDD);
- Source Code;
- Executable Object Code;
- Software Verification Cases and Procedures (SVCP);
- Software Verification Results (SVR);
- Software Life Cycle Environment Configuration Index (SECI);
- Software Configuration Index (SCI);
- Problem Reports;
- Software Configuration Management Records (SCMR);
- Software Quality Assurance Records (SQAR);
- Software Conformity Review (SCR);
- Software Accomplishment Summary (SAS).

2.10 Certification des livrables du cycle de vie du logiciel

Les livrables ou données (« Data ») produits pendant le cycle de vie du logiciel sont évalués par une autorité désignée afin d'avoir l'approbation finale de la FAA. Ces autorités désignées sont appelés « DER » (« Designated Engineering Representative ») et sont des représentants indépendants de la FAA. Tous les projets du logiciel critique reliés au guide DO-178B devront avoir l'approbation du DER pour leur certification (FAA, 2003).

Parmi les plans plus importants pour la certification FAA demandés par le DER sont (Hilderman V. et Baghi T., 2007) :

- Plan for Software Aspects of Certification (PSAC);
- Software Quality Assurance Plan (SQAP);
- Software Configuration Management Plan (SCMP);
- Software Development Plan (SDP);
- Software Verification Plan (SVP).

Le PSAC doit être soumis pour l'approbation du DER. Ce document est composé des références à d'autres plans (les données ou « Data » du cycle de vie du logiciel), ainsi qu'à des artefacts du projet. Le SQAP incorpore une description du comment les audits des processus du logiciel et des artefacts du logiciel devront être exécutés. Ce document doit être élaboré par une personne indépendante aux équipes du développement. Le SCMP spécifie comment le contrôle de la configuration serait établi dans le projet. Ce document inclut les conventions de noms, les stratégies de révision et contrôle ainsi que comment les rapports de problèmes sont créés et administrés. Le SDP inclut les échéanciers pour les jalons, les livrables, les ressources. Ce document explique le cycle de vie utilisé dans le projet (compilateurs, outils pour gérer les exigences, etc. Le SVP est semblable au SDP sauf qu'il est orienté vers les processus de vérification, à savoir : revues, analyses, tests, outils pour la vérification, etc.

2.11 Sommaire

Ce chapitre a présenté le guide DO-178B. Nous avons exploré les concepts clés pour la certification en concordance avec le guide, ainsi que les livrables exigés pour avoir la certification. Également nous avons expliqué les niveaux de criticité du système attribué au logiciel. Le concept et l'importance de la traçabilité aussi ont été abordés dans le sens qu'il est essentiel pour démontrer la cohérence entre les exigences de criticité et les différents composants du logiciel critique. Nous avons mentionné les différentes techniques demandées par le guide DO-178B pour implémenter la vérification dans le logiciel critique.

L'analyse présentée dans ce chapitre a permis d'identifier les exigences du guide DO-178B pour la certification du logiciel critique. Cette étude encadrera l'organisation de notre proposition du référentiel des processus de V&V.

CHAPITRE 3

LA FORMALISATION DES PROCESSUS DU LOGICIEL

3.1 Introduction

Ce chapitre introduit des normes, des modèles des processus, d'amélioration des processus ainsi que des concepts associés à la modélisation et formalisation des processus du logiciel. Une révision des outils pour la formalisation des processus logiciels est également présentée.

3.2 Revue de la littérature pour le chapitre

Cette section présente les activités à accomplir afin d'identifier notre problématique et objectifs de recherche. Notre démarche dans ce chapitre inclut la révision des documents qui nous aideront à identifier notre problématique et objectifs de recherche. Ces documents portent sur le cycle de vie du logiciel, leurs processus, la formalisation de ces processus, des outils pour la formalisation. La figure 3.1 résume notre approche à ce volet de la revue de la littérature.

Les activités de revue de la littérature pour la formalisation des processus du logiciel débutent avec une étude du cycle de vie du logiciel et des processus. Nous explorons les principaux processus du logiciel utilisés à date dans l'industrie. Un survol de la norme IEEE 12207 (IEEE/EIA 12207, 1998), des processus unifiés (« PU » ou « UP ») (Jacobson et al, 1999) ainsi que du RUP (Kroll et Krutchten, 2003) sont présentés dans ce chapitre.

Nous présenterons la version « Open Source » des processus unifiés appelé « OpenUP » (Balduino, 2006). Une exploration des divers cadres pour l'amélioration des processus est aussi présentée : Le CMMI (SEI, 2006), le FAA-iCMM, ISO 90003 (ISO/CEI Std 90003, 2004).

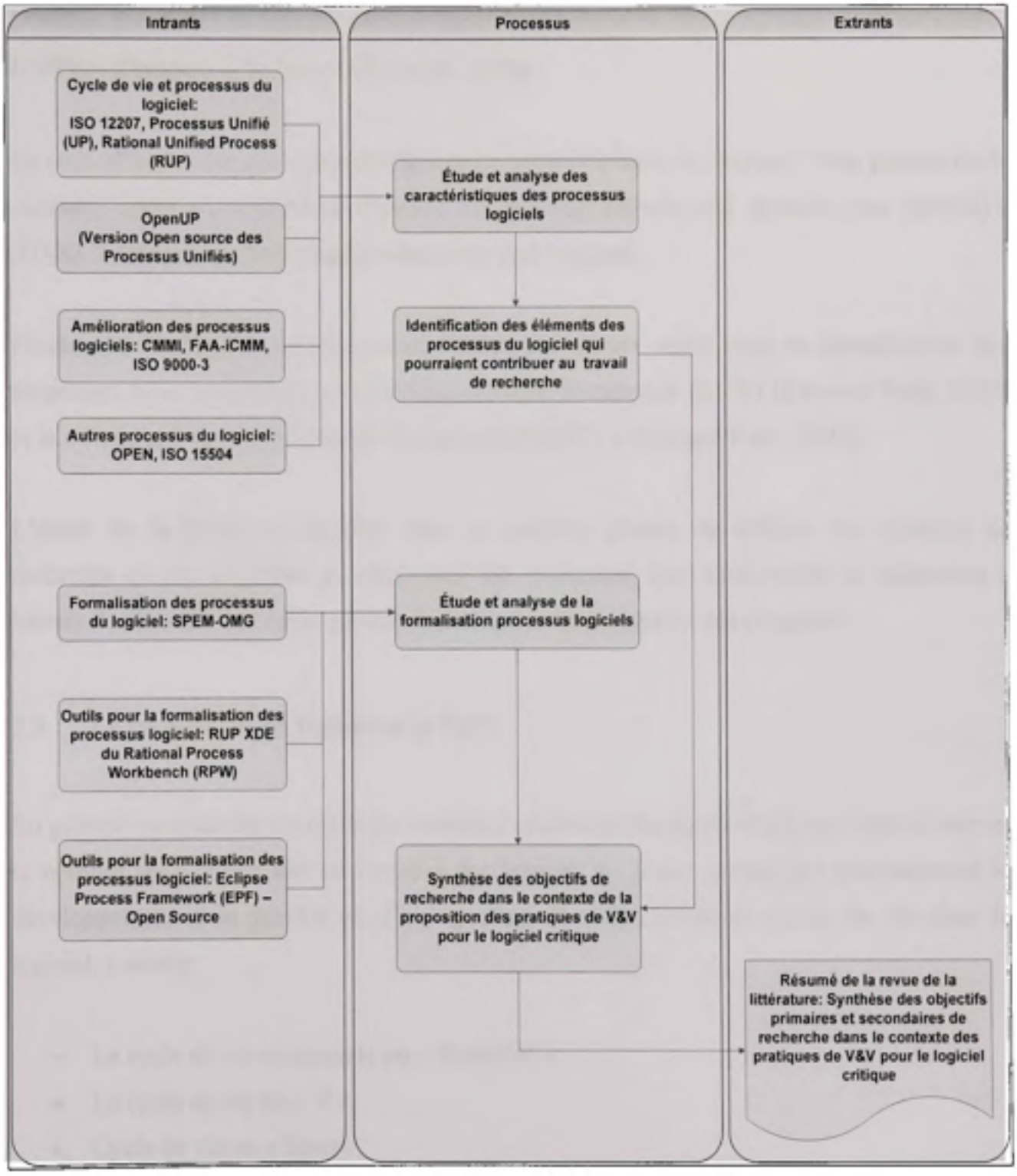

Figure 3.1 ***Revue de la littérature : Formalisation des processus logiciel.***

D'autres processus du logiciel utilisés dans l'industrie et la recherche sont aussi présentés : L'OPEN (Henderson-Sellers et Firesmith, 2002).

La normalisation des processus est également présentée dans ce chapitre. Nous présentons le standard OMG du « Software Process Engineering Méta-model Specification (SPEM) » (OMG, 2008) pour la formalisation des processus logiciels..

Finalement le chapitre conclure avec l'étude des divers outils pour la formalisation des processus. Nous présentons le « Rational Process Workbench (RPW) (Rational Staff, 2004) et le « Eclipse Process Framework Composer (ECPFC) » (Haumer Peter, 2007).

L'étude de la littérature explorée dans ce chapitre permet de raffiner nos objectifs de recherche en vue d'inclure un composant des processus, leur amélioration et adaptation à travers l'utilisation des outils pour la formalisation et la création des plugiciels.

3.3 Le cycle de vie du logiciel et la V&V

En général un cycle de vie est défini comme l'ensemble séquentiel de phases, dont le nom et le nombre sont déterminés en fonction des besoins du projet, permettant généralement le développement d'un service ou d'un produit. Il existe plusieurs cycles de vie dans le logiciel, à savoir:

- Le cycle de vie en cascade ou « Waterfall »;
- Le cycle de vie en « V »;
- Cycle de vie en « Spiral.

3.3.1 Le cycle de vie en cascade ou « Waterfall »

Il s'agit d'un cycle de vie séquentiel, linéaire, «en cascade». Ce cycle a été introduit par Royce (Royce, 1970) dans les années 70. Le cycle comporte 8 phases séquentielles et il est

initié à partir des exigences des systèmes jusqu'à l'opération et la maintenance. Ce cycle est basé sur les productions livrables dans chaque phase. Les activités de vérification et validation se font dans chaque étape du cycle.

Voici quelques difficultés liées au cycle de vie en cascade :

- Il faut bien préciser les exigences dès le début (ou au moins la plupart);
- Il n'accepte pas le changement en cours de route «il faut tout bien faire dès le début»;
- Les spécifications doivent précéder le design, qui elles-mêmes doivent être finalisées avant de passer au codage et tests unitaires;
- Il exige d'accorder une attention très importante aux livrables.

3.3.2 Le cycle de vie en « V »

Le cycle de vie en «V» est une alternative au cycle en cascade. Ce modèle comporte un ensemble des tâches qui doivent être faites en parallèle:

- Horizontalement : préparation de la vérification. Ex. : dès que la spécification fonctionnelle est faite :
    - plan de tests de qualification;
    - plan d'évaluation des performances;
    - documentation utilisateur.
- Verticalement : développement des modules. Ex. : dès que la conception globale est validée :
    - conception détaillée des modules;
    - programmation et tests unitaires.

Voici quelques difficultés liées au cycle de vie en en « V » :

- Difficile de prendre en compte des changements importants dans les spécifications dans une phase avancée du projet;
- Durée parfois trop longue pour produits compétitifs;

- Peu ou pas de possibilité de maquettage et/ou de prototypage;
- Trop de choses reportées à l'étape de programmation (par ex. l'interface utilisateur);
- Pas assez de résultats intermédiaires pour valider la version finale du produit.

3.3.3 Cycle de vie en « Spiral » (Boehm, 1988):

Le modèle en spirale se caractérise par:

- La détermination des objectifs du cycle, des alternatives pour les atteindre et des contraintes, à partir des résultats précédents, ou de l'analyse préliminaire des besoins.
- Le développement en série de prototypes pour identifier les risques en commençant par le plus grand risque.
- L'utilisation du modèle de cycle en « V » ou en cascade pour implémenter chaque cycle.
- La revue des résultats : si un cycle concernant un risque a été achevé avec succès :
    - Évaluer le résultat du cycle et planifier le cycle suivant;
    - Si un risque n'a pu être résolu, terminer le projet immédiatement.

3.4 Les processus du logiciel

Raman (Raman, 1999) explique que le succès du développeur du logiciel est focalisé sur les processus de construction. En général un processus est une séquence d'actions pour accomplir une tâche (IEEE Std 610.12, 1990). Le processus du logiciel pourrait être décrit comme l'ensemble des activités, méthodes, pratiques et transformations qui sont utilisées pour développer et faire la maintenance du logiciel ainsi que des produits associés (par exemple : plan du projet, documents de design, le code, les cas de tests, etc.).

Un des défis importants dans le développement des produits logiciels consiste à identifier les processus appropriés et à les formaliser dans un langage de modélisation convenable. Il existe quelques langages pour la modélisation du logiciel, par exemple : IDEF0 Integrated Definition Method 0 (NIST, 1993), entre autres. Ces langages présentent une caractéristique

commune : ils ne sont pas construits pour supporter le design du logiciel en termes des processus d'affaires.

L'UML (Unified Modeling Language) (Larman, 2005), en revanche, fournit une option pour la modélisation orientée objet au niveau de l'architecture du logiciel en général. L'UML est utilisé largement dans le développement du logiciel, mais il faut noter que l'UML est concerné par la construction du logiciel et il n'est pas concerné par la définition des processus d'affaires.

3.4.1 La norme ISO 12207

Le norme ISO 12207 (Software Life Cycle Processes) (IEEE/EIA 12207, 1998) définit une taxonomie pour les processus de cycle de vie de logiciel. La norme correspond à un consensus international au niveau des activités qui constituent un projet logiciel. La norme est conçu pour être flexible et adaptable aux besoins du produit à être développé. ISO 12207 se compose d'une série de processus, activités et tâches adaptables à tout projet de logiciel. La norme ne spécifie pas comment exécuter ces processus, laissant cette tâche à l'organisation responsable.

La norme est caractérisée par trois types des processus : primaire, de support et d'organisation. Les processus primaires composent le cycle de vie de logiciel et sont responsables de la création du produit logiciel. Les processus primaires sont : acquisition, approvisionnement, développement, entretien et opération. Les processus de support visent à aider les autres processus, principalement adressant la qualité et le succès du projet. Ces processus sont: la documentation, la gestion de la configuration, la résolution de problème, l'audit, l'assurance qualité, la vérification, la validation et la revue. Les processus organisationnels servent à assurer et à améliorer les processus dans organisation. Ces processus sont : la gestion, l'infrastructure, l'amélioration et la formation.
Les paragraphes suivants présentent les processus les plus pertinents pour cette thèse :

Processus de vérification : Ce processus fournit les évaluations liées à la vérification d'un produit ou d'un service d'une activité donnée. La vérification détermine si les exigences pour un système sont complètes, correctes et que les sorties d'une activité remplissent les conditions des activités précédentes. Le processus de vérification couvre le processus de vérification, des exigences, le design, le code, l'intégration, et la documentation.

Processus de validation : La validation détermine si le système final remplit les conditions prévues pour son utilisation. L'ampleur de la validation dépend de la criticité du projet. La validation ne remplace pas d'autres évaluations, mais les complémente.
La vérification ou la validation (V&V) peut être conduite par un client, par le fournisseur, ou par un groupe indépendant. La V&V qui est exécutée par une organisation indépendante du fournisseur ou du développeur, est appelée processus de vérification et validation indépendantes (V&VI).

3.4.2 Le « Rational Unified Processs ou RUP»

Le « Rational Unified Process » (RUP) (Kruchten, 2000) est un cadre pour la gestion des processus de développement logiciel. RUP vise à améliorer la productivité du développement. RUP recommande des bonnes pratiques du développement logiciel durant tout le cycle de vie d'un projet. Le RUP est composé de phases (Inception, Elaboration, Construction, Transition) et de disciplines (Business Modeling, Requirements, Analysis & Design, Implementation, Test, Deployment, Configuration & Change Mangement, Project Management, Environment) (Kruchten, 2000).

RUP propose d'adopter une approche itérative pour le cycle de vie du logiciel. Les phases comportent des objectifs qui doivent être atteints avant de passer à la phase suivante. Une discipline peut être impliquée dans plusieurs phases. Les activités sont créées à partir des disciplines et elles vont donner lieu à la création des artefacts spécifiques à la fois par activité, par discipline et par phase. Une discipline dans le RUP regroupe différents évènements et activités tels que : Rôles, flux, activités, et produits (« workproducts »). Les informations dans le RUP sont organisées de la façon suivante: introduction, concepts, flux,

activités, artefacts, guidelines. Chaque flux conduit à un lien qui détaille les activités à réaliser, ainsi que les rôles, intrants et extrants (Kroll et Krutchten, 2003).

La figure 3.2 présente la modélisation UML de la structure du RUP selon Bencomo (Bencomo, 2005). Cette figure reflète la structure de classes et ses relations pour formaliser les concepts de phases, rôles, artefacts, disciplines, entre autres qui est inclus dans le RUP.

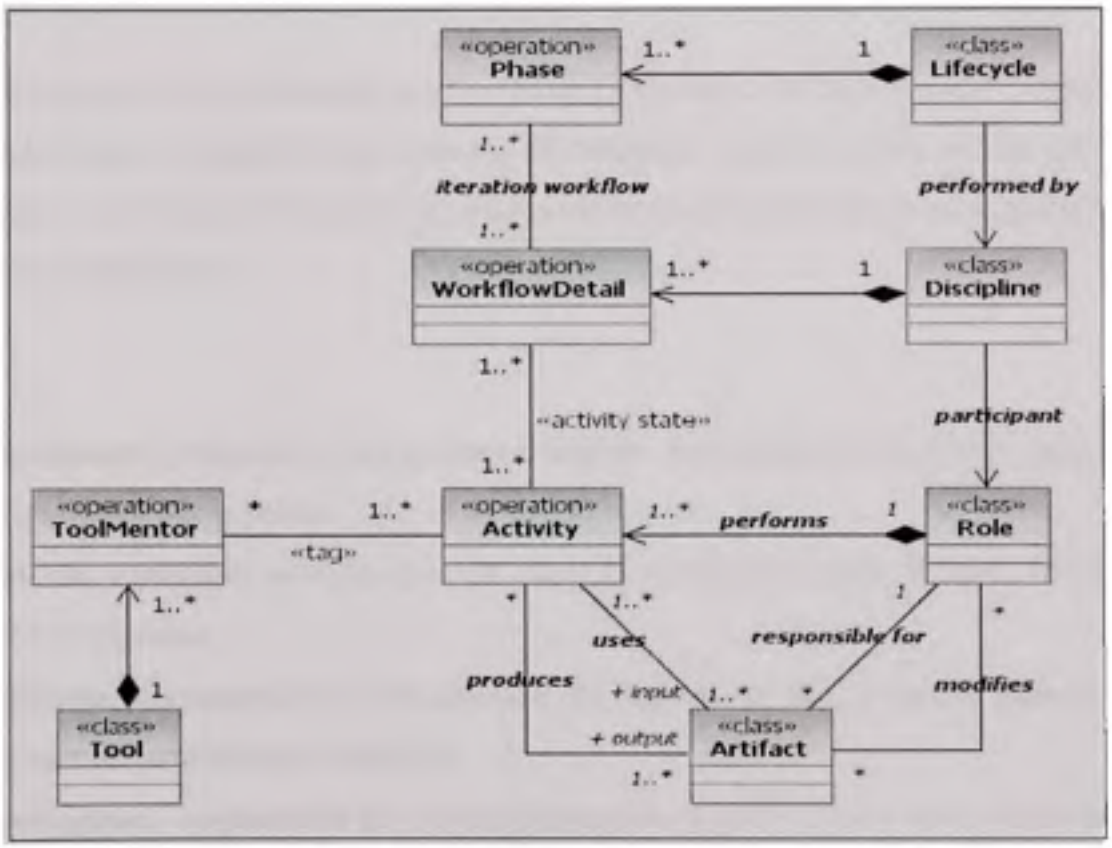

Figure 3.2 ***Représentation partielle du méta-modèle de RUP.***

3.4.3 Le Processus Unifié ou « Unified Process » et l'OpenUP

Balduino (Balduino, 2006) explique que OpenUP est considéré comme un processus de développement logiciel agile et léger, qui favorise les meilleures pratiques de développement du logiciel, à savoir: le développement itératif, la collaboration d'équipes, l'intégration

continue, les tests, les livraisons fréquentes des produits logiciels, l'adaptation aux changements, entre autres.

OpenUP est un processus de développement logiciel « Open Source » qui a l'intention d'être souple et minimaliste mais à la fois complet pour le développement logiciel. Il a la capacité d'être extensible à différent environnements. Cette capacité d'être extensible est accordée par l'outil « Eclipse Process Framwork Composer (EPFC) » présenté dans une section postérieure dans ce chapitre.

OpenUP comporte deux dimensions : le « Method Content » et le « Process Content ». Le « Method Content » comporte les éléments de méthodes appelés : rôles, tâches, artefacts et guidelines). Le « Process Content » est où les éléments des méthodes seront appliqués. Ces « Contents » permettent

Rôles :

- **Intervenants,** représentent des groupes d'intérêts. Les exigences de ces groupes doivent être comblées par le projet.
- **Analyste,** représente les exigences du client et l'utilisateur final. Propose des priorités pour les exigences.
- **Architecte,** responsable de l'architecture du logiciel, et des décisions techniques qui imposent des contraintes au logiciel.
- **Développeur,** responsable du développement du système. Il est aussi responsable du design et de combler les exigences de l'architecture. Il fait des tests unitaires, et l'intégration des composants logiciels.
- **Testeur,** responsable des activités principales de test.
- **Gestionnaire des projets,** conduit la planification du projet en collaboration avec les intervenants et l'équipe. Il coordonne l'interaction les communications et les objectifs du projet.
- **Autre rôle,** représente quelqu'un qui exécute des tâches au niveau général dans le projet.

Disciplines

La partie « method content » d'OpenUP est focalisée sur les disciplines suivantes : Exigences, architecture, développement, test, gestion des projets, et configuration & gestion des changements.

Tâches

Une tâche est une unité de travail qu'un rôle est capable à exécuter. OpenUP consiste en 18 tâches que les rôles peuvent exécuter comme exécuteurs primaires ou autres exécuteurs.

Artefacts

Un artefact est quelque chose qui est produit, modifié ou utilisé par une tâche. Les rôles sont responsables de créer et modifier les artefacts. Il existe 17 artefacts qui sont considérés essentiels en OpenUP.

Processus

Il s'agit d'un « method content » réutilisable qui est créé à part. Cette méthode prend les éléments d'un processus et les associe entre eux pour les adapter à un projet en particulier. Les méthodes sont organisées dans des composants réutilisables appelés « Capability Patterns ». Ces Patterns sont faits par l'organisation des tâches en activités. Ces activités facilitent la planification des objectifs et l'exécution d'un travail. Ces travaux sont exécutés par les développeurs et peuvent être traçables. Un item de travail pourrait être un cas d'utilisation, un scénario, une exigence de changement. Un contexte peut être spécifié pour un item de travail et associé à un développeur. De cette façon il serait facile de savoir si l'item de travail a été évolué ou pas.

OpenUP est basé sur les phases des processus unifiés, à savoir : Inception, Elaboration, Constrcution, et Transition. Ensemble ces phases vont développer les objectifs pour chaque phase.

3.5 Amélioration des processus du logiciel

Tribble (Tribble, 2002) a constaté que la demande de plus haute qualité pour les produits du logiciel a forcé l'industrie du logiciel à considérer une approche plus disciplinée et plus structurée du développement pour ce type de logiciel. Le nombre croissant d'organisations de développement du logiciel qui deviennent reconnues quant à leur maturité du processus reflète ce besoin. On constate le besoin de nombre d'organisations d'être évaluées selon divers modèles, comme les suivants : CMMI (SEI, 2006), ISO 15504 (ISO / IEC 15504, 2004), ISO 9001 (ISO/IEC Std 9001, 2005), etc. Cependant, aucun de ces modèles d'évaluation des processus du développement du logiciel n'est en mesure de garantir la qualité d'un produit du logiciel. Les processus de maturité sont acceptés généralement comme une condition nécessaire, mais pas suffisante pour une organisation dédiée à la production de logiciels de qualité.

3.5.1 Le modèle CMMI

Le CMMI (Modèle intégré d'évolution des capacités logiciel) ou simplement CMMI (SEI, 2006), a été développé par le « Software Enginnering Institute » (SEI) affilié à l'université Carnegie Mellon. Le modèle CMMI pour le logiciel est une collection de bonnes pratiques de développement logiciel, qui expriment les meilleures façons de travailler pour produire du logiciel de qualité, avec une productivité accrue et dans le respect des budgets et délais. Ces pratiques y sont regroupées en secteurs clés ou PA (« Process Area ») qui sont, à leur tour, rattachés à un niveau de maturité. Les cinq niveaux de maturité définis dans le modèle constituent une suite de phases dans l'amélioration des processus de développement logiciel. Chacune de ces phases sert de fondation à la mise en œuvre des pratiques du niveau de maturité supérieur et propose donc des priorités dans l'amélioration des processus logiciel.

Le PA de Vérification (VER) s'assure que les produits de travail (« work products ») déterminés remplissent les exigences spécifiques du processus de développement. Dans le

VER les méthodes de vérifications appropriées sont sélectionnées. La VER est un processus incrémental qui est initié avec la vérification des composants et finit avec la vérification totale des produits. Le PA de Validation (VAL) de façon incrémentale évalue les produits contre les besoins du client. VAL inclut la validation de produits, composants, produits de travail, et processus. Ces éléments valides pourraient avoir besoin d'une ré-vérification ou une ré-validation. Les problèmes rencontrés dans ce PA sont résolus dans les PA « Requirements Developpement (RD) » ou « Technical Solution (TS)». Voici le résumé des « Specific Goals » et « Specific Practices » des PA de VER et VAL (Basque, 2004) :

Vérification

SG 1 Se préparer à la vérification:

- SP 1.1 Choisir les produits de travail de sortie à vérifier;
- SP 1.2 Établir l'environnement de vérification;
- SP 1.3 Établir les procédures et les critères de vérification;

SG 2 Réaliser les revues de pairs :

- SP 2.1 Se préparer pour les revues de pairs;
- SP 2.2 Mener des revues de pairs;
- SP 2.3 Analyser les données des revues de pairs;

SG 3 Vérifier les produits de sortie sélectionnés :

- SP 3.1 Réaliser la vérification;
- SP 3.2 Analyser les résultats de vérification et identifier les actions correctives.

Validation

SG 1 Se préparer pour la validation :

- SP 1.1 Choisir les produits de travail en vue de la validation;
- SP 1.2 Établir l'environnement de validation;
- SP 1.3 Établir les procédures et les critères pour la validation.

SG 2 Valider le produit ou les composants de produit :

- SP 2.1 Réaliser la validation;
- SP 2.2 Analyser les résultats de validation.

3.5.2 Le modèle FAA-iCMM

Le modèle des capacités FAA-iCMM aligné sur le CMM a été développé par la « Federal Aviation Administration » FAA des É.U. La particularité de ce modèle est qu'il traite les aspects de la sûreté et les efforts d'amélioration des processus pour une organisation (Ibrahim, 1997).

Le modèle est composé de « Practices Areas » (PA) qui sont les équivalents des secteurs clés du CMM et aux « Process Areas » du CMMI. Le modèle définit des nouveaux secteurs clés et les place dans une perspective d'amélioration et d'intégration. Les PA sont appelés des pratiques de base (BP). Chaque PA est défini par un objectif, une description, et les relations avec d'autres PA. Les indications par rapport à la sûreté sont incluses dans les BP et PA. De la même façon que CMM ou CMMI, le FAA-iCMM ne parle pas du « comment » implanter les pratiques mais de « Ce qui doit être fait» pour garder la sûreté dans les projets.

3.5.3 ISO 90003

La norme ISO 90003 (ISO/CEI Std 90003, 2004) présente les lignes directrices pour l'application de la norme ISO 9001:2000 au développement, à la mise à disposition, à l'installation et à la maintenance du logiciel. Cette norme est utilisée dans l'industrie de logiciel comme une référence pour la qualité du produit et des processus du logiciel. La norme explique les exigences de la qualité du logiciel pour devenir certifié sous ISO 9001.

ISO 90003 explique que certaines activités sont liées aux phases du développement et d'autres sont applicables tout ou long du processus. La norme ISO 90003 fait souvent référence à la ISO 9000 et ISO 9001.

La norme ISO 90003 présente des recommandations pour l'implantation de la vérification et validation dans le développement logiciel. ISO 90003 explique également le besoin de l'identification de la traçabilité et la gestion de configurations.

3.6 Autres processus du logiciel

3.6.1 OPEN

OPEN (Object-oriented Process, Environment, and Notation) (Henderson-Sellers et Firesmith, 2002) est un méta-modèle des processus pour le développement des systèmes. Il inclut un cycle de vie et des stratégies pour la réutilisation. Les processus dans OPEN peuvent être instanciés pour les besoins d'un projet en particulier. OPEN est appelé « OPEN Process Framework (OPF) ». L'OPF est composé des éléments suivants:

- **Un méta-modèle,** qui définit les types des méthodes à utiliser ainsi que ses relations.
- **Une base de données (« repository ») des méthodes (« methods components »)** , pour la description des processus.
- **Des guidelines** pour la construction et l'utilisation des méthodes afin de produire des processus particuliers adaptés à chaque situation.

Le méta-modèle fournit une terminologie et une sémantique pour l'ensemble des méthodes (Henderson-Sellers et Firesmith, 2002). Voici une brève description de ces méthodes:

- **Work Products**, comportent les documents, les diagrammes, etc.
- **Work Units,** modèlent une fonctionnalité, et sont de trois types:
    - **Activités**, comportent une collection cohésive de tâches;
    - **Tâches**, comportent une opération exécutée par un « Producer »;
    - **Techniques,** modèlent la forme des tâches à être exécutées.
- **Producers,** comportent des méthodes composantes qui modèlent les « Work Products ».
- **Languages,** modèle le langage à utiliser dans les Work products.
- **Endeavours,** modèle les composants au niveau d'entreprise. Ex : projets, etc.
- **Stages,** définissent des périodes dans le développement, comme par exemple, les phases, builds, milestones, etc.
- **Work performances,** modèlent les « Work Units ».

La base de données (« repository ») de méthodes, est un ensemble des méthodes disponibles pour être personnalisées pour les besoins spécifiques d'un projet. Les types des méthodes ont été identifiés dans le méta-modèle dans le paragraphe précédent. La façon d'utiliser les méthodes seront décrites dans la partie « guidelines » de l'OPF. Les types de méthodes vont donner lieu à des sous-types qui serviront pour l'instanciation concrète dans les projets, par exemple : rôles, activités, Work Products, etc.

Par exemple pour les exigences nous pouvons trouver **les méthodes** suivantes (D. Zowghi, D.G.Firesmith et B. Henderson-Sellers, 2005) : Stakeholder profiling (tâche qui représente les intervenants), Customer analysis (tâche qui représente l'étude, l'analyse et modélisation des besoins de l'entreprise), Business Visioning (tâche qui représente la vision du client de l'organisation et sujet de la amélioration), etc. D'ailleurs pour faciliter l'implantation de ces tâches il existe différentes **techniques**, comme par exemple : Brainstorming, abstraction, documentation templates, interviews, inspections, etc. Les «Work products» typiques sont : Software Requirements Specification, Use Case diagrams, Class Diagram, etc. L**es langages** servent à la production de Work products, comme par exemple: les langages naturels, les langages de modélisation (UML), les langages de spécification : langage Z, etc. Parmi les « producers » nous trouvons dans la base des données: Business Architect, Domain expert, Process Engineer, entre autres.

OPEN inclut également diverses **guidelines** pour l'application des méthodes, comme par exemple pour la construction, pour la personnalisation (« tailoring »), l'extension du méta-model (la modification), entre autres.

3.6.2 La norme ISO 15504

La norme ISO 15504 (ISO/IEC 15504, 2004) est un référentiel qui a été démarré en 1993 et s'est appuyée sur des référentiels existants tels que les normes ISO 9001, ISO 12207 et des concepts hérités des modèles de maturité tels Bootstrap, Trillium and the CMM (Capability Maturity Model). Actuellement ISO 15504 est une norme développé par le « Working Group

on Process Assessment (WG10) of the International Committee on Software and Systems Engineering Standards ISO/IEC JTC 1/SC7.

ISO 15504 propose un modèle pour l'évaluation des processus dans deux contextes :

- **L'amélioration des processus :** pour comprendre l'état des processus de l'organisation. Les résultats sont utilisés pour élaborer les plans d'améliorations;
- **La détermination de la capacité :** pour déterminer la pertinence pour les processus d'une organisation.

La version du 2004 du standard comporte 5 parties, à savoir :

- Part 1: Concepts and vocabulary;
- Part 2: Performing an assessment;
- Part 3: Guidance on performing an assessment;
- Part 4: Guidance on use for process improvement and process capability determination;
- Part 5: An exemplar Process Assessment Model.

Le processus d'évaluation est présent dans deux étapes : La dimension des processus (« Process dimension ») et la dimension des capacités (« Capability dimension »). Chaque processus est évalué et se voit attribué à un niveau de maturité. Il existe six niveaux de maturité. Le tableau 3.1 présente quelques principes d'évaluation.

Tableau 3.1

Les niveaux de maturité selon ISO 15504 (ISO/IEC 15504, 2004)

Niveau	Description	Mesures de capacité du processus possibles (Attributs)
Niveau 0 : Processus incomplet	L'objectif du processus n'est pas atteint. Il y a peu ou pas de produits ou de résultats du processus qui sont facilement reconnaissables.	Aucun
Niveau 1 : Exécuté	Un niveau de réussite a été atteint. Le processus comporte des produits qui démontrent l'atteinte de l'objectif fixé.	Mesure dans laquelle le processus atteint les résultats prévus en transformant les produits «intrants» en vue de la génération de produits «extrants».
Niveau 2 : Processus géré	Le processus fournit des produits conformément aux méthodes précisées et il fait l'objet d'une planification et d'un suivi. Le processus génère désormais des produits conformes aux exigences établies en matière de qualité, délais, ressources.	Gestion des performances : Mesure dans laquelle la performance du processus est gérée afin de générer des produits qui devront correspondre aux objectifs fixés. Mesure de gestion de produit : Mesure dans laquelle la performance est gérée en vue de la génération de produits correctement documentés, contrôlés et vérifiés.
Niveau 3 : Processus établi	Le processus est exécuté et géré à l'aide d'un processus défini.	Mesures de définition de processus : Mesure dans laquelle la performance du processus se sert d'une définition de processus se fondant sur un processus standard pour atteindre les résultats prévus. Mesure de ressource de processus : Mesure dans laquelle le processus fait appel aux ressources correspondantes qui sont correctement attribuées afin de l'implantation du processus défini.
Niveau 4 : Processus prévisible	Le processus défini est exécuté de façon constante dans le respect des limites de contrôle établies, en vue de l'atteinte des objectifs fixés. On recueille et analyse des données précises sur les performances. La principale distinction par rapport au niveau de processus établi a trait au fait que le processus défini est exécuté de façon constante à l'intérieur des limites définies, afin de l'atteinte des résultats prévus.	Mesures : Mesure dans laquelle on se sert des objectifs et des mesures des produits et du processus pour s'assurer que la performance du processus favorise l'atteinte des objectifs fixés. Mesure de contrôle de processus : Mesure dans laquelle le processus est contrôlé par la collecte, l'analyse et l'utilisation de mesures des produits et du processus en vue de la correction, au besoin, des performances du processus dans le but d'atteindre les buts fixés pour les produits et le processus.
Niveau 5 : Optimisation.	Les performances du processus sont optimisées afin de répondre aux besoins actuels et futurs de l'organisation et le processus peut être répété afin d'atteindre les objectifs fixés. Il établit l'efficacité quantitative du processus ainsi que les objectifs à cet égard (performances), d'après les objectifs de l'organisation.	Mesures de changement de processus : Mesure dans laquelle on contrôle les changements à la définition, la gestion et les performances du processus afin d'atteindre les objectifs de l'organisation. Mesure d'amélioration continue : Mesure dans laquelle on détermine et apporte des changements au processus afin de garantir une amélioration constante quant à l'atteinte des buts de l'organisation

3.7 La modélisation des processus du logiciel

La modélisation des processus facilite la compréhension et la communication en fournissant une opinion partagée des processus. La modélisation supporte la gestion et l'amélioration des processus. La modélisation peut être aussi employée pour assigner des tâches, faire le suivi du progrès, et pour identifier des problèmes dans les processus de développement logiciel.

3.7.1 La formalisation des processus logiciels : SPEM - OMG

Le SPEM (Software Process Engineering Metamodel) (OMG, 2008) est une spécification qui a été introduite afin de combler les besoins pour la définition dans deux contextes: d'un côté les processus d'affaires et de l'autre le design du logiciel. SPEM est un méta-modèle défini par l'OMG (Object Management Group) et est utilisé pour la description des procédés. SPEM s'appuie sur le formalisme UML et s'intègre dans l'approche MDA (Model Driven Architecture) préconisée par l'OMG. C'est donc une instance du méta-méta-modèle MOF (Meta Object Facility) et à la fois un profil UML.

Le SPEM est représenté dans une architecture de 4 couches, à savoir :

- **Le niveau M3**, est composé d'une entité qui est le langage unique de définition des méta-modèles appelé le MOF (Meta-Object Facility). Le MOF est un modèle de méta-modèle (ou méta méta-modèle) qui se situe à la base de toute définition de méta-modèle (i.e. il est auto-descriptif).

- **Le niveau M2**, utilisé pour la définition des méta-modèles qui contient un ensemble des concepts, des relations entre ces concepts, et de contraintes. Ces méta-modèles vont permettre de définir des modèles (niveau M1). Typiquement, le méta-modèle UML qui est décrit dans le standard UML appartient au niveau M2; il définit la structure interne des modèles UML.

- **Le niveau M1**, est composé de modèles d'information, utilisé pour la description des instances type niveau M0.

- **Le niveau M0**, le niveau des données réelles, est composé des informations que l'on souhaite modéliser.

SPEM est concordant avec le principe qu'un processus de développement de logiciel est une collaboration entre des entités actives et des abstractions: les rôles, les activités, les produits. Les rôles ont une relation entre eux et collaborent en échangeant des produits et en provoquant des activités.

Les notations proposées par SPEM comprennent des diagrammes basés sur UML (diagramme des classes, diagramme de composants, diagramme de cas d'utilisation, diagramme de séquence, diagramme d'états/transitions et diagramme d'activités) et des symboles appropriés pour représenter certaines conditions/artefacts.

3.8 Outils pour la formalisation/modélisation des processus

3.8.1 Rational Process Workbench (RPW)

Le RPW est une extension du RUP. Le RPW est un outil qui sert à instancier le RUP pour les exigences précises des projets. Cet outil facilite l'implémentation des pratiques de RUP dans le cadre des particularités d'une organisation. Le RPW comporte: RUP Builder, RUP Organizer, et RUP Modeler. Il inclut aussi le Process Engineering Process (PEP).

Le RUP Modeler inclut l'outil RUP XDE qui permet la création de nouveaux rôles, artefacts, activités, relations dans le RUP. Cette capacité est conférée par le « Process Engineering Process (PEP) » qui est le moteur du RUP. Le RUP Organizer est un autre outil qui facilite la tâche de réutiliser les rôles et activités du RUP dans le nouveau modèle qu'on est en train de bâtir. Après la modélisation des processus avec RUP XDE, il existe la possibilité de

développer et publier des plugiciels dans un format HTML, par le biais de l'utilisation de l'outil RUP Builder. Au moment de la rédaction de cette thèse les outils sont encore disponibles pour le téléchargement sur le site Web d'IBM (Rational Staff, 2004).

3.8.2 Eclipse Process Framework Composer (EPFC)

EPFC est un outil développé dans le cadre du projet « Eclipse Process Framework (EPF) » (Haumer Peter, 2007). Il est une application construite sur le « framework Eclipse ». EPFC est un projet du logiciel libre qui vise à fournir à la communauté un cadre pour la formalisation des processus. Cet outil permet de standardiser et d'administrer les processus de développement logiciel. EPFC est basé sur l'infrastructure du méta-modèle SPEM 2.0 (OMG, 2008) qui est un standard pour la modélisation et normalisation des processus.

EPFC permet de faire une extension des processus cohérents avec OpenUP facilitant la personnalisation des processus à travers des plugiciels (« Process Authoring and Tailoring »). EPFC permet une installation facile, configuration et publication des processus (plugiciel) par le biais de HTML dans les environnements de travail. Il permet aussi une intégration avec un système de gestion de configurations.

EPFC est aussi applicable aux différents types des projets, comme par exemple pour différents environnements de développement. Les plugiciels sont appelés aussi des méthodes ou des « Libraries » ou bibliothèques. Dans l'environnent de travail, l'outil définit une « Library » par défaut, nommée « base_concepts », proposant des conseils, des termes et vocabulaire pouvant être réutilisés dans les méthodes qu'on pourrait définir.

EPFC distingue deux aspects dans la description d'une bibliothèque ou « Library » :

- L'aspect définition des concepts manipulés, que l'on appelle « Method Content » (ou contenu de la méthode);
- L'aspect enchaînement de ces concepts que l'on appelle « Process » (ou processus).

Une fois que la méthode est définie, l'outil permet de créer une configuration, c'est-à-dire un filtre sur le processus, qui permet de générer tout ou partie de la méthode en un site Web interactif. L'outil propose aussi un espace permettant de décrire toutes les entités manipulées dans une méthode.

« Method Content »

Le contenu de la méthode (« Method Content ») est décomposé en trois sous-sections, à savoir:

- **« Content packages »** (paquetages de contenu);
- **« Standard categories »** (catégories standard);
- **« Custom categories »** (catégories personnalisées).

« Content packages »

L'outil offre la possibilité d'organiser les éléments de la méthode dans les packages. Cette organisation établit une hiérarchie l'information afin de mieux la récupérer dans l'environnement.

Chaque package est décomposé en quatre sections qui pourront être remplies ou non:

- **Roles (rôles)** : Comporte tous les intervenants du processus, qui appartiennent au package.
- **Tasks (tâches)** : Comporte toutes les actions possibles du package.
- **Work products (produits de travail)** : Comporte tout ce qui peut être utilisé ou produit et qui est relatif au package. Il faut distinguer les « artifacts » (artefacts), les « outcomes » (produits) et les « deliverables » (livrables).
- **Guidance (conseils)** : Comporte la liste de conseils relatifs au package.

« Standard categories »

La section « Standard categories » permet d'ordonner le contenu de la méthode sous un autre angle, c- à-d. celui de la méthode, du processus. Les informations sont regroupées en cinq catégories standard, à savoir :

- **Disciplines** : il s'agit d'un ensemble de tâches qui peuvent être regroupées afin de réaliser une phase dans le processus.
- **Domains :** il s'agit du pendant des disciplines pour les « Work products » et non plus les tâches. Un domaine est une hiérarchie logique de « Work products » regroupés entre eux sur des critères temporels (timing), de ressources ou de relations entre eux.
- **Work Product Kinds** : il s'agit d'une autre façon de regrouper les « work products » entre eux. Un « work product » peut se retrouver dans plusieurs « Work Product Kinds ».
- **Role Sets** : sert à regrouper des rôles ayant un certain nombre d'associations entre eux.
- **Tools** : peut fournir une description générale des outils et de leurs caractéristiques.

« Custom categories »

Cette section permet une catégorisation personnalisée des informations. L'outil donne la possibilité d'organiser les informations selon les besoins.

3.8.2.1 « Processes »

« Delivery processes »

Un Delivery Processes est le processus qui couvre l'ensemble du cycle de vie de développement du début à la fin. Il peut être utilisé comme un « template » ou gabarit pour la planification et l'exécution du projet. Il fournit un modèle de cycle de vie avec ses phases, ses itérations et ses activités.

Ce processus est construit à partir des groupes d'activités définies dans la section « Capability patterns », séparés généralement par des jalons (« milestones »), c'est-à-dire des évènements déterminant des fins ou des débuts de phase.

« Capability patterns »

Un « Capability pattern » est un groupe d'activités qui partagent une même problématique. Ce groupe décrit un ensemble de phases de développement sans décrire les relations entre ces phases, un « Delivery processes » s'occupe d'établir ce lien.

3.9 Sommaire

La revue de la littérature introduite dans ce chapitre a fait un survol des concepts et principes reliés aux cycles de vie et processus du logiciel. Nous avons présenté l'organisation des concepts à l'intérieur de RUP et la norme IEEE 12207.

Nous avons également fait un survol de l'amélioration des processus du logiciel et nous avons présenté le CMMI, le FAA-iCMM, l'ISO 90003 dans un contexte liée à la V&V et la construction du logiciel critique. Les principes de la norme ISO15504 et le cadre OPEN ont été également présentés.

Nous avons exploré la représentation formalisée des processus appelé SPEM de l'OMG, ainsi que les outils pour la formalisation des processus basé sur SPEM. Nous avons introduit le RPW et particulièrement le RUP XDE. Nous avons fait la présentation de la version « Open Source » des processus unifiés appelé « OpenUP » ainsi que l'outil « Eclipse Process Framewrok Composer - EPFC » qui permet la création des produits de travail (artefacts, livrables, résultats) type plugiciel dans les projets.

CHAPITRE 4

L'ANALYSE DES MESURES

4.1 Introduction

Ce chapitre introduit le concept des mesures pour le contrôle et le suivi des projets, l'importance de la création d'un entrepôt générique de mesures pour l'évaluation des activités de V&V dans les projets. Ce chapitre présente des concepts des mesures pour les activités de V&V. Le chapitre fait référence à la norme ISO 15939 (ISO/IEC Std 15939, 2007) ainsi que certaines technologies pour la création de l'entrepôt.

4.2 Revue de la littérature pour le chapitre

La figure 4.1 illustre les activités de cette revue de la littérature dans le cadre de notre travail de recherche pour l'entrepôt des mesures. D'un côté nous allons focaliser notre travail dans le contexte des mesures, comme par exemple l'idée des mesures de base, des mesures dérivées, des indicateurs. Ces concepts vont guider la construction de l'entrepôt des mesures proposé dans cette thèse. D'ailleurs nous allons aussi aborder le sujet de l'implantation des mesures dans les organisations avec les recommandations de la norme ISO 15939 ainsi que le sujet de la technologie OLAP et du « Datamart » afin de comprendre les enjeux du développement d'un entrepôt des mesures, ainsi que de l'implémentation des caractéristiques clefs pour l'analyse des mesures. Voici la description de la séquence d'activités mentionnés dans la figure 4.1.

Les activités de la revue de la littérature pour ce chapitre débutent avec l'étude sommaire des concepts et l'importance des mesures dans les projets. Ensuite nous présentons la norme ISO 15939 (ISO/IEC Std 15939, 2007) et le PSM (Practical Software Measurement) (McGarry, 2001) pour l'identification, la définition, la sélection et l'application des mesures dans les projets. Nous discuterons des différentes étapes d'un processus des mesures.

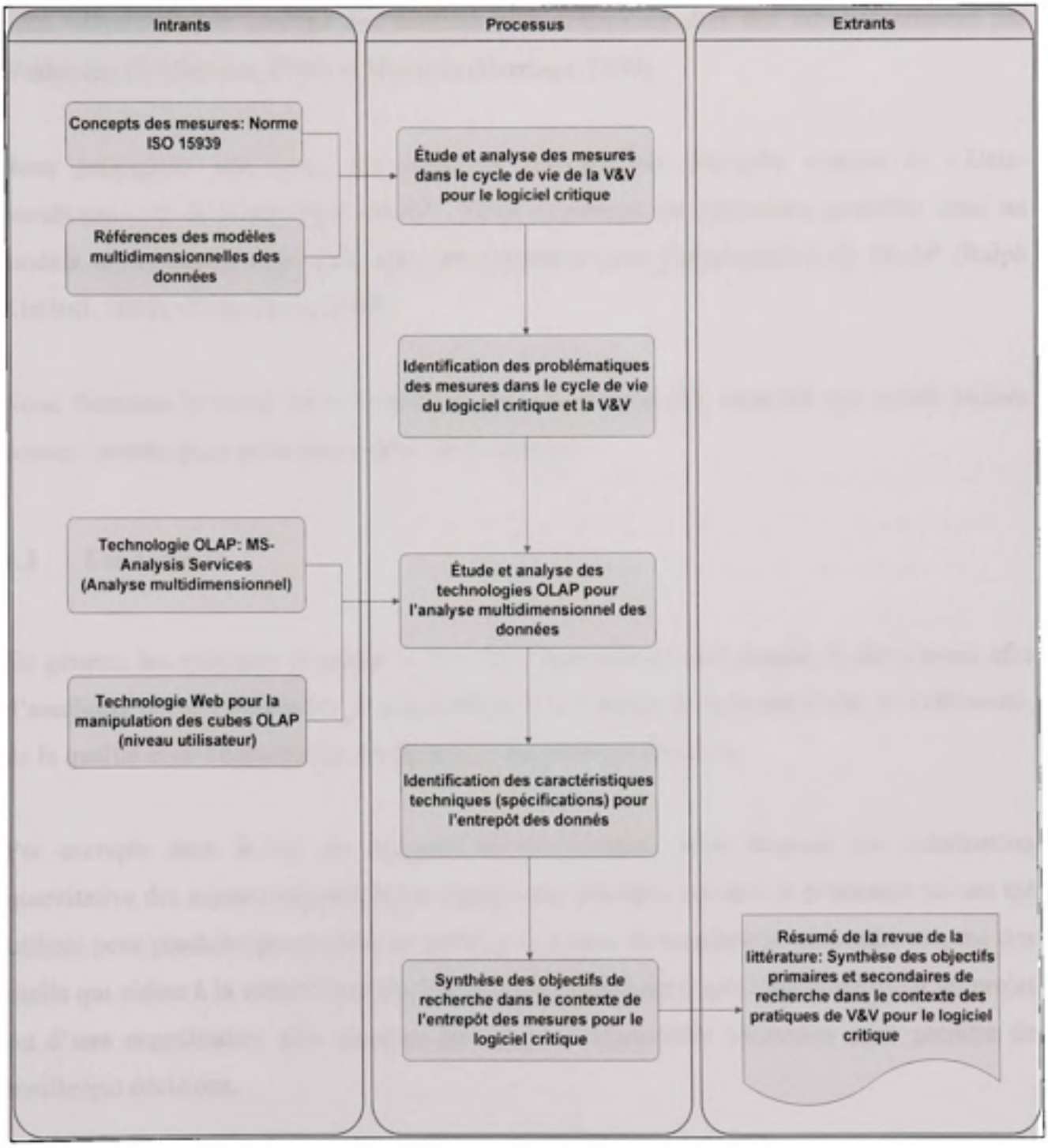

Figure 4.1 ***Revue de la littérature pour l'entrepôt des mesures de V&V.***

Nous introduisons le concept des données multidimensionnelles des mesures proposé par Walkerden (Walkerden, 1995) et Harrison (Harrison, 2000).

Nous présentons une revue des concepts associés aux entrepôts comme le « Data-warehouse », et la technologie OLAP. Nous explorons les opérations possibles dans un modèle multidimensionnel ainsi que des approches pour l'implantation de OLAP (Ralph Kimball, 2002), (Olap Train, 2000).

Nous finissons la revue de la littérature avec un résumé des objectifs qui seront utilisés comme intrants pour notre proposition de recherche.

4.3 Les mesures

En général les systèmes de mesures dans les organisations sont conçus et développés afin d'améliorer la compréhension, la planification et le contrôle de la productivité, de l'efficacité, de la qualité et de l'opportunité des projets et des produits logiciels.

Par exemple dans le cas des mesures de performance, elles donnent une information quantitative des aspects importants par rapport aux produits, services et processus qui ont été utilisés pour produire ces produits et services. Ce type de mesures est considéré comme des outils qui aident à la compréhension, facilitent la gestion ainsi que l'amélioration d'un projet ou d'une organisation. Ces mesures fourniront l'information nécessaire pour prendre de meilleures décisions.

En résumé nous mesurons dans le contexte du logiciel pour (Zubrow David, 2007) :

- Comprendre d'une meilleure façon les activités du logiciel, afin de pouvoir prédire certains comportements.
- Comprendre ce qui existe ou ce qui a été fait dans le passé.
- Améliorer les méthodes de développement de logiciels.

- Faciliter la compréhension de l'état d'un projet et faciliter la planification des stratégies de développement.
- Améliorer la qualité des processus de construction du logiciel, par le biais d'une évaluation et comparaison des pratiques de développement.
- Améliorer des produits logiciels en termes des méthodes, ressources, etc.
- Prédire des efforts requis, des coûts, des échéanciers, des activités, entre autres.

4.4 Programme des mesures : La norme ISO 15939

La ISO 15939 - *Software Engineering - Software Measurement Process*, (ISO/IEC Std 15939, 2007) est une norme internationale qui définit des processus de mesures pour le développement de logiciel et le génie des systèmes. Cette norme est utilisée lors de l'implantation des programmes des mesures dans les projets et dans les organisations. Elle décrit des processus et des tâches pour l'implantation des mesures. Ces processus visent à adresser les exigences d'information de l'organisation ou du projet.

La norme explique comment identifier, définir, sélectionner, appliquer et améliorer les mesures du logiciel dans un projet du développement logiciel. La norme ne décrit pas un ensemble des mesures à implanter dans un projet mais elle établit un guide pour définir des mesures cohérentes avec les besoins d'information. L'ensemble des processus recommandés par la norme ISO 15939 (ISO/IEC Std 15939, 2007) pour l'implantation des programmes de mesures sont les suivants (voir aussi figure 4.2):

- **« Établir et supporter un engagement des mesures»,** dans ce processus la portée du projet des mesures est déterminée, les engagements des parties sont établis et les ressources pour accomplir les activités sont assignées.
- **«Planifier les processus des mesures »,** dans ce processus les besoins d'information sont identifiés, les procédures de mesures sont définis ainsi que les technologies pour supporter le projet.
- **« Exécuter les processus des mesures »**, dans ce processus les données sont collectées, les principaux résultats de mesures sont produits et communiqués.

- « **Évaluer l'utilisation de la mesure** », les résultats des mesures sont évalués, et les possibles améliorations aux processus des mesures sont identifiées.

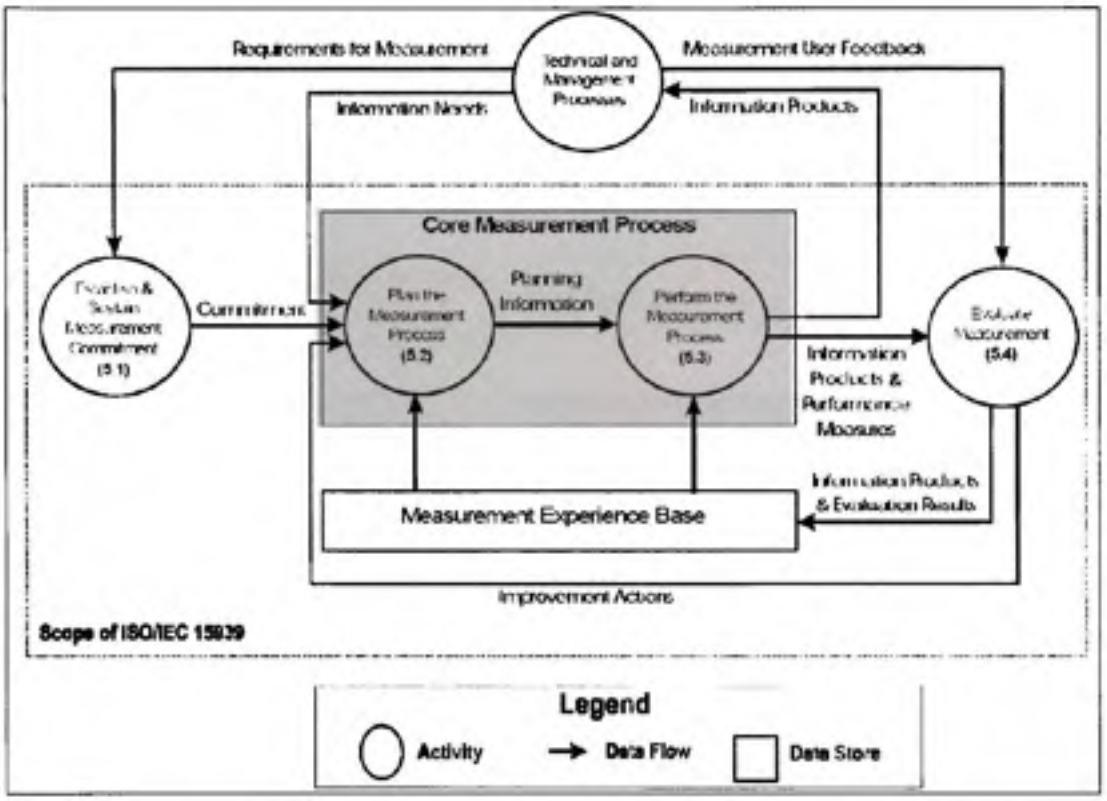

Figure 4.2 ***Modèle des processus de mesures du logiciel selon ISO/IEC 15939.***

Les mesures candidates pour un projet sont guidées par les besoins d'information. La norme ISO 15939 propose un modèle pour la sélection des mesures afin de satisfaire les besoins d'information de l'organisation ou du projet. Le modèle est illustré et décrit dans la figure 4.3. Le modèle explique que les mesures candidates sont déterminées à partir des besoins d'information et par une analyse pour se transformer en mesures de base, mesures dérivées et indicateurs.

4.5 Données multidimensionnelles des mesures

- Walkerden (Walkerden, 1995) et Harrison (Harrison, 2000) proposent le concept de données multidimensionnelles des mesures. Ces mesures sont décrites comme des données qui ont la capacité de stoker différent type de données des mesures. On parle de données génériques de mesures qui sont récupérées par divers types d'indicateurs selon le besoin du projet.

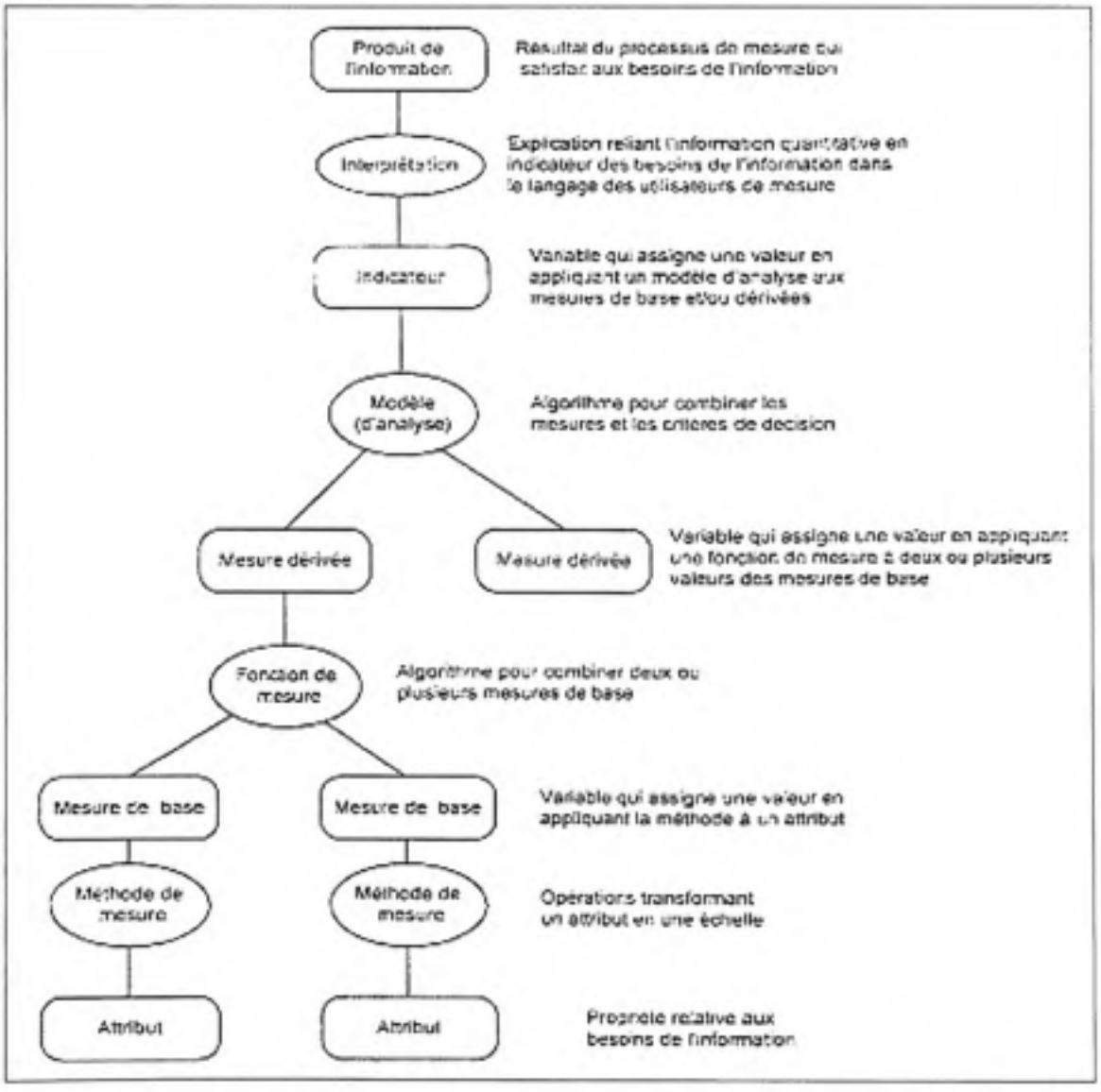

Figure 4.3 ***Modèle de mesures selon ISO 15939, traduit par (Sellami, 2005).***

4.6 La technologie OLAP et les entrepôts des données

OLAP (On-Line Analytical Processing) ou l'analyse multidimensionnelle des données a pour but de supporter la prise de décisions basée sur un ensemble des données organisées en forme d'agrégats. Les bases de données des systèmes conventionnels ont été développées pour supporter principalement les applications OLTP (On-Line Transaction Processing). Ces OLTP sont normalement liées à des opérations dans les organisations. D'ailleurs les systèmes de gestion des base des données (DBMS) n'offrent pas de fonctions puissantes pour la synthèse des données, l'analyse et la consolidation (Codd, Codd S.B. et Salley C.T., 1993).

OLAP offre comme solution des données centralisées et multidimensionnelles afin de faciliter la prise de décisions. Une base de données multidimensionnelle comporte une grande quantité de données-agrégats et pré-calculées à partir des DBMS. Dans la pratique, un pré-calcul des données-agrégats est nécessaire car les requêtes OLAP sont complexes et peuvent demander des heures ou des jours d'exécution sur les données directes contenues dans les OLTP.

OLAP et « data-warehouse » sont considérés comme des synonymes. Thomsen (Thomsen E, 1997) considère ces termes complémentaires dans le sens où le data-warehouse rend accessible les données aux usagers en assurant la précision et la cohérence, et OLAP met l'emphase sur les exigences analytiques de l'usager.

Il faut noter qu'un « data-warehouse » est typiquement orienté sur stoker l'ensemble des données historiques variant dans le temps, organisées par sujets, consolidés, aidant à la prise de décision au niveau de l'entreprise. Les « datamarts » sont un sous-ensemble de données extraites du « data-warehouse » et ciblées sur un sujet unique (ensemble des données sur un sujet particulier, ex. marketing, production, etc.). Les « datamarts » sont plus performantes étant donné qu'ils ne demandent pas une consultation globale de l'information de toute l'entreprise.

Les données agrégats sont usuellement numériques et mesurables. Alors, ces données sont représentées par des « attributs » ou « attributs mesurables ». Les facteurs de base où les données agrégats sont analysées sont appelés « dimensions ». Par le biais de sélection de dimensions spécifiques où les données-agrégats sont analysées, on pout obtenir une « vue ». Par le biais de changement des dimensions, on peut construire différentes vues. Ces fonctionnalités sont possibles grâce à des requêtes OLAP.

4.6.1 Opérations possibles dans un modèle multidimensionnel

Les opérations possibles dans un modèle multidimensionnel des données sont les suivantes (Ralph Kimball, 2002), (Olap Train, 2000):

- **L'agrégat (« roll-up ») :** cette opération permet de réduire les dimensions du cube, par exemple : faire le calcul du total dépenses par phase de développement, ou bien l'agrégat sur un hiérarque : total de dépenses par type de projet (selon le niveau de criticité) et par année.
- **Sélection et protection (« Slice and dice») :** Cette opération permet la définition d'un sous ensemble du cube, par exemple : les expansés pour le projet = Airbus 330 et mois = 2005
- **Navigations des données détaillés (« Drill down »):** exemple : dépenses projetés – dépenses actuelles), par type de projet
- **Visualisation des opérations (« Pivot ») :** cette opération permet faire la rotation des dimensions du cube.

4.6.2 Approches pour implanter la technologie OLAP

Les approches plus utilisées pour implanter la technologie OLAP son les suivantes :

- **Le « OLAP » relationnelle « Relational OLAP » ou « ROLAP » :** Dans cette approche les solutions ROLAP résident dans un environnement relationnel ou des tables

d'agrégation sont créées dans le même espace que l'entrepôt de données et les « data marts » qui servent de sources pour les cubes ROLAP.

- **Le « OLAP » Multidimensionnelle « Multidimensionnel OLAP » ou « MOLAP ».** Les données sont pré-agrégées dans un environnement séparé et remplacent les tables d'agrégation relationnelles de la solution ROLAP
- **Le « OLAP » Hybride « Hybrid OLAP »** cette approche propose de cumuler les avantages des deux modèles précédents. Les données agrégées sont stockées sous forme multidimensionnelle, alors que les données détaillées sont stockées dans des structures relationnelles.

4.6.3 Représentations multidimensionnelles des données

Il existe un ensemble de techniques pour représenter les données multidimensionnelles dans un entrepôt des données. Parmi les plus importantes nous avons les suivants (Ralph Kimball, 2002), (Olap Train, 2000):

« Le schéma en étoile » (« Star Scheme ») : ce schéma comporte une « table de faits » (« fact tables ») et une table pour chaque dimension (« dimension tables »). Chaque registre dans la « table des faits » contient un pointeur (« foreing key ») pour chaque dimension. Ce pointeur maintient les coordonnées multidimensionnelles et enregistre les mesures numériques de ces coordonnées. Chaque table de dimension comporte des colonnes qui correspondent à des attributs de la dimension.

« Le schéma « Flocon de neige » (« Snowflake ») : Ce schéma fournit un raffinement du schéma en étoile. Il faut mentionner que les schémas en étoile ne supportent pas directement les hiérarchies des attributs. Dans le schéma flocon de neige, la hiérarchie dimensionnelle est explicitement représentée par la normalisation des tables de dimension. Cette configuration apporte des améliorations pour la maintenance des tables de dimension. D'un autre côté, dans le cas des schémas en étoile, une de-normalisation de la structure des tables des dimensions pourrait être plus appropriée pour montrer des dimensions.

4.7 Sommaire

Dans ce chapitre nous avons montré le concept des mesures et les différentes approches pour implanter les mesures dans les projets. Nous avons présenté un survol de la norme ISO 15939 qui explique comment construire des mesures et comment implanter un programme mesures dans les projets ou organisations. Nous avons inclus ce niveau des connaissances pour nous aider à mieux positionner les exigences des mesures dans l'entrepôt des mesures que nous allons implémenter comme un de nos résultats de recherche. Ensuite nous avons présenté les différentes caractéristiques et techniques pour bâtir des cubes ou agrégats des données des mesures avec la technologie OLAP. Ces connaissances vont nous servir à implémenter le côté pratique de notre entrepôt des mesures pour la V&V.

CHAPITRE 5

METHODOLOGIE DE RECHERCHE

5.1 Introduction

Ce chapitre identifie l'approche et la stratégie que nous avons identifiées pour atteindre l'objectif de ce travail de recherche. Nous présentons aussi la méthodologie de recherche. Nous donnons aussi un survol sur les possibles approches pour la vérification et la validation de notre référentiel des processus de V&V et leurs données multidimensionnelles.

5.2 Problématique de recherche

Suite à notre analyse de la littérature nous avons constaté qu'il existe un certain nombre de lacunes dans le cadre de la construction, du contrôle et de la certification du logiciel critique. Nous allons décrire ces lacunes dans les paragraphes suivants.

5.2.1 Problématique d'ordre général

- Manque d'un cadre détaillé pour la construction du logiciel critique en concordance avec des pratiques des processus unifiés (tel que OpenUP) et les exigences de certification du guide DO-178B.
- Manque d'un outil ou artefact logiciel qui permettrait :
    - la définition et l'implémentation des rôles, responsabilités, processus, activités, tâches, etc., dans les projets du logiciel critique selon le guide DO-178B;
    - l'instanciation des pratiques de V&V dans les projets selon les niveaux de criticité exigés par le guide DO-178B.

- Manque d'un outil de type multidimensionnel (Olap Train) pour l'estimation, le contrôle et le suivi des coûts, livrables, processus, délais, ressources, etc. dans les projets du logiciel critique.
- Manque d'une solution pour la construction du logiciel critique qui soit cohérente avec les contraintes des référentiels de bonnes pratiques du génie logiciel tels que le CMMI (SEI, 2006) et ISO 90003 (ISO/CEI Std 90003, 2004) entre autres.

5.2.2 Problématique d'ordre particulier de la recherche

- Manque d'un modèle unifié pour les pratiques de V&V pour le logiciel critique et qui soit cohérent avec le guide DO-178B. Ce modèle permettrait : D'avoir une collection cohérente et intégrée des processus et des artefacts de V&V (procédures, flux, acteurs, rôles, responsabilités, etc.) en concordance avec les exigences de la certification du guide DO-178B et avec les principes des processus unifiés de développement logiciel (« Unified Process »). Cette capacité faciliterait à une entreprise de définir avec précision ses propres activités qui satisfassent les objectifs imposés par le guide DO-178B.

- Manque d'un outil flexible (artefact logiciel type plugiciel) pour la formalisation et l'instanciation des processus de V&V dans les projets du logiciel critique. Cet outil permettrait de :
    - définir facilement la formalisation et l'instanciation des processus, tâches, rôles, des produits de travail (artefacts, deliverables, résultats) etc., de la V&V dans les projets;
    - mieux planifier les ressources, les processus et les artefacts de V&V dans les projets. Cette capacité serait possible à instaurer étant donné la structure technique flexible et adaptable du plugiciel;
    - maintenir pour chaque projet la traçabilité entre les processus, les produits et leurs artefacts;
    - maintenir un historique des projets, facilement accessible;

- maintenir une communication fluide entre les différentes équipes du projet par rapport à ces responsabilités associées. Cette communication se ferait par le biais de la publication du plugiciel, par exemple dans l'intranet du projet.
- faciliter l'institutionnalisation et la standardisation des pratiques de V&V dans l'organisation par le biais d'un outil et des pratiques uniques pour les projets de logiciel critique. La standardisation contribuerait à :
    - une clarté et une meilleure compréhension des projets, des objectifs, des processus, des artefacts etc;
    - une terminologie commune à utiliser par les différents groupes impliqués dans le projet;
    - une uniformité dans la méthodologie de construction du logiciel critique;
    - une utilisation en commun des artefacts pour différents projets;

- Manque d'un entrepôt basé sur des données des mesures génériques qui pourrait fournir la capacité multidimensionnelle pour la définition, collection, évaluation et représentation des données des mesures des projets du logiciel critique. Un tel entrepôt de mesures permettrait :
    - de mieux planifier les coûts des pratiques de V&V à partir des historiques des projets (étant donné qu'il existerait un entrepôt des données de mesures historiques des projets);
    - de mieux faire le suivi des pratiques de V&V dans les phases des projets en termes de ressources, du temps, etc;
    - d'avoir la possibilité d'un suivi et contrôle permanent de l'état de l'avancement des activités dans les projets de V&V;
    - de mieux évaluer, planifier et exécuter la gestion des changements dans les projets. Ceci est possible en vertu des capacités de collecte, contrôle, d'évaluation et d'accès à des données historiques des données des mesures qu'offre l'entrepôt.
        - de mieux quantifier et par conséquence une amélioration continue dans la réalisation des projets grâce à la capacité de l'entrepôt de garder la mémoire des projets en terme d'indicateurs de mesures;

5.3 Objectifs de recherche

5.3.1 Objectifs primaires :

- Proposition d'un référentiel formalisé et unifié des artefacts du logiciel (modèle) pour l'instanciation des processus de V&V du logiciel critique (voir figure 5.1).

- Proposition d'un entrepôt générique des données des mesures de V&V pour la prise de décisions dans les projets du logiciel critique (voir figure 5.1).

5.3.2 Objectifs secondaires:

Les objectifs secondaires consolideront les objectifs primaires, sans l'exécution des objectifs secondaires les objectifs primaires ne seront pas atteintes.

Pour le modèle des processus de V&V

- Proposition de la modélisation des artefacts de V&V de la norme IEEE 1012 (IEEE Std 1012, 2004) avec l'outil RUP XDE (Rational Staff, 2004).
- Proposition de la modélisation des processus de développement du logiciel selon le guide DO-178B.
- Proposition de l'identification des artefacts du logiciel critique en concordance avec le DO-178B et l'UP pour les processus suivants :
    - exigences du logiciel ;
    - design du logiciel ;
    - codification du logiciel ;
    - intégration du logiciel ;
    - vérification du logiciel.
- Proposition de la définition des spécifications pour la construction d'un plugiciel pour l'implantation des artefacts de V&V pour le logiciel critique.

Pour l'entrepôt des mesures

- Proposition de l'identification d'ensemble des mesures pour le cycle de vie des projets de V&V.
- Proposition de la modélisation d'un ensemble des cubes OLAP pour les mesures des projets.

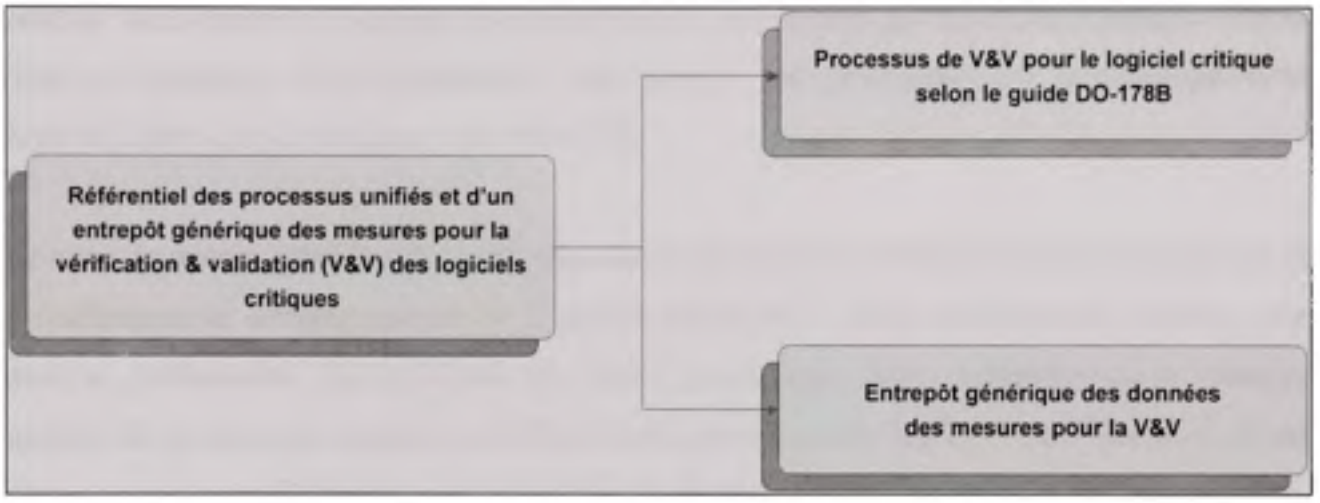

Figure 5.1 ***Composants du référentiel des processus de V&V du logiciel critique.***

5.4 Méthodologie générale de recherche

Dans cette section nous donnons un aperçu des activités que nous allons réaliser pour combler les objectifs de notre proposition de recherche. La méthodologie générale sera divisée en quatre phases. Ces phases présentent les détails de notre démarche en vue de développer et valider chaque contribution de recherche.

Les sous-sections suivantes nous donneront un aperçu par rapport au contenu de chaque phase de notre méthodologie de recherche.

5.4.1 Phase 1 : analyse préliminaire des processus de V&V

La figure 5.2 présente un résumé des activités que nous allons accomplir dans le cadre la modélisation partielle des processus de V&V.

Au début de notre travail de recherche et comme premier livrable nous réaliserons une analyse du chapitre 5 du guide DO-178B afin de positionner les différents concepts décrits dans ce document. Nous présenterons une analyse des processus pour le développement logiciel décrit dans le chapitre 5 du DO-178B.

Comme deuxième livrable dans cette phase nous allons faire un inventaire de recherche sur la modélisation de certains aspects de la norme IEEE 1012. Cette modélisation constitue une analyse préliminaire qui permettra de mieux positionner notre recherche sur le premier objectif de la thèse par rapport au référentiel des processus de V&V. L'outil que nous allons utiliser pour la modélisation sera le RUP XDE (Rational Staff, 2004). Une analyse détaillée, ainsi que les livrables pour cette analyse partielle seront présentés dans le chapitre 6.

Figure 5.2 ***Phase 1 : Méthodologie de l'analyse partielle des processus de V&V.***

5.4.2 Phase 2 : conception et construction du modèle de processus de V&V

Dans cette section nous allons présenter la méthodologie que nous avons conçue pour développer notre première contribution, à savoir : le modèle de processus de V&V pour le logiciel critique. Voir la figure 5.3 pour une illustration de la méthodologie.

Nous allons débuter notre travail en prenant en considération le résumé de la revue de la littérature décrite dans les chapitres 1 à 3. Ensuite nous continuerons avec l'étude et l'analyse du thème de V&V dans la littérature comme par exemple le « SWEBOK » (Abran, 2004) afin de valider la pertinence de nos propositions avec les connaissances tel qu'ils sont présentés dans le corpus de connaissances en génie logiciel « SWEBOK ».

Nous allons continuer avec l'étude et l'analyse des pratiques de qualité du logiciel (Rosenberg, 2001), (Rosenberg, 2002) ainsi que de l'amélioration des processus au sein de la NASA (Rosenberg et Godfrey, 2001). Ceci devrait nous donner une compréhension et une

évaluation des activités liées à la qualité et à l'amélioration des processus au sein de l'institution.

Un autre intrant que nous allons utiliser pour bâtir notre référentiel des processus de V&V c'est le résultat du travail de recherche auquel nous avons participé pour l'identification des artefacts du CMMI (SEI, 2006) pour faciliter l'implantation de la V&V indépendante (Fuhrman, Palza et Do, 2004).

D'autre part, Le modèle proposé devait être composé d'un ensemble d'artefacts du logiciel en concordance avec les exigences du guide DO-178B (RTCA/DO-178B, 1992) et de Processus Unifié (Jacobson et al, 1999), nous analyserons la conception et la construction d'un modèle des pratiques de V&V basé sur « OpenUP ». Nous allons faire l'utilisation de l'outil « Eclipse Process Framework Composer (EPFC) » pour la construction d'un plugiciel des pratiques de V&V. Une analyse détaillée, ainsi que les livrables pour cette analyse seront présentés dans le chapitre 7 et 8.

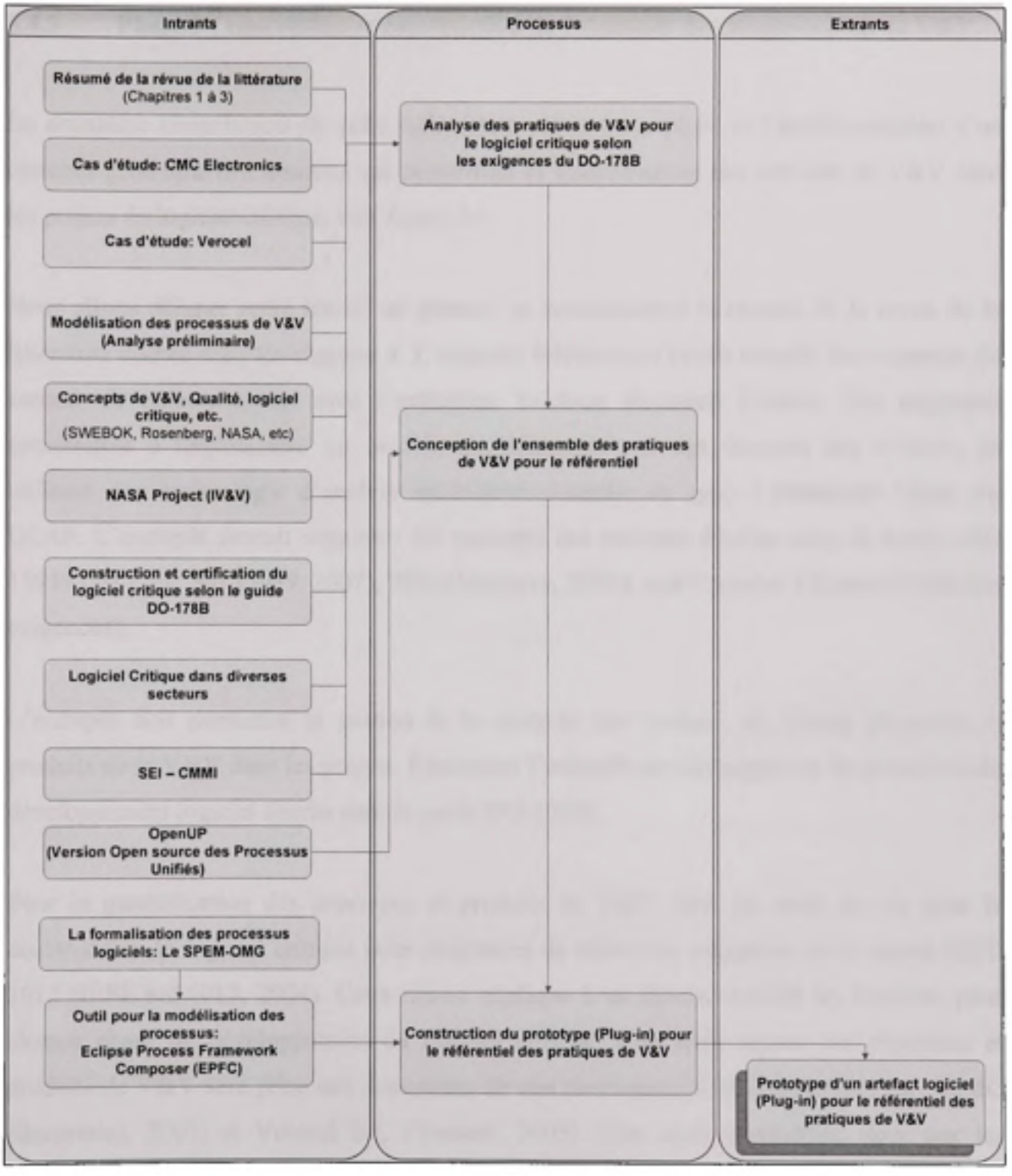

Figure 5.3 ***Phase 2 : Méthodologie pour la conception du référentiel de V&V***

5.4.3 Phase 3 : conception et construction de l'entrepôt des mesures pour la V&V

La deuxième contribution de cette thèse porte sur la conception et l'implémentation d'un entrepôt générique des mesures qui permettrait la quantification des activités de V&V dans les projets du logiciel critique, voir figure 5.4.

Nous allons débuter notre travail en prenant en considération le résumé de la revue de la littérature décrite dans les chapitre 4. L'entrepôt initialement devait remplir les exigences du contrat de recherche fait avec l'entreprise Ericsson Research Canada. Ces exigences consistaient à implémenter un modèle multidimensionnel des données des mesures en utilisant une technologie d'analyse multidimensionnelle de type « Datamart» basée sur OLAP. L'entrepôt devrait supporter les concepts des mesures décrites dans la norme ISO 15939 (ISO/IEC Std 15939, 2007), PSM (McGarry, 2001), voir l'annexe VII pour la liste des exigences).

L'entrepôt doit permettre la gestion et le contrôle des mesures au niveau processus et produits de la V&V dans les projets. Également l'entrepôt devrait supporter les processus du développement logiciel décrits dans le guide DO-178B.

Pour la quantification des processus et produits de V&V dans un cycle de vie pour la construction du logiciel critique nous proposons de suivre les exigences de la norme IEEE 1012 (IEEE Std 1012, 2004). Cette norme explique à un niveau détaillé les livrables pour chaque phase du développement du logiciel critique. Une autre source des processus et produits de V&V sera prise des documents de nos partenaires CMC Electronics Inc. (CMC Electronics, 2007) et Verocel Inc. (Verocel, 2005). Une analyse détaillée, ainsi que les livrables pour cette analyse partielle sera présentée dans le chapitre 9.

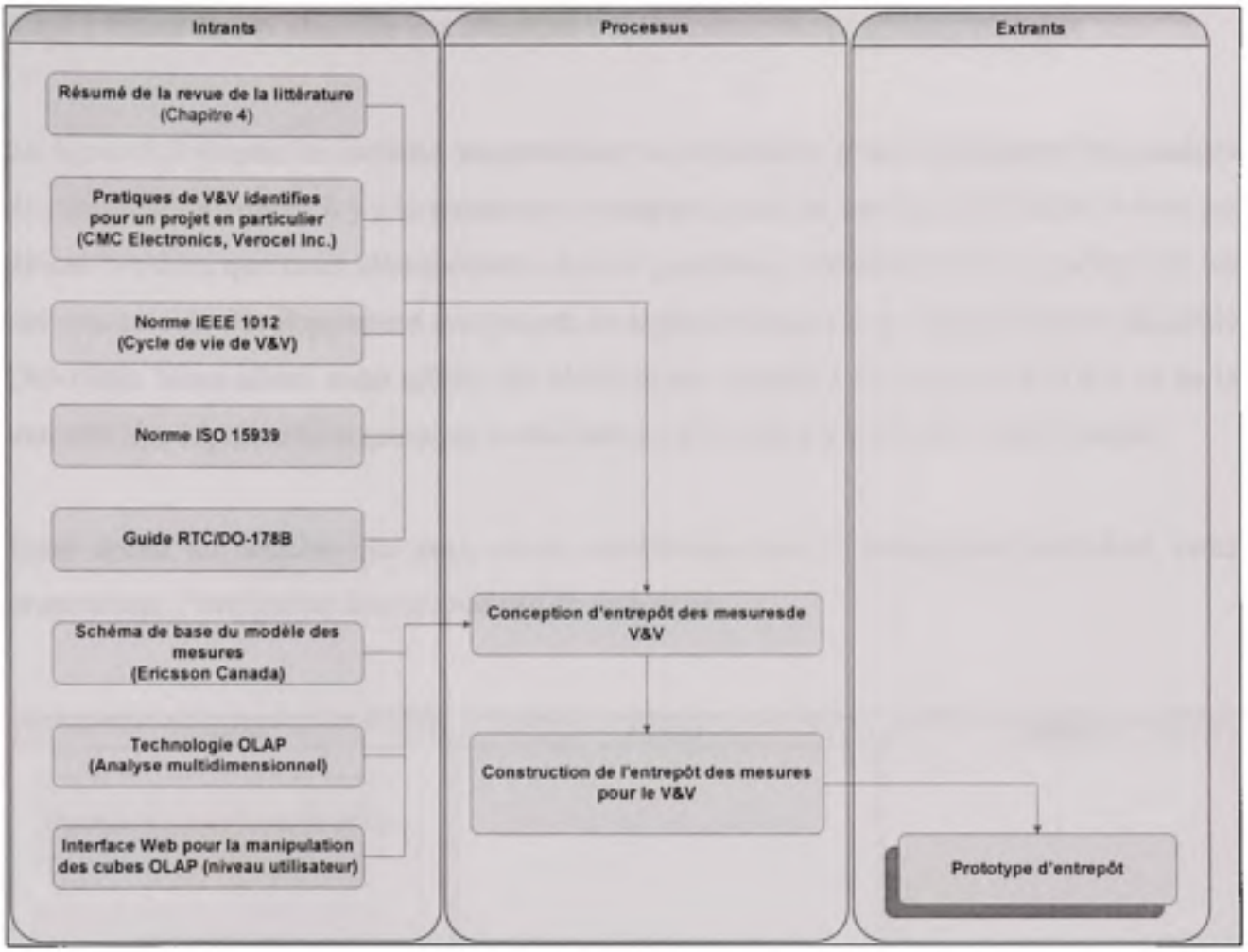

Figure 5.4 ***Phase 3 : Méthodologie de recherche pour l'entrepôt des mesures de la V&V.***

5.4.4 Phase 4 : méthodes d'évaluation des résultats de recherche

Dans cette section nous présentons les diverses étapes pour l'évaluation de nos résultats de recherche. Ceci comprend la participation des partenaires industriels à différent moments pour encadrer et vérifier divers livrables de notre proposition.

L'évaluation des résultats de recherche seront appelés :

- phase 4a, pour le référentiel des processus de V&V;
- phase 4b, pour l'entrepôt des mesures de V&V.

5.4.4.1 Phase 4a : évaluation des résultats : Référentiel des processus de V&V

La figure 5.5 illustre les intrants, les processus et les extrants pour l'évaluation des résultats de recherche. Au début il y a le prototype du plugiciel pour les pratiques de V&V, ensuite les divers livrables que nous allons obtenir de nos partenaires industriels et qui portent sur les cas concrets du développement des projets du logiciel critique avec la certification du guide DO-178B. Nous allons aussi utiliser les résultats par rapport à l'analyse de la V&V et de la maturité des organisations que nous avons faite pour le projet à la NASA IV&V Facility.

Étant donné les intrants que nous avons mentionnés dans le paragraphe précédent, nous proposerons l'évaluation de nos résultats de recherche.

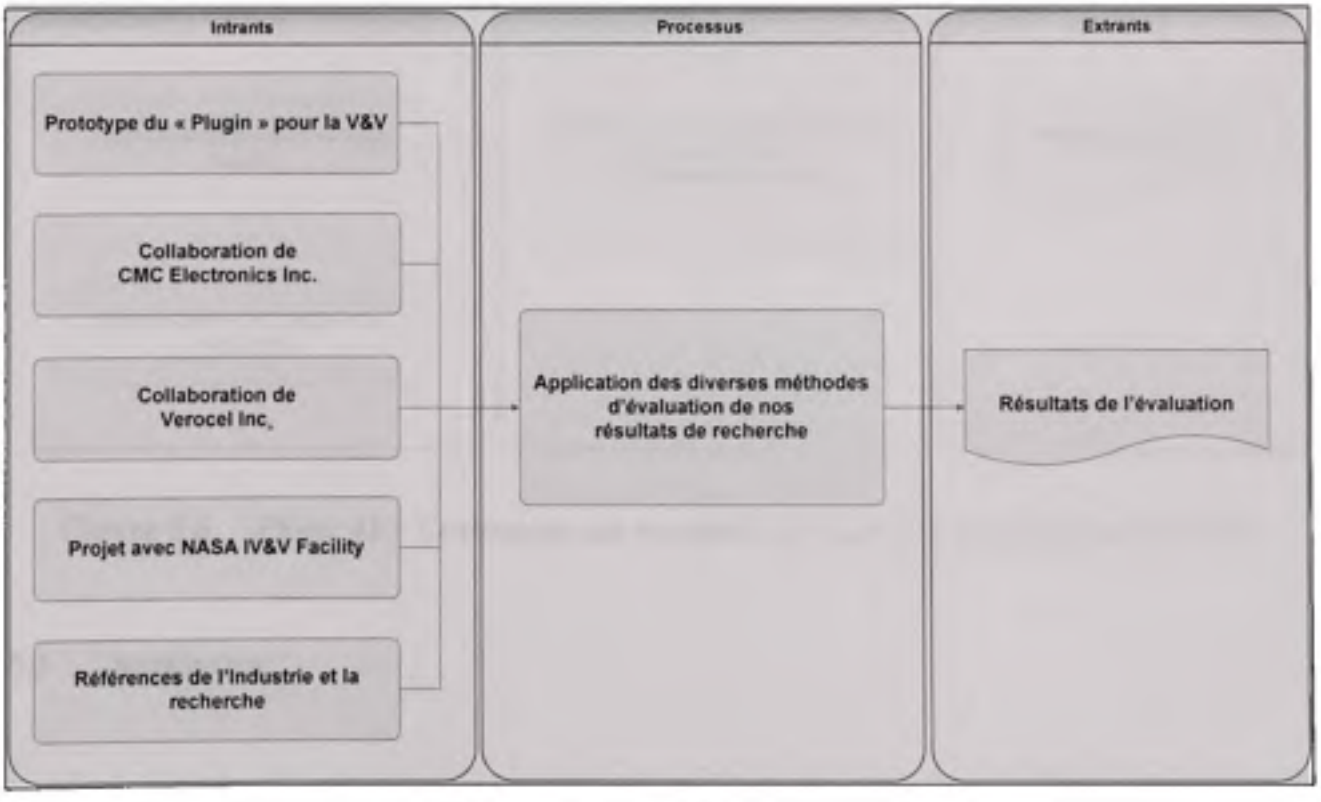

Figure 5.5 ***Phase 4a : Évaluation des résultats du modèle des processus de V&V.***

5.4.4.2 Phase 4b : Évaluation des résultats : Entrepôt des mesures de V&V

Nous allons faire appel éventuellement à nos partenaires industriels pour une évaluation des nos résultats de recherche. Une autre possibilité envisagée serait de soumettre nos résultats de recherche aux comités de révision pour la publication des articles. Diverses possibilités de d'évaluation des nos résultats de recherche ont été déterminées. Dans le chapitre 10 nous aborderons en détail l'évaluation de nos résultats de recherche pour le référentiel de V&V. Voir la figure 5.6 ci-dessous :

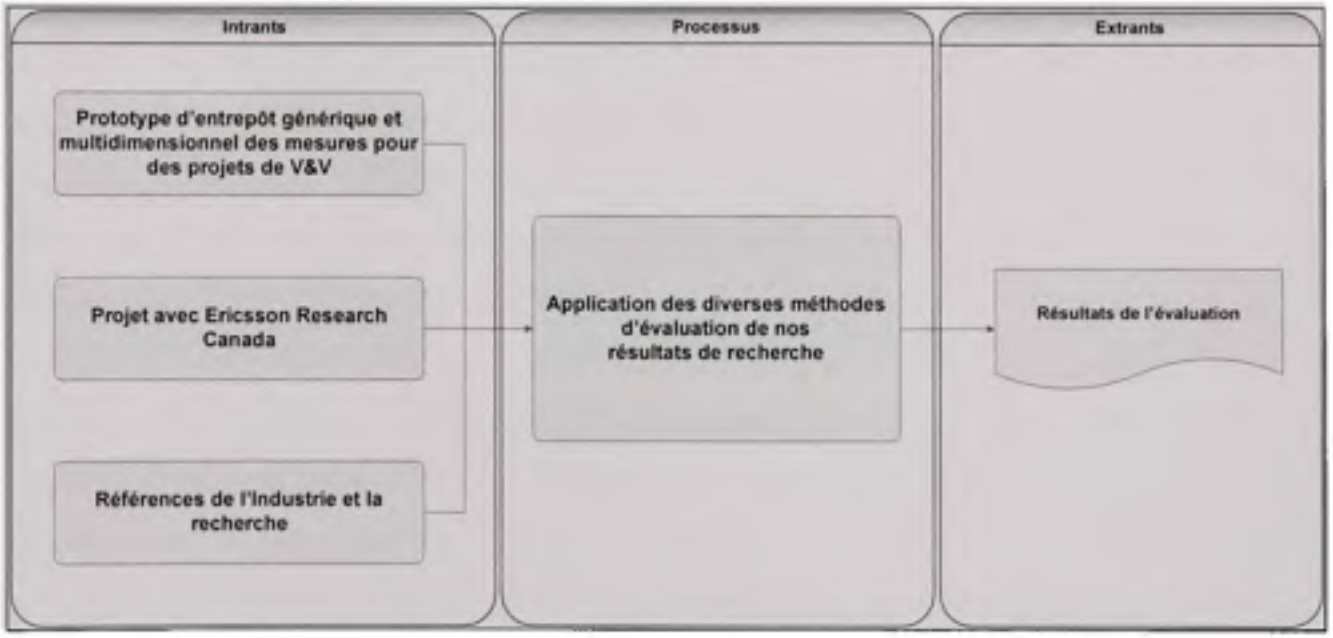

Figure 5.6 ***Phase 4b : Évaluation des résultats de l'entrepôt des mesures de V&V.***

5.5 Sommaire

Dans ce chapitre nous avons fait une présentation de différentes étapes, activités et livrables pour la démarche de notre projet de recherche. Nous avons également présenté l'évaluation de nos résultats de recherche.

Nous avons aussi mentionné la participation des différentes entreprises qui ont participé dans le cadre de notre recherche. Dans les chapitres suivants nous allons montrer en quoi consisterait la contribution de ces entreprises partenaires.

CHAPITRE 6

ANALYSE ET MODÉLISATION DES PROCESSUS DE V&V

6.1 Introduction

Ce chapitre présente la phase 1 de la méthodologie de recherche qui porte sur les analyses préliminaires pour notre travail de recherche. Ces contributions nous ont permis de mieux orienter et définir notre recherche. Par exemple la sous-section « Analyse de processus de développement selon la guide DO-178B » nous a permis de bien assimiler la structure, principes, recommandations, intrants/extrants du guide par rapport aux processus de construction du logiciel. La section « Modélisation des processus de V&V » nous a permis d'identifier les forces et faiblesses entre le RUP et les exigences pour la V&V de la norme IEEE 1012. D'ailleurs nous avons également produit des modélisations partielles des processus de V&V avec l'outil RUP XDE.

Il est important de mentionner que le projet de production d'un plugiciel avec le RUP XDE a été abandonné en cours de projet comme étape de recherche étant donné qu'un outil « OpenUP » (« Open Source ») a fait son apparition dans la communauté du génie logiciel. OpenUP présente des possibilités pour la formalisation des processus du développement du logiciel similaires à ce qui avait été prévu pour RUP XDE. Alors, nous avons inclus comme une analyse préliminaire dans cette thèse la modélisation avec l'outil RUP XDE d'un possible plugiciel qui faciliterait l'implantation des pratiques de V&V telles que décrites dans la norme IEEE 1012.

Nous diviserons cette phase en deux parties : La Phase 1a, qui comporte l'analyse du guide DO-178B et la Phase 1b, qui comporte la modélisation des processus de l'IEEE 1012 avec l'outil RUP XDE. La figure 6.1 résume la phase 1 du travail de recherche incluant les deux modèles réalisés.

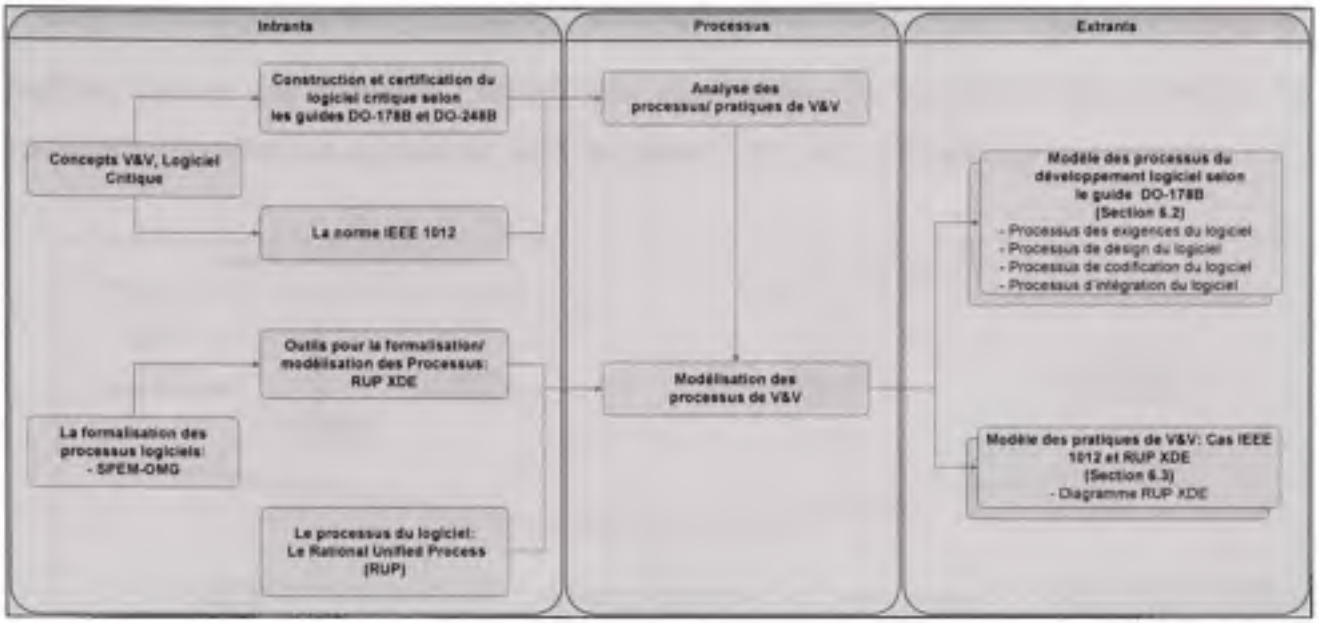

Figure 6.1 ***Schéma de l'analyse partielle des processus de V&V.***

6.2 Phase 1a : Analyse des processus selon le guide DO-178B

La première analyse correspond à une perspective schématisée des processus du développement logiciel selon les recommandations du guide DO-178B. Ces diagrammes sont basés sur le chapitre 5 du guide DO-178B. Nous analyserons les diagrammes basés sur les conseils et directives du guide DO-178B, DO-248B et les documents fournis par les partenaires CMC Electronics Inc. (CMC Electronics, 2007) et Verocel Inc. (Verocel, 2005), ainsi que des articles de recherche.

6.2.1 Processus des exigences du logiciel

Dans ce processus les exigences du système sont attribuées au logiciel (« System Requirements Allocated To Software – STRATS »). La figure 6.2 présente un diagramme proposé qui résume le processus des exigences tel que décrit dans le guide DO-178B. Cette figure montre que les exigences du logiciel sont le résultat des besoins du système. Une évaluation de la criticité au niveau de système est effectuée et cette criticité détermine la

criticité du logiciel. Divers composants du système vont influencer et dicter la criticité du logiciel, comme par exemple, l'architecture du système, les exigences fonctionnelles, les considérations relatives au matériel où le logiciel est exécuté, entre autres.

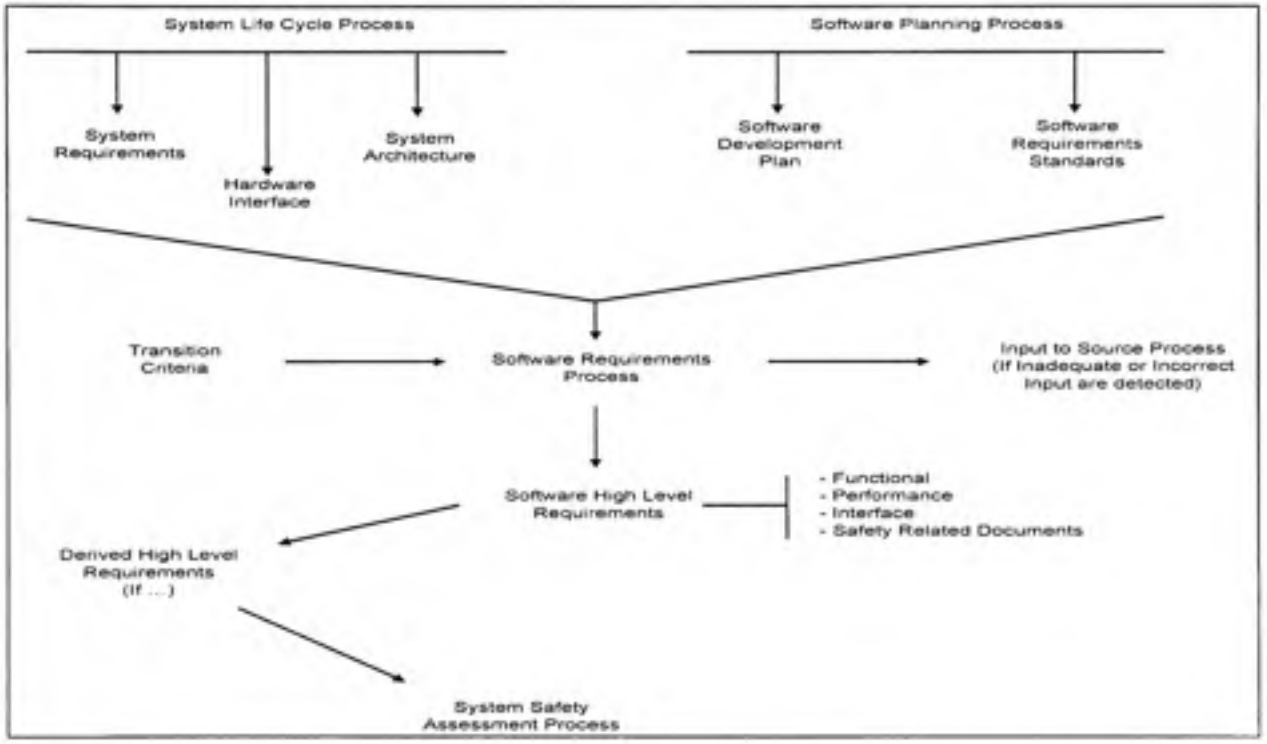

Figure 6.2 ***Diagramme des processus des exigences du logiciel selon la DO-178B.***

Selon le diagramme de la figure 6.2, le début de ce processus est initié avec l'analyse de toutes les exigences du système allouées au logiciel, â savoir : des exigences fonctionnelles, de sûreté, d'interface avec le logiciel. Cette analyse va donner lieu aux exigences de haut niveau pour le logiciel.

L'objectif de cette analyse est d'assurer que des exigences de système allouées au logiciel n'ont pas d'ambiguïtés, de contradictions ou de conditions non définies. Une clarification devrait être établie si des erreurs sont détectées. Également cette analyse sert à s'assurer que les exigences de haut niveau du logiciel sont vérifiables. La méthode de vérification peut inclure des analyses, des revues ou des tests.

Les exigences de haut niveau pour le logiciel incluent des exigences fonctionnelles, exigences de sûreté, exigences des interfaces, exigences de performances et capacités, exigences de données, exigences de programmation, exigences de design, exigences des algorithmes, entre autres.

Les exigences de haut niveau pour le logiciel doivent être documentées, incluant les exigences de sûreté. D'ailleurs toutes les exigences dérivées doivent être identifiées et documentées séparément pour qu'elles puissent servir d'intrants pour le processus d'évaluation de la sûreté du système.

Toutes les exigences de haut niveau autres que des exigences dérivées de hauts niveaux doivent être traçables à une ou plusieurs exigences du système. Ceci permettra la vérification de l'exécution complète des exigences du système et donnera la visibilité aux exigences dérivées.

6.2.2 Processus de design du logiciel

La figure 6.3 présente un diagramme résumé du processus de design tel que décrit dans le guide DO-178B.

Le but de cette activité est de produire l'architecture du logiciel et les exigences de bas niveau du logiciel, à partir des exigences de haut niveau. Pendant le processus de design l'analyse est effectuée pour s'assurer que :

- Les entrées au processus de codification sont correctes. Des entrées incorrectes ou insatisfaisantes seront clarifiées avec le processus des exigences ou le cycle de vie de système.
- Les exigences de haut niveau ne sont pas présentées en tant qu'exigences dérivées de bas niveau.
- L'architecture de logiciel et les exigences de bas niveau sont vérifiables. Des méthodes de vérification comme les revues, les analyses et les tests sont identifiées pour chaque exigence de bas niveau qui comporte des exigences dérivées de bas niveau.

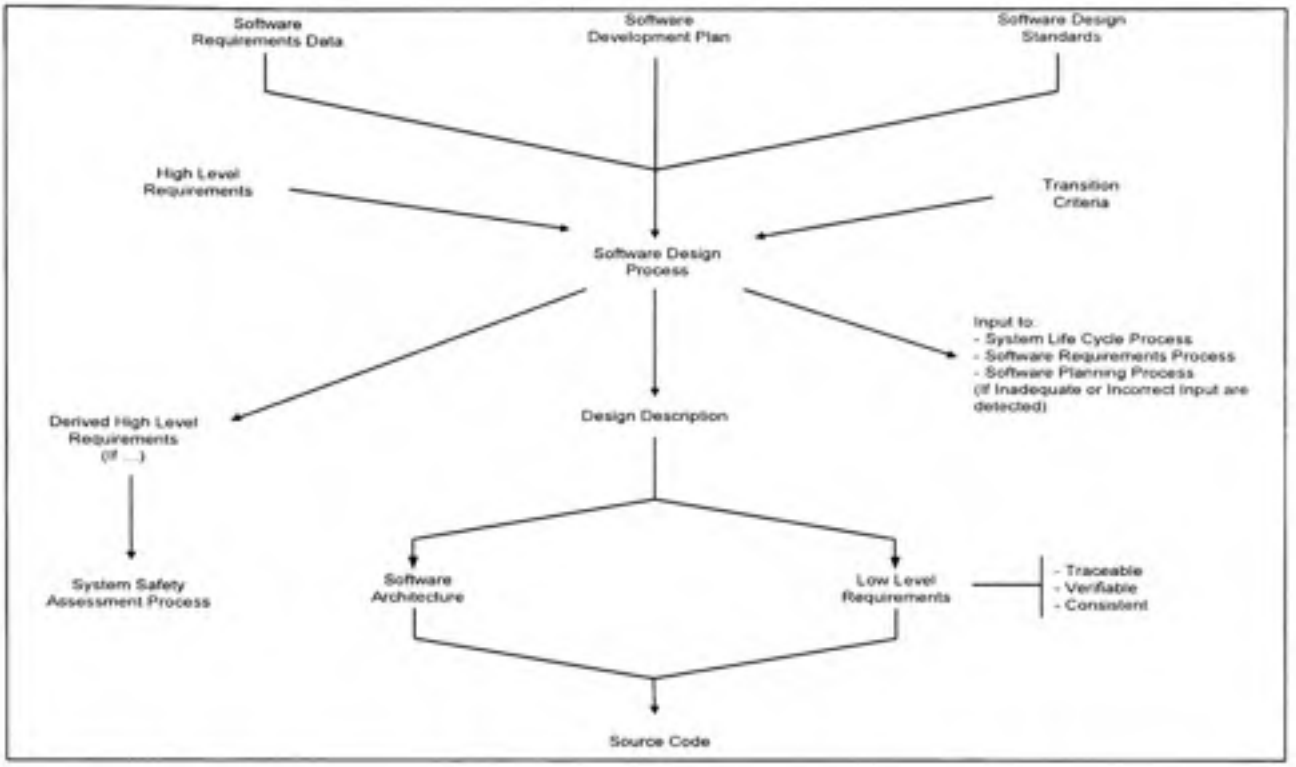

Figure 6.3 ***Diagramme des processus de design du logiciel selon la DO-178B.***

Le processus du design du logiciel produit le document de « Design Description ». Ce document contient l'architecture du logiciel et les exigences de bas niveau du logiciel. L'architecture du logiciel est définie comme l'organisation et l'interrelation des différents composants du logiciel qui vont implémenter les exigences de ce logiciel (RTCA/DO-178B, 1992).

Selon les documents fournis par nos partenaires nous pouvons affirmer ce qui suit (CMC Electronics, 2007) et Verocel Inc. (Verocel, 2005), (Hilderman V. et Baghi T., 2007):

- L'architecture du logiciel identifie les composants, les entités du logiciel et ses interactions comme par exemple les tâches, les librairies, entre autres.
- L'architecture du logiciel pourrait être développée en parallèle au « Low Level Requirements – LLR ».
- Le « Low Level Requirements – LLR » est appelé aussi « Low Level Design », et spécifie les fonctions et comment ces fonctions sont implémentées dans le logiciel.
- Le code source est implémenté à partir du LLR.

Le document de design du logiciel devrait contenir entre autres les informations suivantes :

- Une description détaillant comment le logiciel satisfait les exigences de haut niveau du logiciel incluant algorithmes, la structure des données, et comment les exigences du logiciel sont allouées dans les composants et tâches.
- Une description de l'architecture du logiciel avec la définition de la structure du logiciel pour implémenter les exigences.
- Une description des entrés et sorties.
- Le flux des données et du design.
- La limitation des ressources, et des alternatives pour gérer les problèmes.
- L'échéancier des procédures pour gérer la communication entre processeurs, tâches, interruptions, etc.
- Les méthodes de partitionnement utilisées
- La description des composants logiciel : s'ils sont nouveaux, réutilisés, etc.
- Les exigences dérivées de ce processus.
- Le code désactivé (s'il existe), dan le cas de réutilisation.
- Une structure logique du design qui soit traçable aux exigences de sûreté du système.

6.2.3 Processus de codification du logiciel

L'objectif du processus de codification du logiciel est de produire un code source qui soit traçable, vérifiable et qui correctement implémente les exigences de bas niveau du logiciel. Le code source devrait être conforme avec l'architecture du logiciel définie dans le document de la description du design (« Design description ») et les standards de codification du logiciel.

La traçabilité entre les exigences de bas niveau et le code source est fournie afin de permettre la vérification de l'implémentation complète des exigences de bas niveau et de l'absence du code non documenté. Le code source produit est testé pour assurer la conformité avec les exigences de bas niveau du logiciel.

La figure 6.4 propose un diagramme résumé du processus de codage tel que décrit dans le guide DO-178B.

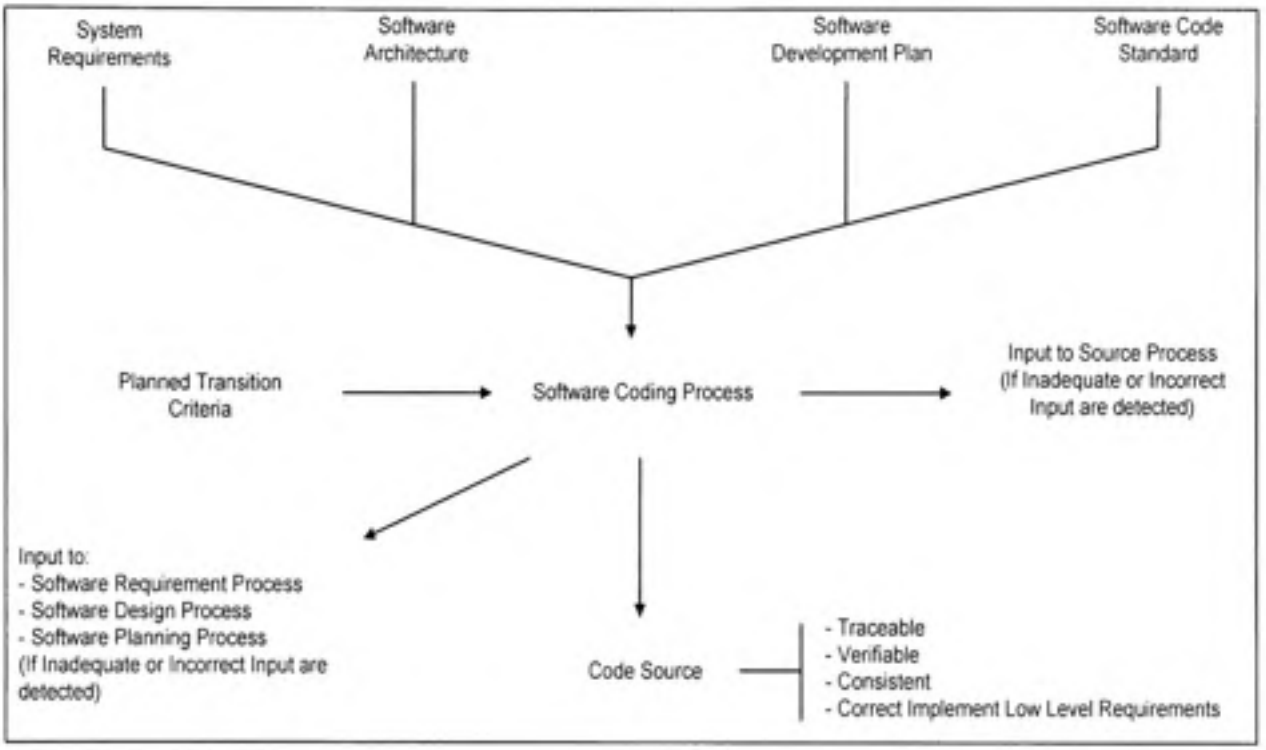

Figure 6.4 ***Diagramme des processus de codification du logiciel selon la DO-178B.***

6.2.4 Processus d'intégration du logiciel

Le processus d'intégration du logiciel comporte l'intégration des composants du logiciel ainsi que l'intégration du logiciel sur le matériel. Le logiciel devrait s'intégrer aussi avec les autres logiciels avec lesquels il va collaborer ou interagir ainsi que le matériel sur lequel il doit être installé et s'exécuter.

L'architecture logicielle, le code source, le code objet sont des éléments qui vont participer dans ce processus.

La figure 6.5 présente un diagramme résumé du processus d'intégration tel que décrit dans le guide DO-178B.

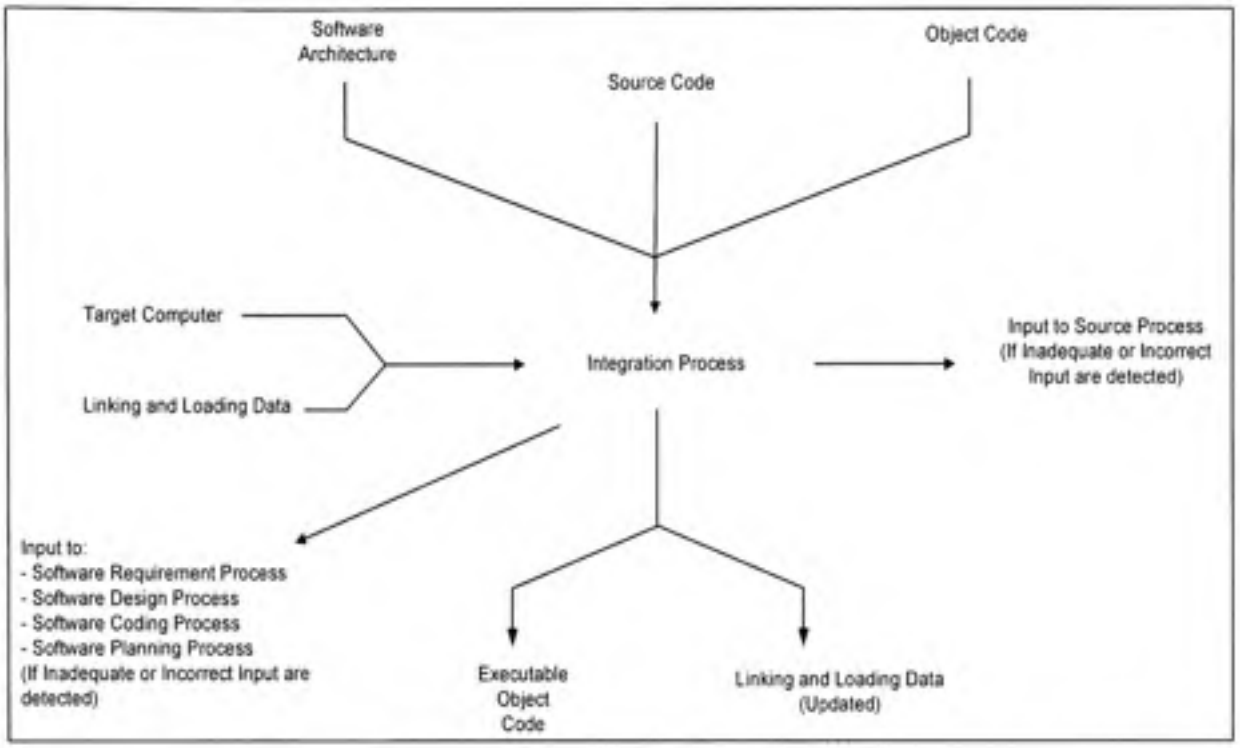

Figure 6.5 ***Diagramme des processus d'intégration du logiciel selon la DO-178B.***

6.3 Phase 1b : La modélisation des processus de V&V (RUP XDE & IEEE 1012)

6.3.1 But de l'analyse de la modélisation des processus de V&V

Cette section présente l'analyse des pratiques de V&V décrites dans la norme IEEE 1012 versus les pratiques de développement logiciel recommandées par RUP. Le but de cette phase de recherche est d'explorer la possibilité de combler les exigences de la norme IEEE 1012 pour la V&V du logiciel critique avec les processus unifiés tel que définis dans le RUP. Ceci a compris plusieurs analyses qui nous ont permis d'explorer et comprendre les particularités de la V&V et des projets du logiciel critique.

Dans le cadre de notre recherche par rapport à la norme IEEE 1012 et RUP nous avons produit plusieurs analyses (principalement des diagrammes) et conçu des éléments d'une proposition cohérente. Cette proposition a servi de spécifications pour débuter la modélisation des processus avec l'outil RUP XDE.

6.3.2 Modélisation de la traçabilité dans IEEE 1012

Cette section présente un exemple de modélisation de la norme IEEE 1012 avec l'outil RUP XDE. L'analyse que nous présentons traite des exigences de traçabilité demandée par la norme IEEE 1012 lors de l'implantation de la V&V dans les projets. Plus précisément nous montrons comment déterminer la traçabilité des exigences à partir des tests de V&V.

Après l'analyse de complétude entre RUP et la norme IEEE 1012 nous avons procédé à la modélisation de ces pratiques de V&V identifiées dans le cadre de notre analyse, avec l'environnement XDE de la suite de produits RUP.

La section 5.4 de la norme IEEE 1012 explique que les activités de V&V vérifient et valident les produits logiciels des activités et des tâches du processus du développement logiciel. La norme établit que les activités de V&V sont organisées en différentes activités à l'intérieur du processus du développement, à savoir: le concept de la V&V, les exigences de V&V, le design de V&V, l'implémentation de V&V, les tests de V&V et l'installation / inspection et checkout de V&V.

L'activité choisie pour illustrer notre analyse est par rapport aux tests de V&V et à la traçabilité. La section 5.4.5 de la norme IEEE 1012 (IEEE Std 1012, 2004) mentionne que l'objectif des tests de V&V est de s'assurer que les exigences du système allouées au logiciel ont été validées par l'exécution des tests d'intégration, tests de système et tests d'acceptation.

D'ailleurs il faut mentionner que la norme IEEE 1012 établit également que l'effort de V&V à déployer dans le cycle de vie du logiciel est dépendant du niveau de criticité du logiciel. Les tâches recommandées par l'IEEE 1012 pour les tests de V&V sont les suivantes:

1) Tâche: Analyse de la traçabilité.
2) Tâche: Génération de la procédure de test d'acceptation de V&V.
3) Tâche: Exécution de test d'intégration de V&V.
4) Tâche: Exécution de test de système de V&V.

5) Tâche: Exécution de test d'acceptation de V&V.
6) Tâche: Analyse des dangers.
7) Tâche: Analyse de sécurité.
8) Tâche: Analyse de risques.

Dans notre cas, nous présentons l'analyse et la modélisation de la tâche numéro 1 « Analyse de la traçabilité » (voir figure 6.6). Dans le diagramme nous constatons qu'il y a des artefacts qui sont des intrants pour l'analyse de la traçabilité, à savoir: les procédures des tests, les design des tests, les cas des tests, le plan de test, et également qu'il y a d'autres artefacts qui sont plutôt des rapports ou extrants du résultat de l'analyse de la traçabilité, à savoir: rapport des anomalies, rapport des tâches exécutées et rapport de l'analyse de traçabilité.

La tâche « Analyse de la traçabilité » de tests de V&V a pour objectif d'analyser en termes de conformité et complétude les relations entre les différents artefacts intrants de cette tâche (ie. les procédures des tests, les designs des tests, les cas des tests, le plan de test pour la V&V). La conformité s'assure qu'il existe une relation cohérente et logique entre les procédures des tests, les designs des tests, les cas des tests et le plan de tests de la V&V. La complétude vérifie que toutes les procédures des tests sont traçables au plan de la V&V pour le projet.

Nous avons remarqué que la tâche « Analyse de la traçabilité » de tests de V&V n'est pas identifiée de façon précise dans le RUP. Une correspondance explicite n'a pas été trouvée dans le RUP mais toutefois le RUP identifie des activités où il faut créer un plan de tests, des designs de tests, procédures des tests et cas des tests, pour le test du logiciel, mais pas avec la rigueur demandée avec norme IEEE 1012 et dans le cadre d'un plan de V&V.

Comme proposition on pourrait considérer avoir une activité dans le RUP « V&V traceability report generation » et son artefact serait « V&V traceability report », ce rapport devrait inclure des sections par rapport à : rapport des anomalies, rapport des tâches exécutées, et rapport de l'analyse de traçabilité.

L'annexe II présente d'autres exemples d'analyse et la modélisation des processus de V&V décrits dans la norme IEEE 1012 et le Rational Unified Process (RUP).

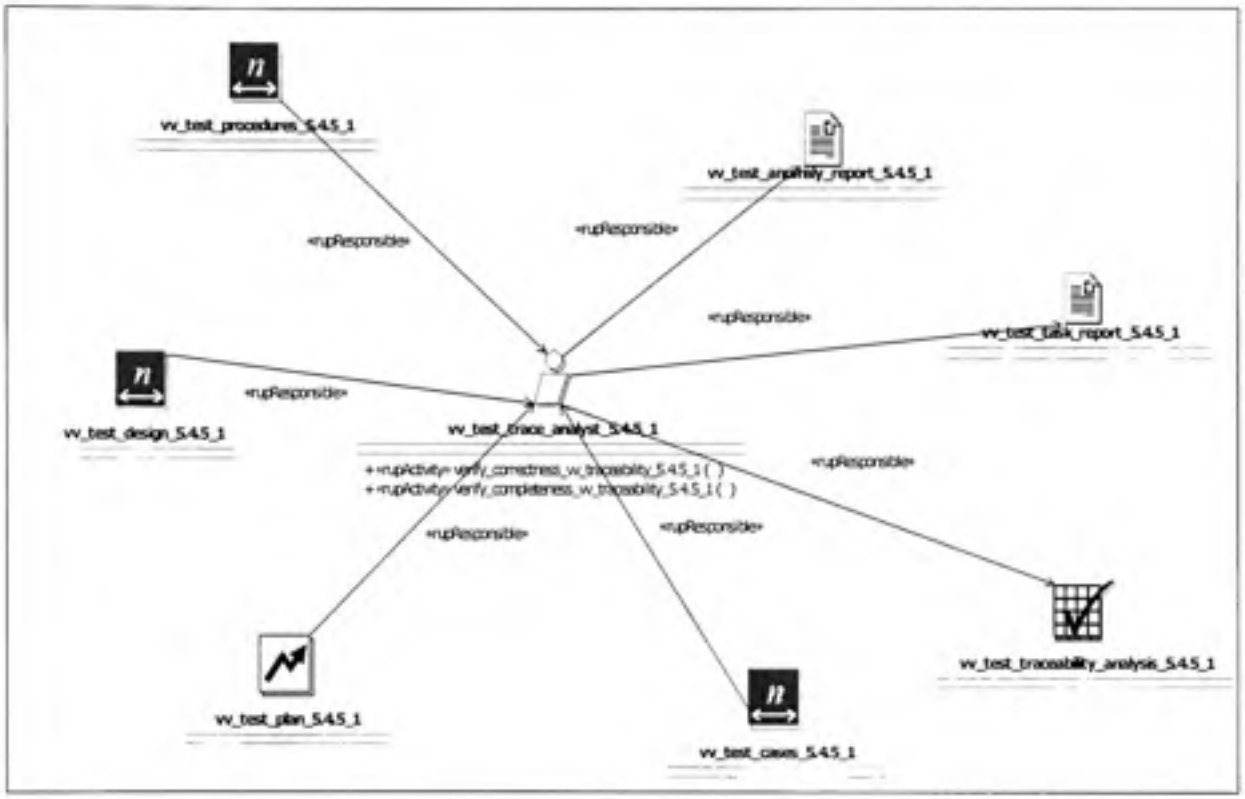

Figure 6.6 ***Modélisation d'analyse de la traçabilité selon IEEE 1012 et RUP XDE.***

Observation importante : Notre proposition originale de thèse comporte la modélisation des processus de V&V avec l'outil RPW. En fait, nous avons décidé d'inclure la référence à RPW car nous avons débuté notre modélisation des pratiques de V&V avec RPW. Dans le chapitre 6 nous avons montré une analyse de recherche qui a été faite dans le cadre de la modélisation de la norme IEEE 1012 et le RUP avec l'outil de RUP XDE (qui fait partie de RPW).

IBM propriétaire de RUP RPW a décidé de ne plus supporter le RUP XDE dans l'année 2006 (Bencomo, 2005). Depuis le « Eclipse Process Framework Composer (EPFC) » (Haumer Peter, 2007) fait son apparition dans le marché. Alors nous avons décidé de finaliser notre modélisation des processus de V&V avec OpenUP et EPFC.

6.4 Sommaire

Dans ce chapitre nous avons présenté les premiers livrables de notre recherche. Il s'agit d'un effort initial que nous avons fait, d'un côté pour améliorer notre compréhension des principaux processus du cycle de vie du logiciel présenté dans le guide DO-178B et d'un autre côté pour démarrer une analyse entre RUP et IEEE 1012 ainsi que débuter avec la modélisation des processus en utilisant l'outil RUP XDE.

L'analyse voulait montrer si RUP est capable de supporter les exigences de la norme IEEE 1012 pour la vérification et validation du logiciel. Nous avons publié un article de recherche à ce sujet : (Fuhrman. Djlive. Palza, 2003). Le chapitre aussi montre un des diagrammes résultat de la modélisation que nous avons faite avec l'outil RUP XDE.

CHAPITRE 7

ANALYSE ET CONCEPTION DES PRATIQUES DE V&V POUR LE REFERENTIEL DE V&V

7.1 Introduction

Ce chapitre présente la phase 2 de la méthodologie de recherche par rapport à la conception du modèle des processus de V&V. Dans ce chapitre nous analysons les objectifs du guide DO-178B et les processus d'OpenUP. Nous identifierons et proposerons de nouvelles pratiques dans le cadre d'OpenUP afin de combler des objectifs de la DO-178B pour la construction du logiciel critique. En particulier dans ce chapitre nous proposerons des pratiques pour les exigences du logiciel en concordance avec le DO-178B et l'OpenUP. Les autres processus de vérification mentionnés dans le guide ont été analysés et les résultats sont présentés en tant qu'annexe de cette thèse.

7.2 Démarche pour la conception du modèle des processus de V&V

Nous avons déterminé que la conception du modèle de processus de V&V serait basée sur la proposition d'une structure complète, équipée avec des directives, processus, procédés, outils. Nous avons utilisé la méthodologie OpenUp pour incorporer les principes du processus unifiées dans notre référentiel. Une des caractéristiques importantes que devrait avoir le modèle est la flexibilité à s'adapter aux différents types de projets dans le contexte du logiciel critique. Cette flexibilité a été atteinte avec l'outil « Eclipse Process Framework Composer (EPFC)» que nous allons utiliser pour bâtir le modèle des processus de V&V et qui sera présenté dans le chapitre 8.

D'ailleurs la partie plus importante de notre proposition est que le référentiel de V&V observe une harmonisation avec les activités décrites dans le guide RTCA DO-(RTCA/DO-178B, 1992) ainsi qu'avec les autres cadres comme le CMMI (SEI, 2006). Cette

harmonisation de notre référentiel devrait faciliter l'implantation des objectifs qui sont liés à la certification des logiciels critiques pour le domaine de l'aérospatial et avec le guide DO-178B.

Un aspect important dans la démarche de notre recherche a été l'accès privilégié à des documents de projets de logiciel critique certifiés DO-178B. Ces documents nous ont permis d'encadrer tout notre travail de recherche. Ces documents ont été fournis par nos partenaires industriels: CMC Electronique (CMC Electronics, 2007) et Verocel Inc. (Verocel, 2005).

Après avoir étudié et analysé les différentes pratiques décrites dans les documents que nous avons utilisés comme intrants (voir figure 7.1), nous allons commencer à bâtir un prototype du référentiel des processus de V&V en concordance avec les objectifs du guide DO-178B qui sera présenté dans le chapitre 8.

L'ensemble des propositions reliées aux processus de construction du logiciel critique ont donné lieu à des spécifications pour le prototype. Ces spécifications ont été des intrants pour bâtir l'artefact logiciel plugiciel qui devrait être la structure logicielle pour implanter le référentiel de V&V dans les projets.

Le prototype a été conçu à partir des propositions des pratiques de V&V basées sur l'analyse de la DO-178B et les processus unifiées présentés dans OpenUP. Nous avons fait des propositions pour les 5 principaux processus du développement du logiciel critique décrits dans le guide DO-178B, à savoir : les exigences, le design, la codification, l'intégration et la vérification. Les résultats de notre analyse sont montrés dans ce chapitre (section 7.5.1) et dans les annexes III à VI.

7.3 Les exigences du logiciel dans le système

Toutes les approches rigoureuses au développement de logiciel impliquent une certaine forme de spécifications de conditions comme une phase de l'essai au niveau système destinée

à démontrer la conformité du comportement mis en application avec les conditions. Les conditions sont généralement définies dans une partie du développement de système. Vers la fin du développement, ces conditions sont alors vérifiées par un ensemble de cas de tests choisis pour démontrer le comportement exprimé par les conditions. Le progrès est souvent mesuré en termes de « conditions vérifiées avec succès» et des incitations d'exécution peuvent même être liées à réaliser une étape importante programmée basée sur la conformité. En plus de « Requirement-based Testing », DO178B fournit les détails spécifiques au sujet des techniques pour réaliser le « Test Coverage Analysis » basé sur la structure de code.

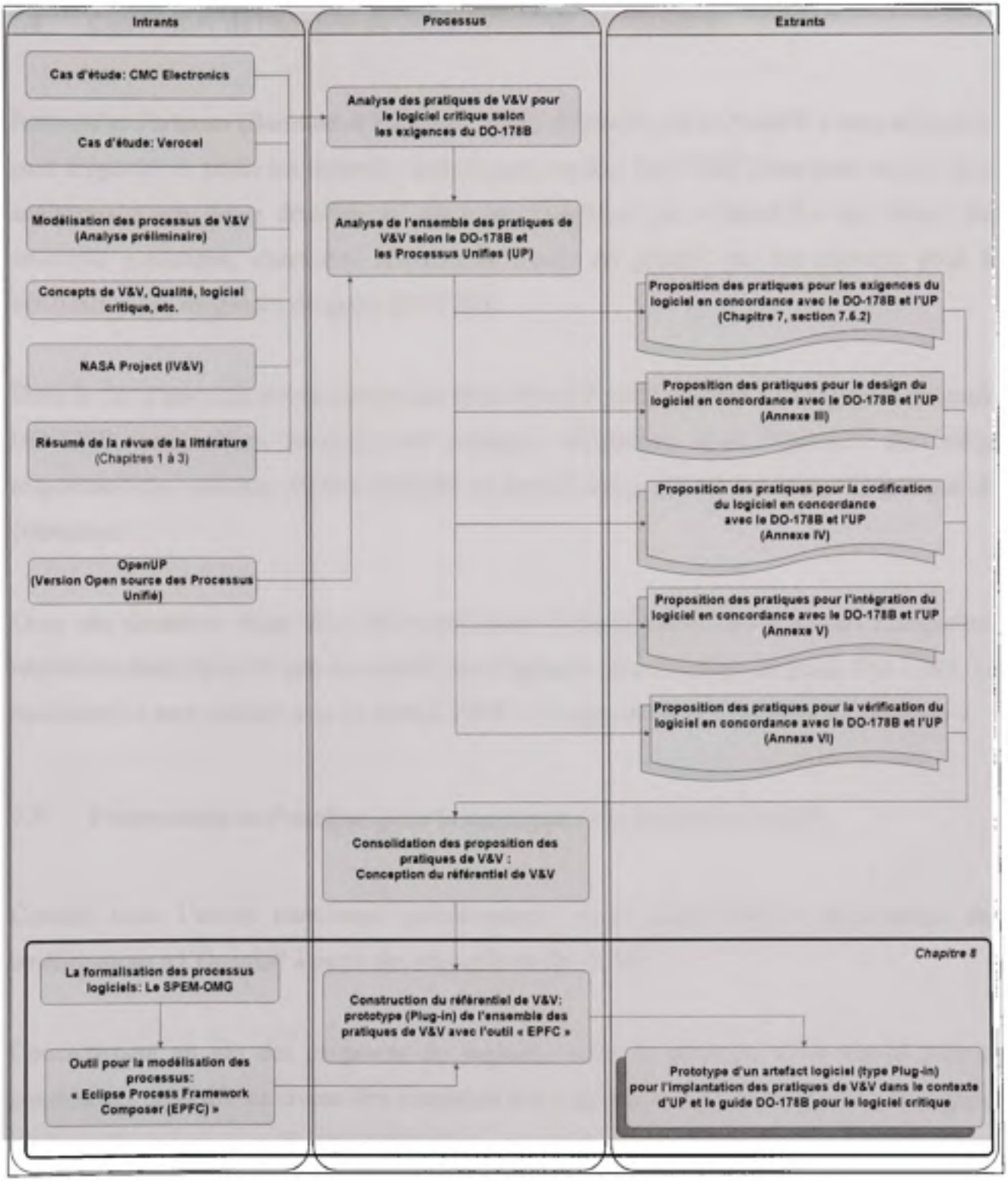

Figure 7.1 *L'analyse et la conception du référentiel de V&V.*

7.4 Couverture de l'analyse du guide DO-178B et l'OpenUP

Bertrand et Furhman (Bertrand et Fuhrman, 2008) affirment que « OpenUP » sans adaptation peut supporter en partie les objectifs imposés par le guide DO-178B. Dans cette section nous analysons d'une façon détaillée le degré de couverture de « OpenUP » au niveau des concepts, guidelines, check-list, produits de travail en général par les objectifs pour la vérification des exigences du guide DO-178B.

Dans le cas d'une manque de couverture dans OpenUP pour combler les exigences du guide DO-178B, nous allons proposer des pratiques pertinentes dans l'OpenUP (ex. rôles, responsabilités, activités, tâches, produits de travail, etc.) afin de compléter le manque de couverture.

Dans une deuxième étape nous allons présenter la modélisation des pratiques manquantes identifiées dans OpenUP afin de combler les exigences de ce chapitre du guide DO-178B. La modélisation sera réalisée avec le produit EPFC « Eclipse Process Framework Composer ».

7.5 Présentation de l'analyse pour la conception du plugiciel de V&V

Comme nous l'avons mentionné précédemment nous analyserons et proposerons des améliorations à l'OpenUP à partir des objectifs du DO-178B.

Concrètement au cas des exigences du logiciel : nous focaliserons notre travail afin de combler les objectifs au niveau des processus des exigences pour la construction du logiciel critique décrit dans le guide DO-178B. Pour ce travail, nous analyserons les objectifs décrits dans le tableau A-3 (Vérification du résultat des processus des exigences du logiciel) de l'annexe A du guide DO-178B et nous proposerons des pratiques, autrement dit : disciplines, rôles, tâches et produits de travail (artefacts, livrables, résultats) dans l'OpenUP en vue de combler les objectifs du guide par rapport aux exigences du logiciel.

Il faut noter que les objectifs de la vérification des processus des exigences du guide DO-178B font référence au tableau A-3 de l'annexe A du guide DO-178B (voir Annexe VII tableau VII.1.).

Les pratiques proposées constitueront des éléments de la conception du référentiel de V&V. Autrement dit, ces pratiques proposées serviront à la conception et construction de la discipline des exigences du plugiciel pour la V&V.

L'analyse et la conception des pratiques pour les exigences du logiciel sont présentées dans les tableaux 7.1 à 7.7. Ces tableaux incluent des analyses, des commentaires et la conception des pratiques de V&V pour les exigences du logiciel et l'OpenUP. Les tableaux ont la structure suivante :

- La définition de l'objectif selon la DO-178B;
- L'analyse de l'objectif par rapport aux pratiques type OpenUP;
- La Proposition des pratiques pour la conception du référentiel de V&V.

Il faut noter que les rôles associés à la vérification des résultats des processus des exigences peuvent être exercés comme suit :

- **Le V&V Analyste (ou groupe)**, qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel.

- **Un Analyste ou Développeur** tel qu'il est défini dans l'OpenUP. Les rôles doivent être adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

Les rôles devront utiliser les disciplines, tâches, produits de travail (« Work Products ») proposés dans la section « Proposition des pratiques pour la conception du référentiel de V&V » de chaque tableau de ce chapitre (tableaux 7.1 à 7.7).

Note

L'organisation des tableaux et la méthode d'analyse déployée pour le processus de gestion des exigences, que nous avons montrée dans ce chapitre, seront réutilisées pour l'analyse des processus restants du guide DO-178B, à savoir les processus de design, codification, intégration, vérification. L'analyse et la proposition des pratiques pour la conception des processus de design, codification, intégration et vérification sont présentées dans les annexes III à VI.

Tableau 7.1

Analyse des résultats de vérification des exigences : Objectif 01

Définition de l'objectif 01:
"High-level requirements **comply** with system requirements"
Analyse de l'objectif :
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification (point de vue analyse/revue/test) de la **conformité** entre les exigences de **haut niveau du logiciel** et **les exigences du système**. Il est mentionné dans le guide que les aspects **fonctionnels, de performance et sûreté** du système devront être satisfaits pour les exigences de haut niveau du logiciel. Les exigences dérivées (ce résultat des processus du développement) doivent être aussi bien justifiées. Il faut noter que les exigences de haut niveau sont les exigences du logiciel développées à partir de l'analyse des exigences du système, des exigences de sûreté et de l'architecture du système. **Analyse de l'objectif du point de vue d'OpenUP :** Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système. Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Référence dans le guide DO-178B:
6.3.1a
Proposition des pratiques pour la conception du référentiel de V&V
Au niveau des disciplines, des tâches et des artefacts: Les disciplines, tâches et artefacts mentionnés ci-dessous devront être considérés comme obligatoire afin de combler les objectifs du guide DO-178B. • En référence à la tâche: « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait demander d'exécuter obligatoirement : ▪ La « Check-list « Supporting Requirements ». Cet artefact propose des recommandations pour vérifier les exigences incluses dans le logiciel. Il inclut la vérification des aspects relies à la performance, la fonctionnalité, la conformité avec des standards, entre autres ▪ Le guideline « Effective Requirement Reviews ». Cet artefact discute comment faire des revues pour évaluer entre autres la qualité des exigences. ▪ Le guideline « Supporting Requirements ». Cet artefact explique comment développer et utiliser les exigences fonctionnelles (Il existe une référence explicite à la sécurité mais pas à la sûreté). **Note :** Le concept « Supporting Requirements » mentionne que des aspects de sécurité (pas de sûreté) doivent être évalués dans la fonctionnalité du logiciel mais il n'y a pas non plus une référence explicite à la vérification ou à la conformité.

Tableau 7.2

Analyse des résultats de vérification des exigences : Objectif 02

Définition de l'objectif 02:
"High-level requirements are **accurate and consistent**"
Analyse de l'objectif :
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification que chaque exigence de haut niveau soit **précise, non ambiguë** (Claire), et **suffisamment détaillée**. L'objectif met aussi emphase sur l'**absence de conflits** entre ces exigences. **Analyse du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Référence dans le guide DO-178B:
6.3.1b
Proposition des pratiques pour la conception du référentiel de V&V
Au niveau des disciplines, tâches et des artefacts: Les disciplines, tâches et artefacts mentionnées ci-dessous devront être considérés comme obligatoire afin de combler les objectifs du guide DO-178B. • L'exécution de la tâche : « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait demander d'exécuter obligatoirement le suivant: ▪ Le guideline « Requirement Pitfals ». Cet artefact fait mention au fait d'éviter les ambigüités lors de l'écriture des exigences. ▪ Le guideline « Supporting Requirements ». Cet artefact dans la section « Reliability» fait mention de la capacité des exigences à être précises. • L'exécution de la tâche : « Design the solution » de la discipline « Development ». Cette tâche devrait exécuter obligatoirement : ▪ Le « Step 1 : Understand requirement details» fait référence d'un côté évaluer les « uses cases » et « supporting requirements specifications », ainsi que clarifier « Ambigous our Missing Information » qui pourrait être détecté dans l'évaluation.

Tableau 7.3

Analyse des résultats de vérification des exigences : Objectif 03

Définition de l'objectif 03:
"High-level requirements are **compatible** with the target computer"
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification des conflits entre des exigences de haut niveau et les caractéristiques du matériel/logiciel de l'ordinateur sur lequel les exigences seront implantées. L'objectif fait mention particulièrement des conflits type temps de réponse, E/S. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Référence dans le guide DO-178B:
6.3.1c
Proposition des pratiques pour la conception du référentiel de V&V
Au niveau des disciplines, tâches et des artefacts: Les disciplines, tâches et artefacts mentionnées ci-dessous devront être considérés comme d'exécution obligatoire afin de combler les objectifs du guide DO-178B. • L'exécution de la tâche : « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait exécuter obligatoirement : ▪ Le « Step 7 » de la check-list "Supporting Requirements". Le « Step » indique la recommandation de s'assurer que les interfaces avec des systèmes doivent être considérées. ▪ Le guideline « Supporting Requirements ». Cet artefact mentionne dans les sections « Performance» et « Interface Requirements » la capacité des exigences à être précises ainsi qu'à garder la capacité et le débit requis. **Observation :** Les exigences mentionnées dans ce guideline pourraient mieux remplir les objectifs du guide DO-178B si on ajoute de façon explicite une référence à vérifier les conflits entre les exigences de haut niveau et le matériel/logiciel, et particulièrement vérifier les conflits type temps de réponse, E/S.

Tableau 7.4

Analyse des résultats de vérification des exigences : Objectif 04

Définition de l'objectif 04:
"High-level requirements are **verifiable**"
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite du fait de s'assurer que chaque exigence de haut niveau doit être vérifiable. **Analyse de l'objectif du point de vue d'OpenUP :** Selon le guide DO-178B le terme vérification implique l'analyse, la revue et les tests. Donc, les pratiques de « OpenUP » par rapport à ces concepts pourront être utilisées pour cet objectif.
Référence dans le guide DO-178B:
6.3.1d
Proposition des pratiques pour la conception du référentiel de V&V
Au niveau des disciplines, tâches et des artefacts: Les disciplines, tâches et artefacts mentionnés ci-dessous devront être considérés comme d'exécution obligatoire afin de combler les objectifs du guide DO-178B. • L'exécution de la tâche : « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait exécuter obligatoirement : ▪ Le « Step 5 », « Step 6 » de la Check-list « Quality of Good Requirements ». Ces « Steps » incluent des recommandations pour déterminer la vérifiabilité et la traçabilité des exigences. • L'exécution de la tâche : « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait exécuter obligatoirement : ▪ Les diverses check-lists et guidelines pour la vérification des exigences de haut niveau pourraient combler les exigences de la DO-178B. Par exemple : le guideline « Effective Requirement Reviews » dans la même la tâche : « Detail Requirements », et les autres sont également valides. • L'exécution de la tâche : «Create Test Cases» de la discipline «Test». Cette tâche devrait exécuter obligatoirement : ▪ Le « Step 7 » de la tâche : « Create Test Cases». Ce « Step » fait mention à la revue des exigences avant le testing

Tableau 7.5

Analyse des résultats de vérification des exigences : Objectif 05

Définition de l'objectif 05:
"High-level requirements **conform** to standards"
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la conformité des exigences aux normes pendant le processus des exigences. Il fait également référence à ce que chaque soit être justifié. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Référence dans le guide DO-178B:
6.3.1e
Proposition des pratiques pour la conception du référentiel de V&V

Au niveau des disciplines, tâches et des artefacts:
Les disciplines, tâches et artefacts mentionnés ci-dessous devront être considérés comme d'exécution obligatoire afin de combler les objectifs du guide DO-178B.
• L'exécution de la tâche : « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait exécuter obligatoirement :
  • L'exécution du « Step 9 » de la check-list "Supporting Requirements". Le « Step » mentionne une référence à la conformité des exigences.
  Observation : Il faut juste ajouter à l'exécution de cet artefact que tout écart respect à des standards devrait être documenté.

Tableau 7.6

Analyse des résultats de vérification des exigences : Objectif 06

Définition de l'objectif 06:
"High-level requirements are **traceable** to system requirements"
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification que toutes les exigences du système allouées au logiciel (fonction, performance, sûreté) aient été développées dans les exigences de haut niveau du logiciel **Analyse de l'objectif du point de vue d'OpenUP :** Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système.
Référence dans le guide DO-178B:
6.3.1f
Proposition des pratiques pour la conception du référentiel de V&V
Au niveau des disciplines, tâches et des artefacts: Les disciplines, tâches et artefacts mentionnés ci-dessous devront être considérés comme d'exécution obligatoire afin de combler les objectifs du guide DO-178B. • La tâche : « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait exécuter obligatoirement : ▪ L'exécution du « Step 6 » de la Check-list « Quality of Good Requirements ». Le « Step » indique des recommandations pour déterminer la traçabilité des exigences. **Note:** Le concept de traçabilité est aussi décrit dans la discipline « Requirements ».

Tableau 7.7

Analyse des résultats de vérification des exigences : Objectif 07

Définition de l'objectif 07:
"Algorithms are **accurate**"
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification des comportements des algorithmes particulièrement dans le contexte de discontinuités **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Référence dans le guide DO-178B:
6.3.1g
Proposition des pratiques pour la conception du référentiel de V&V
Au niveau des disciplines, tâches et des artefacts: Les disciplines, tâches et artefacts mentionnés ci-dessous devront être considérés comme d'exécution obligatoire afin de combler les objectifs du guide DO-178B. • L'exécution de la tâche : « Detail Requirements » de la discipline « Requirements ». Cette tâche devrait exécuter obligatoirement : ▪ Le « Step 8 » de la check-list « Supporting Requirements ». Le « Step » indique une référence à la création et validation des cas d'utilisation liés à des formules. ▪ Le « Step 3 » du guideline « Supporting Requirements ». Le « Step » fait mention en général à la fiabilité, vérification des erreurs, terminassions anormaux, entre autres. **Observation :** On pourrait considérer par extension l'exécution de cet artefact pertinent si on ajoute dans leur description, la vérification des comportements des algorithmes particulièrement dans le contexte de discontinuités.

7.6 Sommaire

Dans ce chapitre nous avons présenté une analyse des caractéristiques des pratiques d'OpenUP et du guide DO-178B. Nous avons établi des ponts en commun entre l'OpenUP et le guide DO-178B et nous avons interprété et organisé diverses recommandations pour la construction du logiciel et en particulier du logiciel critique. L'analyse que nous avons faite

d'OpenUP a servi pour identifier des pratiques qui pourraient être utilisées pour combler les objectifs du guide DO-178B. Nous avons également identifié des pratiques qui devraient être fortifiées dans l'OpenUP afin de combler les exigences du guide.

Les pratiques proposées sont principalement au niveau des disciplines, tâches, et produits de travail en général. Nous avons aussi identifié des rôles qui serviront à combler les exigences du guide DO-178B. Les propositions que nous avons montrés dans le chapitre vont donner lieu à des spécifications de l'artefact plugiciel que nous allons créer dans le chapitre suivant. Le plugiciel permettra d'identifier les pratiques de V&V pertinentes pour les adapter ou les personnaliser selon les exigences des projets. Dans ce chapitre nous avons fait une analyse du processus des exigences présenté dans le chapitre 5 du guide DO-178B.

CHAPITRE 8

CONSTRUCTION D'UN ARTEFACT LOGICIEL (PLUGICIEL) POUR L'IMPLANTATION DES PRATIQUES DE V&V

8.1 Introduction

Ce chapitre présente la phase 3 de la méthodologie de recherche par rapport à l'implémentation des composants du modèle des processus de V&V qui supporte le référentiel de V&V. Les diverses exigences du logiciel critique sont modélisées par la construction et l'implémentation du composant logiciel appelé plugiciel. Ce composant logiciel permettra la définition de différentes tâches, rôles, produits de travail (artefacts, livrables, résultats) du projet (i.e., acteurs, rôles, artefacts, processus, etc.) du logiciel critique qui devrait faciliter, dans un sens pratique, la réalisation des objectifs en concordance avec le guide DO-178B[2].

8.2 Le référentiel de V&V et la création du plugiciel avec OpenUP

Le plugiciel qui exprime le référentiel de V&V pour le projet permet de transformer la solution retenue en spécifications techniques d'une part et, organisationnelles pour le projet d'autre part. Le plugiciel du référentiel de V&V facilite la définition de tout ce qui permet d'intégrer et d'implanter efficacement la V&V dans les projets.

Nous avons déjà mentionné que le guide DO-178B définit l'ensemble des exigences de V&V à respecter pour certifier le logiciel critique. Ces exigences doivent être traduites en objectifs

---

[2] L'artefact logiciel « Plug-in ou plugiciel pour la V&V » (i.e., l'application exécutable) décrit dans ce chapitre est disponible pour les membres du Jury comme un livrable séparé de cette thèse.

de V&V pour le projet. Le référentiel joue un rôle important dans le projet car il facilite la définition et la gestion de ces exigences par le biais du plugiciel. Le plugiciel est un composant logiciel qui confère au référentiel la possibilité de définir ce qui doit être fait et pourquoi, c'est-à-dire la définition de l'essentiel du référentiel.

Le guide DO-178B spécifie un ensemble de processus, des objectifs et des artefacts pour chaque phase de la construction du logiciel et en fonction de leur criticité. Le référentiel est construit d'une façon qui incorpore un ensemble de pratiques qui facilitent l'implantation de la V&V, dans les termes du guide, pour un projet donné du logiciel critique.

Cette perspective du référentiel répond aux questions qui vont servir à leur l'implantation, à savoir:

- quels objectifs de V&V nous allons satisfaire dans un projet de logiciel en particulier?
- et que faut-il faire pour cela?

On peut dire qu'il s'agit de particularité du référentiel pour faciliter l'adaptation des pratiques de V&V dans différents contextes. Autrement dit, on construit un modèle « abstrait » de V&V en fonction de l'exigence particulière du projet critique. Cette caractéristique du référentiel indiquera ce qui doit être fait pour atteindre, en termes des pratiques, les objectifs de la DO-178B dans le projet.

Le référentiel de V&V adapté pour les besoins d'un projet fixe les frontières du système à évaluer, les limites du projet de V&V. Ce plugiciel permet de réaliser et de contrôler par la suite des produits et processus de V&V, ainsi que d'encadrer tout le travail à effectuer au cours de l'implantation de la V&V, puisqu'il aide à formaliser les objectifs et contraintes de qualité que toute solution devra viser et respecter.

Dans une étape suivante, les objectifs de V&V identifiés dans le plugiciel, en concordance avec DO-178B, seront traduits en tâches concrètes à réaliser pour le projet. Ces tâches seront directement liées au processus de développement tel que présenté dans l'OpenUP.

Au niveau de l'analyse de la couverture entre ce qui est spécifié dans l'OpenUp et ce qui est demandé par le DO-178B il faut noter ce qui suit:

- Le guide DO-178B présente la relation des Processus à Objectifs;
- L'OpenUP présente la relation Discipline à Tâches et par pas (« Steps »).

Alors étant donné les structures des l'OpenUP et le DO-178, nous avons fait les hypothèses suivantes:

- Un processus du DO-178B est équivalent à une discipline d'OpenUP;
- Un objectif du DO-178B est équivalent à un « step » d'une tâche d'OpenUP;
- Un ensemble d'objectifs du DO-178B est équivalent à une tâche d'OpenUP;
- Une tâche d'OpenUP est équivalente à un tableau des résultats défini dans l'annexe A du DO-178B (voir l'annexe VII de cette thèse).

La capacité du plugiciel de représenter et pouvoir adapter les divers acteurs, activités, flux, produits de travail, etc. à partir d'OpenUP, facilite entre autres la façon de comprendre, discuter et évaluer l'implantation des activités de V&V avant de les intégrer en tant que solution définie pour un projet concret. Cette particularité du référentiel permet de modéliser, dans le contexte de V&V, le comportement du projet, sa dynamique (en termes des artefacts, les intrants, les extrants et aussi les règles qui gouvernent ce comportement entre ces éléments).

Avec toutes les caractéristiques que nous avons incluses dans le référentiel de V&V on cherche à faciliter la définition détaillée des activités et tâches pour un projet du logiciel critique en particulier. Le plugiciel qui modélise le référentiel de V&V est construit avec les exigences du projet en termes de criticité. Ce plugiciel facilitera le dialogue entre les différents intervenants du projet, soit au niveau technique ou organisationnel.

8.3 Résumé des caractéristiques du plugiciel de la V&V

Après la présentation des diverses caractéristiques du plugiciel dans les différents chapitres, nous pouvons les résumer dans les termes suivants:

Adaptable: Cette caractéristique est définie de la façon suivante : Le référentiel de V&V pour être efficace doit permettre un niveau d'adaptation au contexte particulier des projets des logiciels critiques. Autrement dit, le référentiel de V&V est sensé donner des pratiques pour le choix à l'implantation qui conviennent, parmi toutes celles envisageables. C'est le cas, par exemple, lorsque l'on considère les différents niveaux de criticité dans les projets du logiciel critique. Le référentiel de V&V fournit un cadre qui permet de choisir les tâches, des rôles, les produits de travail les plus appropriés.

Itérative : Le référentiel de V&V est construit sous le concept du développement incrémental, autrement dit, chaque étape ou itération génère des informations qui peuvent remettre en cause les précédentes (changements).

Agrégation/Décomposition : Ce principe est matérialisé par le fait que l'on doit prévoir que le référentiel de V&V a la capacité d'examiner un projet à différents niveaux de granularité. Selon les besoins du projet V&V particulier, on est capable d'agréger ou décomposer la situation (i.e. au niveau des processus, activités, tâches, etc.). Il s'agit de trouver un niveau de détail qui convienne. On doit toujours pouvoir situer un détail dans son contexte général.

8.4 Contexte d'utilisation du plugiciel pour la V&V

La structure du plugiciel permet l'instanciation des pratiques pour la V&V dans les projets dans les termes suivants :

- par le biais de l'utilisation des composants définis directement dans l'OpenUP sans adaptation (lorsque l'objectif du DO-178B est comblé avec la pratique de l'OpenUP).

- par le biais de l'utilisation des pratiques de l'OpenUP avec un renforcement des exigences précises pour l'objectif du DO-178B (lorsque l'objectif du DO-178B est partiellement comblé avec la pratique de l'OpenUP).
- par le biais de l'utilisation des nouvelles pratiques dans l'OpenUP. Ces nouvelles pratiques sont spécifiques au guide DO-178B et elles n'existent pas dans l'OpenUP.

Ce type d'approche que nous avons incorporé dans le plugiciel de la V&V permet l'implantation par exemple de la V&V indépendante lorsque le logiciel est catégorisé comme hautement critique. Dans ce cas la V&V est exécutée par un groupe indépendant du groupe du développement.

8.5 Construction du plugiciel pour la V&V

Le plugiciel pour la V&V a été créé. à l'intérieur de l'environnement EPFC (Eclipse Process Framework Composer) version 1.2. Le plugiciel pour la V&V est appelé « VV_plugin » (voir figure 8.1). Le plugiciel que nous avons créé est composé de méthodes qui vont contenir les pratiques que nous avons définies dans le plugiciel.

Après la création du plugiciel pour la V&V, la méthode « Method Content » a été créée de façon automatique. C'est dans cette méthode que nous avons défini tous les composants du plugiciel pour la V&V (produits, guides, disciplines, etc).

Il existe aussi une sous-ensemble de la méthdoe « Method Content » appelée « Content Packages » qui nous facilite la possibilité de créer à la fois les autres « Content Packages » spécifiques pour le plugiciel de la V&V (par exemple avec des tâches). Ces « Content Package » permettent de mieux structurer et distinguer les différents composants du plugiciel. Les « Content Package » que nous avons créés sont (voir aussi figure 8.1) :

- Le « vv_copyright »;
- Le « vv_core_plugin»;
- Le « vv_core_wp ».

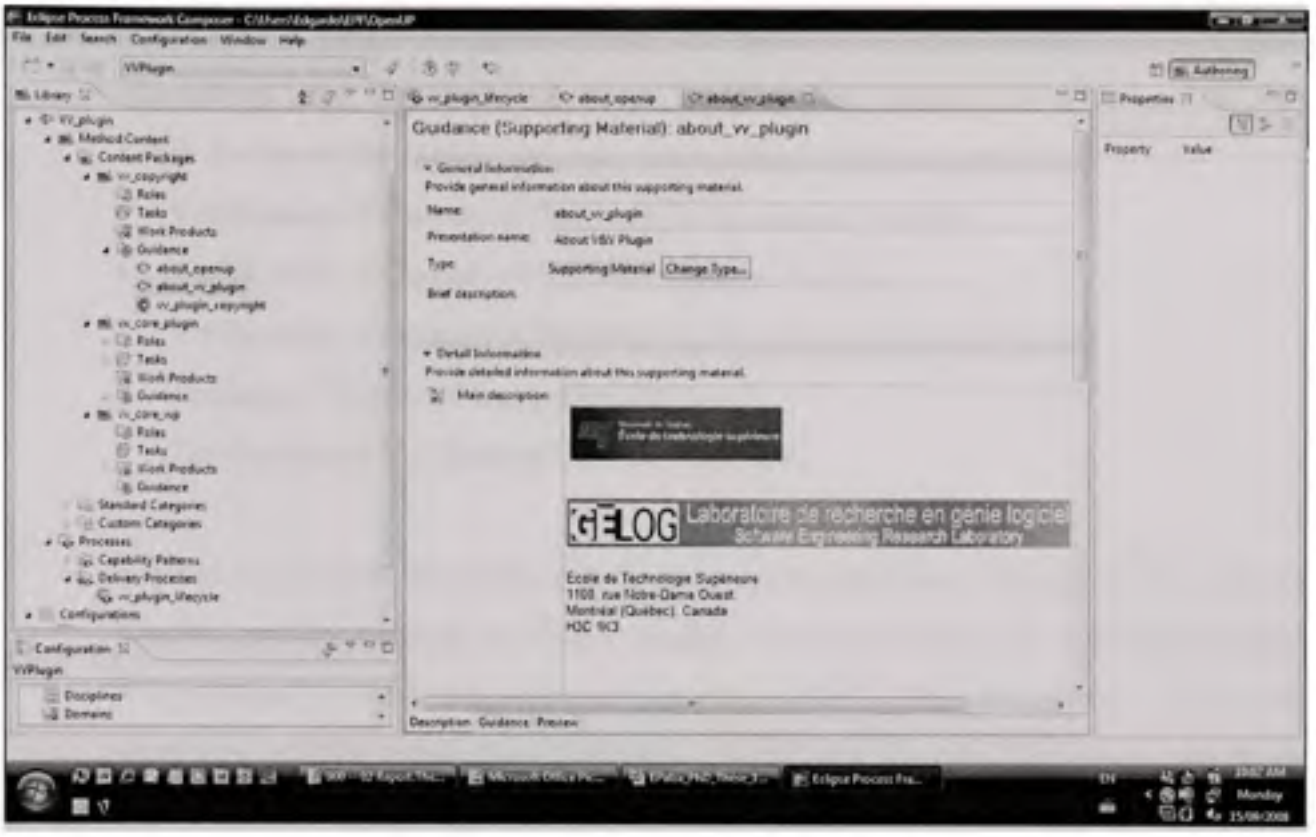

Figure 8.1 ***Les « Contents Packages » du plugiciel de la V&V.***

Voici une description des « Content Packages » que nous avons créés :

Le « package » « vv_copyright » contient toutes les informations concernant le « copy right » et droits d'auteur du plugiciel »; ils sont dans la forme de « supporting materials » et groupés sous la classification de « Guidance », dans ce cas nous avons suivi la structure des packages proposé dans le plugiciel pour l'OpenUP.

Le « package » « vv_core_plugin» contient tous les artefacts essentiels pour instancier la V&V dans les projets.

- la division « Guidance » contient un ensemble de concepts, composants, supports et autres qui sont importants pour l'implantation de la V&V conformément à des exigences du guide D0-178B (voir figure 8.2);

- la division « Tasks» comporte toutes les tâches ainsi que les pas (eg., « steps ») nécessaires pour l'instanciation de la V&V conformément aux exigences du DO-178B. Les noms des tâches que nous avons créées sont (voir aussi 8.2) :
  - Verification of Outputs of Software Requirement Process;
  - Verification of Outputs of Software Design Process;
  - Verification of Outputs of Software Coding & Integration Process;
  - Testing of Outputs of Integration Process;
  - Verification of Verification Processes Results.

- la division « Roles» comporte tous les rôles pertinents pour le plugiciel. Nous avons identifié un rôle principal, le « V&V Analyst » et 5 autres rôles qui vont contribuer au rôle principal : « V&V Requirement Analyst », « V&V Design Analyst », V&V Code Analyst », « V&V Integration Analyst », « V&V Verification Analyst » (voir figure 8.2). Ces rôles ont été identifiés en fonction de nos besoins des acteurs par chaque processus du développement décrit dans le DO-178B.

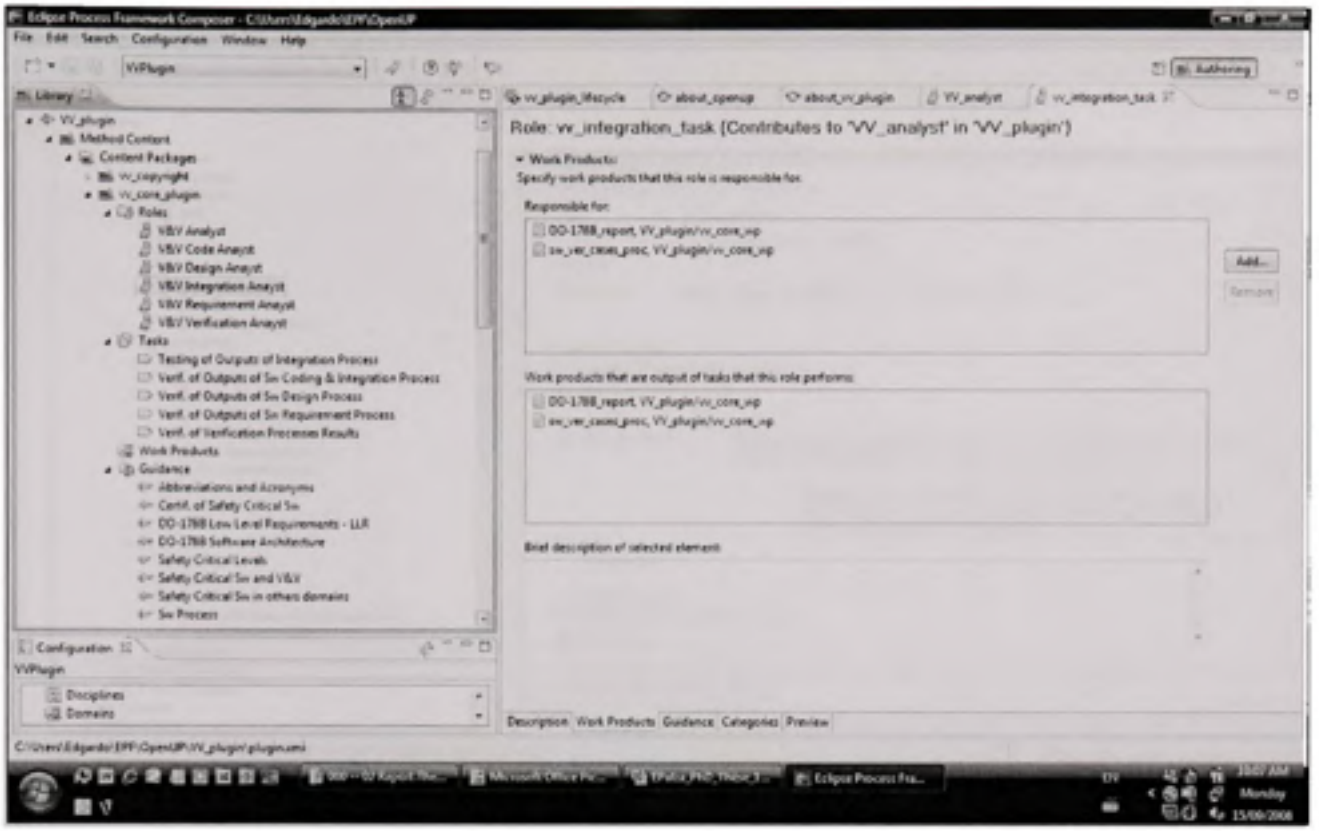

Figure 8.2 ***Les rôles dans le plugiciel de la V&V.***

Le « package » « vv_core_plugin» contient les produits de travail ou « Workproducts ». Les produits de travail servent comme des livrables ou rapports d'entrée/sortie afin de montrer l'accomplissement des objectifs du DO-178B. Les produits de travail sont mis à jour dans chaque objectif du guide DO-178B, voir figure 8.3.

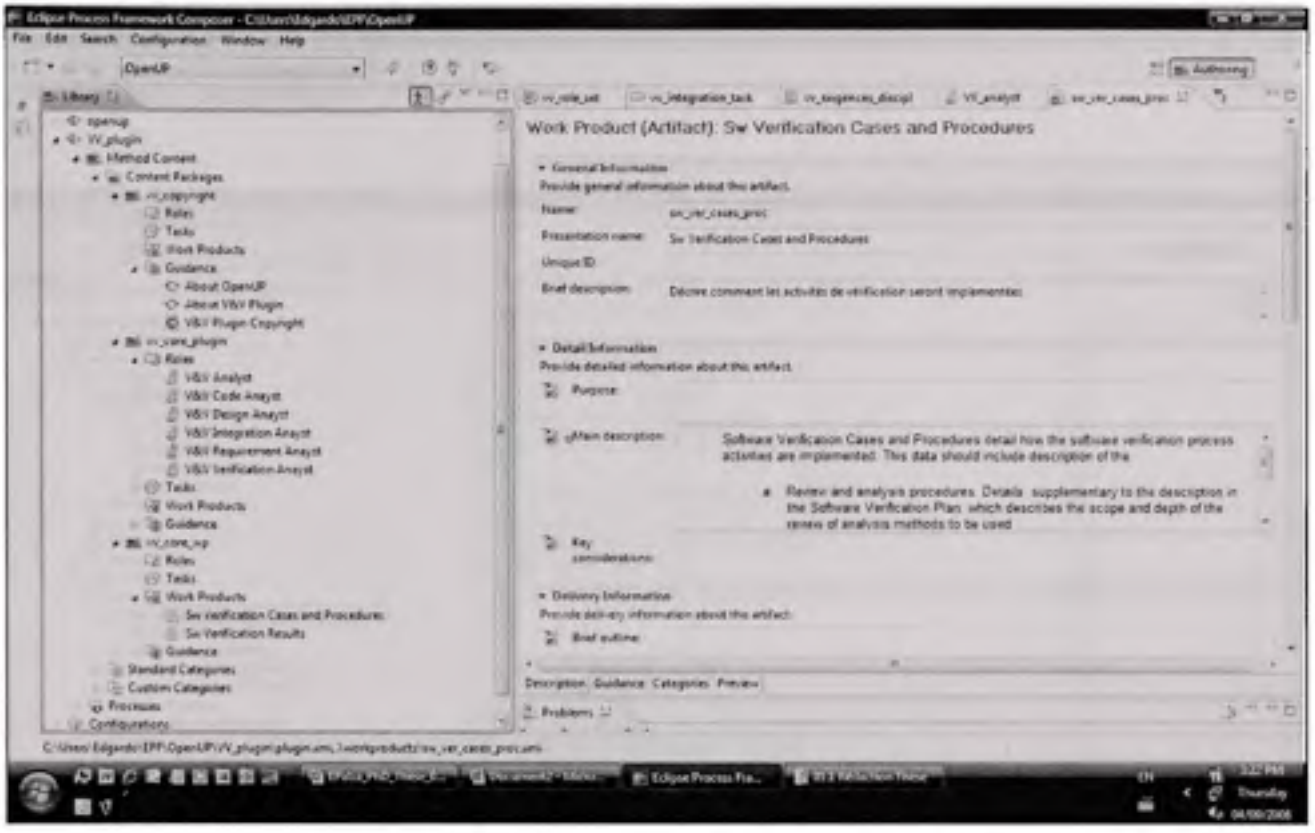

Figure 8.3 ***Les produits de travail dans le plugiciel de la V&V.***

8.6 Organisation du plugiciel pour la V&V

L'organisation qui sera générée par le plugiciel pour la V&V est définie dans l'environnement EPFC (Eclipse Process Framework Composer). C'est dans cet environnement qui nous avons défini toutes les pratiques de la V&V pour notre projet. Ce processus est appelé « Process Authoring ».

Le plugiciel défini avec l'EPFC, après un processus de génération, est passé à une version HTML qui est publiée dans un « Internet Browser » comme l'Internet Explorer. Le plugiciel est alors prêt à être utilisé par les usagers du projet.

Les pratiques de V&V dans le plugiciel sont organisées selon une structure proposée par l'EPFC. Voilà les deux catégories ou méthodes (voir figure 8.4), à savoir :

- « Standard categories » ;
- « Custom categories ».

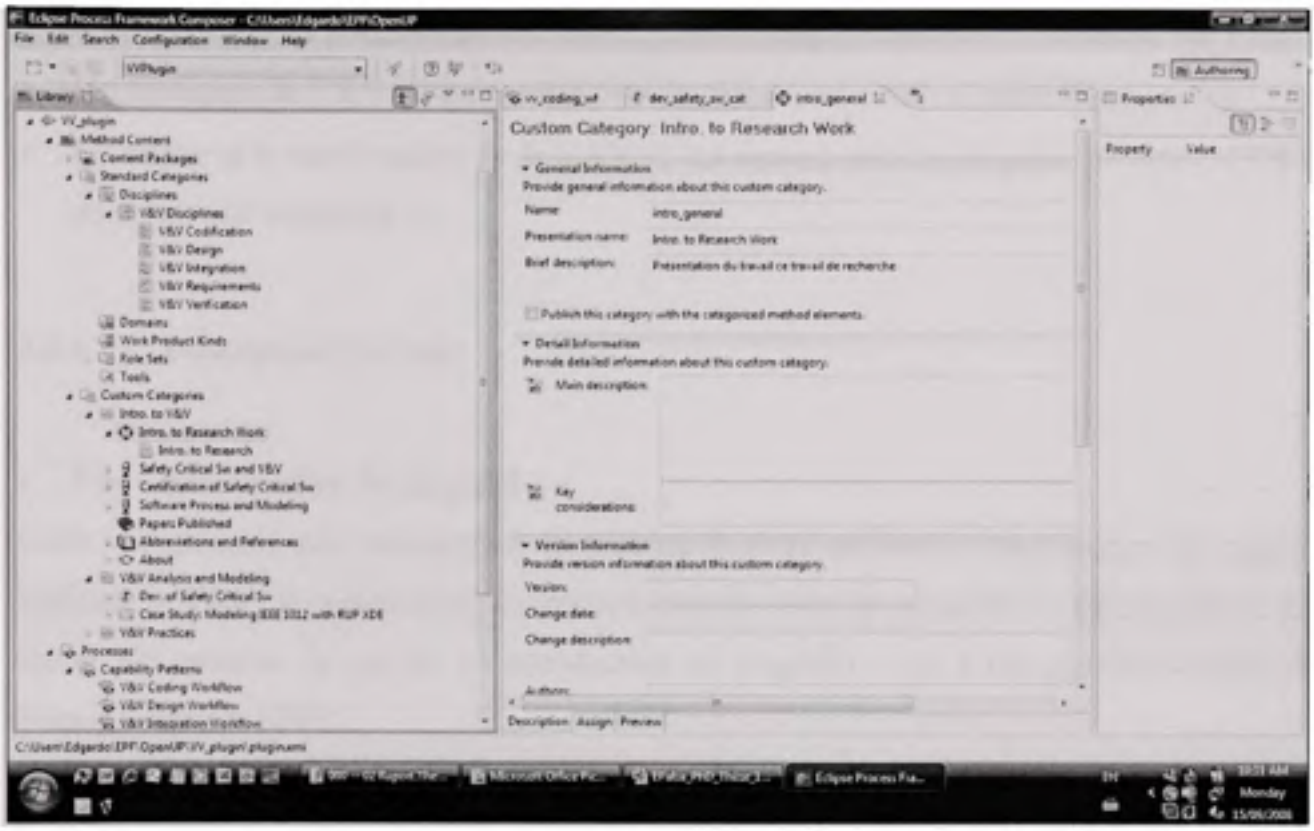

Figure 8.4 ***« Standard Categories » et« Custom Categories » du plugiciel de la V&V.***

Dans la catégorie **« Standard categories »** nous avons créé les différentes disciplines du plugiciel, à savoir (voir aussi figure 8.4) :

1. V&V Exigences;
2. V&V Design;
3. V&V Codification;
4. V&V Intégration;
5. V&V Vérification.

Ces disciplines sont associées à des tâches, des « workflows » et des artefacts de la V&V qui ont été définis pour le plugiciel.

Dans la catégorie « **Custom categories** » permet de créer les différentes vues ou perspectives du plugiciel (reflétés comme onglets dans le plugiciel au niveau de l'interface). Ces vues sont les suivantes :

- L'introduction au plugiciel (C'est montré dans le plugiciel comme : « Intro. to V&V »);
- Les pratiques de V&V (C'est montré dans le plugicielcomme : « V&V Practices »);
- L'analyse et la modélisation de la V&V (C'est montré dans le plugiciel comme : « V&V Analysis and Modeling »).

8.6.1 Description des vues

- **Vue : introduction du** plugiciel

Cette vue présente une introduction au contenu de cette recherche, les concepts du logiciel critique, la V&V, la certification du logiciel critique selon le guide DO-178B, et autres. La figure 8.5 présente la vue de « l'introduction au plugiciel » qui a été générée à partir de l'environnement EPFC.

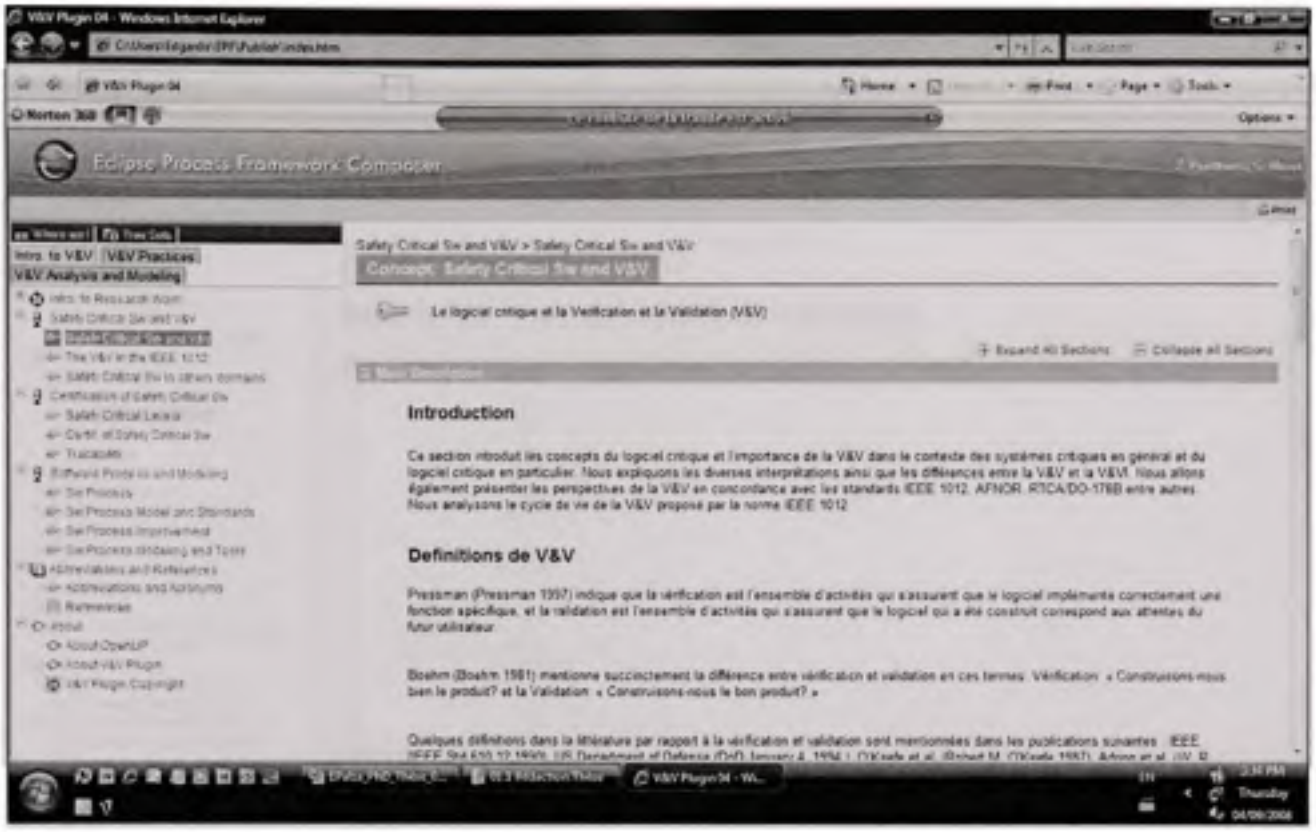

Figure 8.5 ***Vue « Introduction au* plugiciel *de la V&V ».***

- **Vue : pratiques de V&V**

  Cette vue présente l'ensemble des disciplines, tâches, et des rôles, artefacts, produits des travaux associés. La figure 8.6 présente la vue de « Pratiques de V&V» qui a été générée à partir de l'environnement EPFC.

  La vue pratique de V&V présente également le cycle de vie de la V&V pour le plugiciel. Ce cycle de vie a été défini en fonction des tâches d'OpenUP que nous avons proposées pour remplir les exigences du guide DO-178B. Ces tâches proposées sont présentées dans les tableaux du chapitre 7 et des tableaux des annexes III à VI.

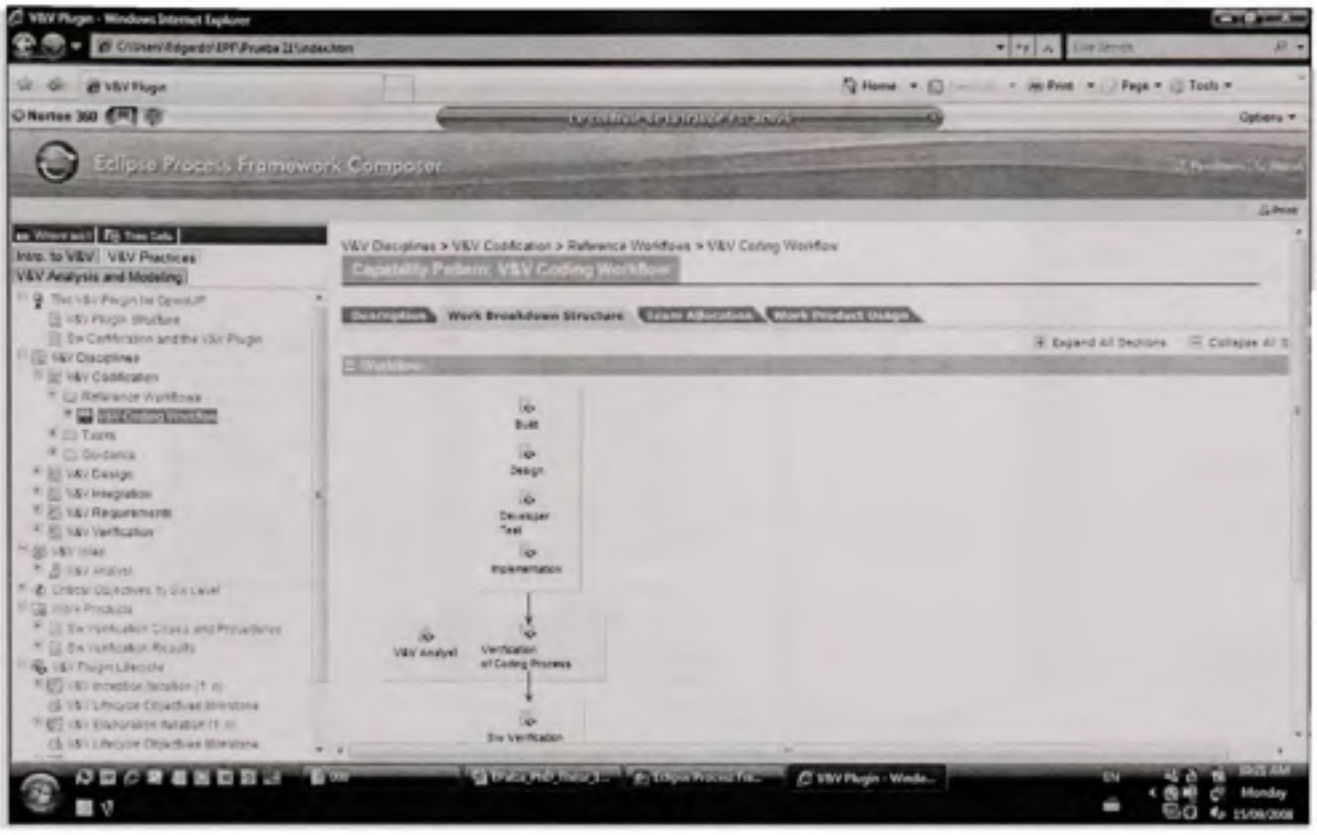

Figure 8.6 ***Vue « Pratiques de la V&V » du* plugiciel**

La figure 8.7 présente la construction du cycle de vie du logiciel selon l'OpenUP mais avec le contenu des pratiques de V&V. Le cycle de vie de la V&V est composé des itérations des phases du cycle de vie. Ces itérations comportent des activités, des tâches, des rôles, des « Steps », des jalons et des produits de travail (« Workproducts »).

D'ailleurs chaque itération est composée par un « Workflow » particulier. Le « Workflow » permet d'organiser la séquence d'exécution de l'ensemble de tâches pour le projet. Aussi l'ensemble des tâches est organisé comme une activité. Voir figure 8.8 pour les détails.

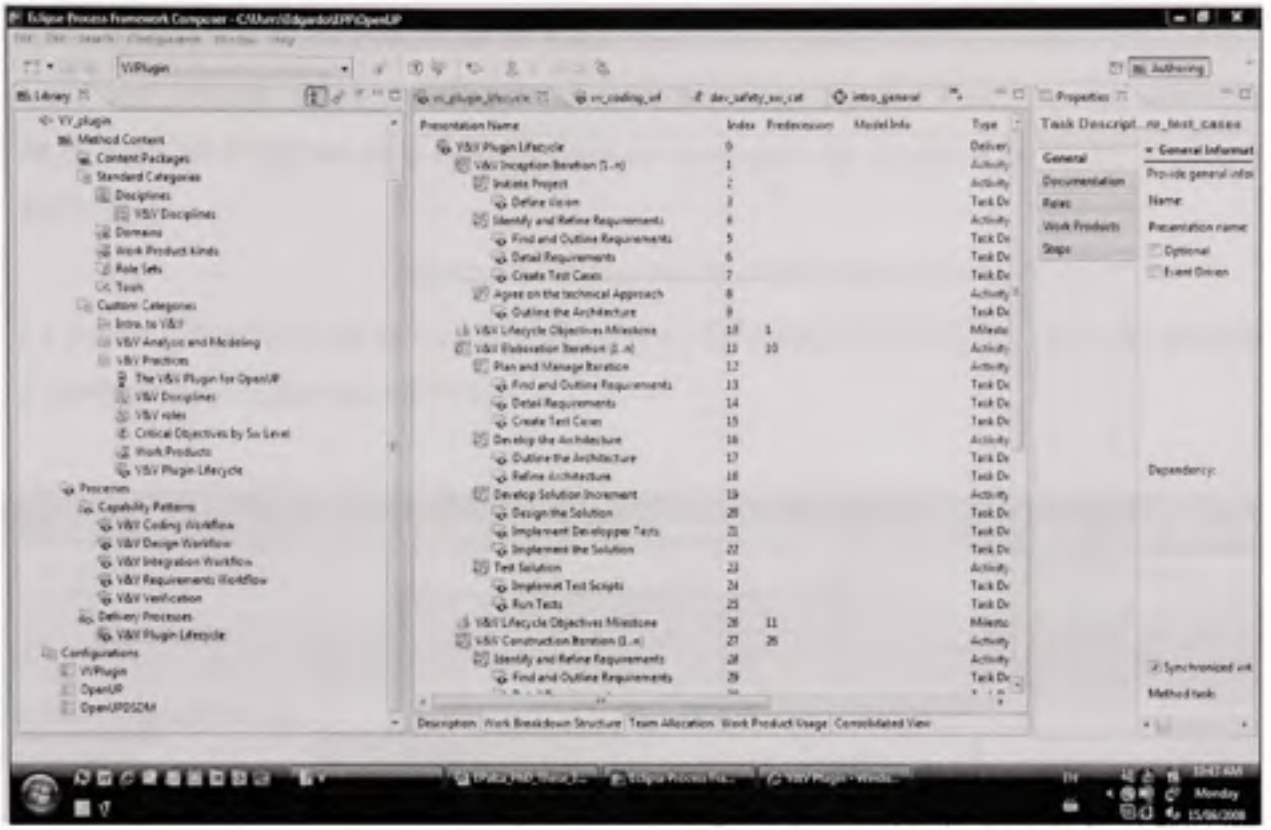

Figure 8.7 ***Construction du cycle de vie de la V&V avec l'EPFC.***

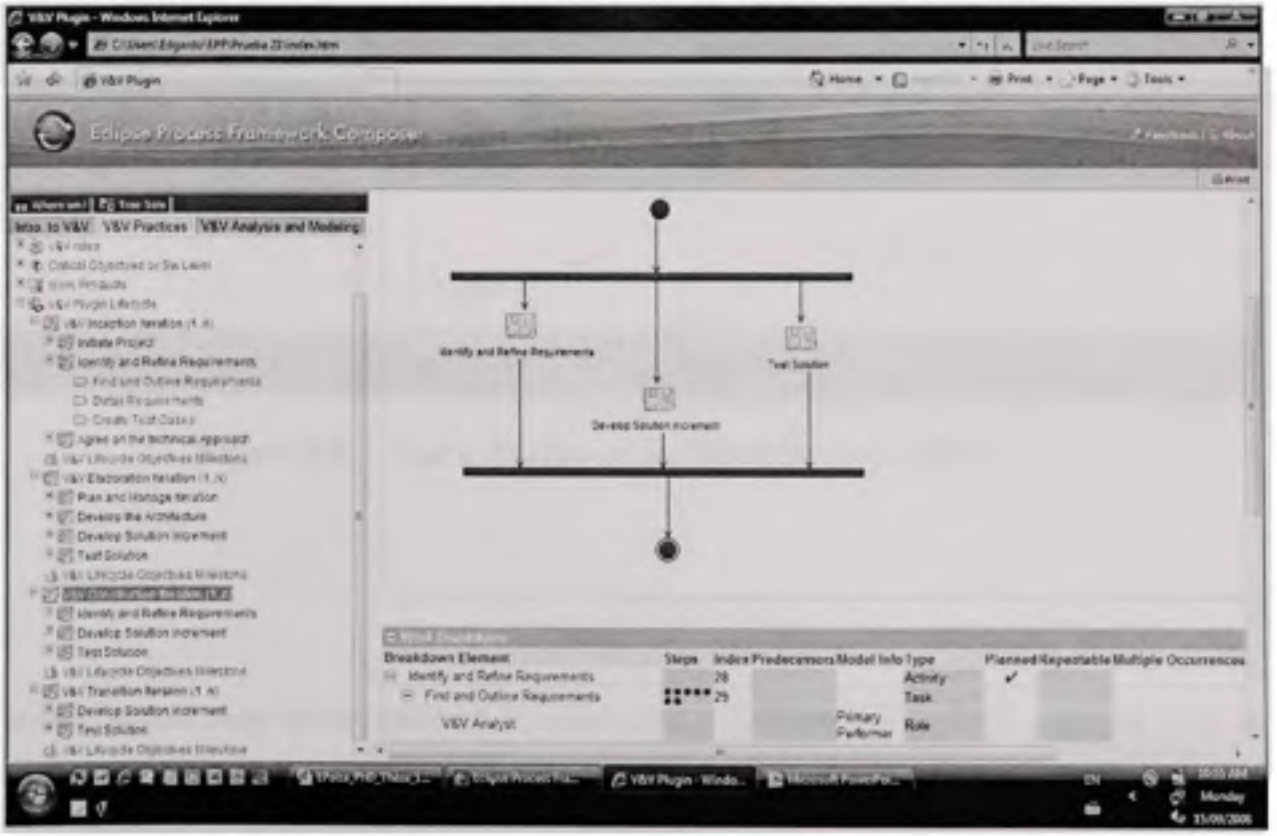

Figure 8.8 ***Le « Workflow » dans le cycle de vie du plugiciel de la V&V.***

- **Vue : analyse et modélisation de la V&V**

  Cette vue présente notre contribution de recherche par rapport à l'analyse des processus de la V&V et l'analyse et la modélisation des pratiques de la norme IEEE 1012 avec le RUP XDE.

  La figure 8.9 présente la vue de « Analyse et modélisation de la V&V» qui a été générée à partir de l'environnement EPFC.

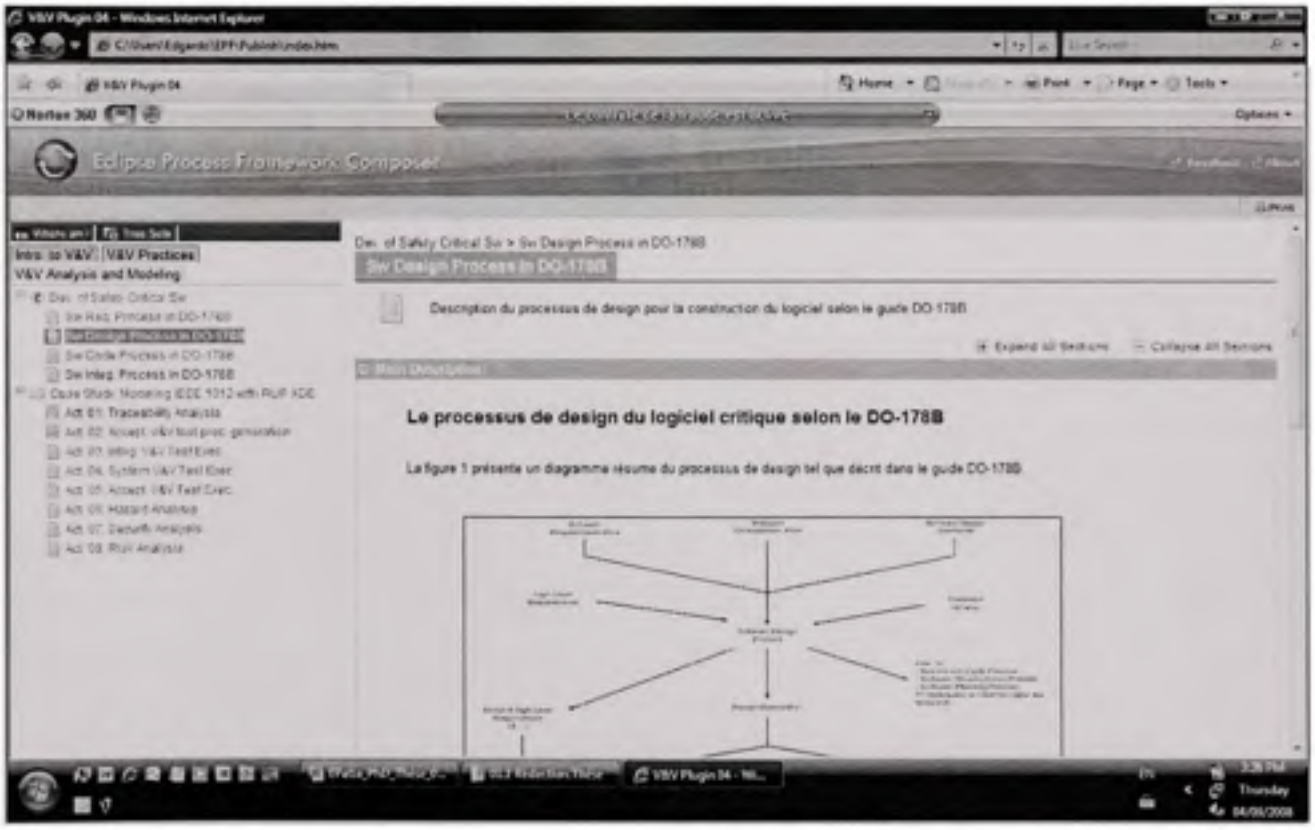

Figure 8.9 ***Vue « Analyse et modélisation de la V&V ».***

8.7 Sommaire

Dans ce chapitre nous avons présenté la construction de l'artefact logiciel plugiciel pour la définition des pratiques V&V dans les projets du logiciel critique et plus particulièrement pour la certification du logiciel avec le guide DO-178B.

Nous avons montré en détail les différentes étapes de la construction du plugiciel de la V&V ainsi que sa structure et organisation interne. Nous avons montré l'environnement EPFC ainsi que les différentes méthodes, disciplines, tâches, rôles et produits de travail que nous avons créés pour la construction du plugiciel. Nous avons montré les différentes vues lors de la génération du plugiciel et comment l'usager pourrait s'en servir.

CHAPITRE 9

CONCEPTION ET CONSTRUCTION D'UN ENTREPÔT GÉNÉRIQUE DES MESURES POUR LA V&V

9.1 Introduction

Ce chapitre présente la phase 4 de la méthodologie de recherche par rapport à proposition de la conception et la construction d'un entrepôt générique et flexible des mesures dans le cadre des activités des V&V (« L'entrepôt des mesures ») pour le développement du logiciel critique. Nous proposons un entrepôt des mesures pour soutenir les multiples concepts et les conditions de mesure dans un projet du logiciel critique.

Prendre des mesures pour l'amélioration des pratiques de V&V impose des contraintes. Souvent les organisations développent des produits logiciels dans des environnements complexes et très dynamiques. Cela produit souvent une quantité significative de données historiques des projets, stockées sans avoir la capacité de les utiliser. Dans ces conditions dynamiques et de changement constant, les organisations ont besoin d'un outil qui permet de définir des mesures avec une flexibilité de modification et tout en gardant la valeur des données historiques.

L'entrepôt des mesures pour la V&V doit permettre de spécifier et établir le suivi des mesures de base et des mesures dérivées tel qu'il est décrit dans le « Practical Software Measurement – PSM » (McGarry, 2001) et la norme ISO 15939 (ISO/IEC Std 15939, 2007), voir figure 4.3 (Sellami, 2005).

9.2 Démarche pour la conception et réalisation de l'entrepôt des mesures

Différentes activités que nous avons accomplies pour la conception et réalisation de l'entrepôt générique des mesures pourraient être groupées en deux étapes (voir figure 9.1), à savoir :

Première étape : conception et construction de l'entrepôt des mesures

Cette première étape a été développée dans le cadre d'une entente avec Ericsson Research Canada. La méthodologie de recherche proposée a été la suivante :

- prendre connaissance des exigences de l'organisation par rapport à un outil efficace et flexible pour des mesures dans les projets;
- identification du contexte des mesures et des projets à être gérés par l'entrepôt;
- étude et analyse de l'état de l'art par rapport aux modèles des mesures, des programmes de mesures (ISO/IEC Std 15939, 2007) et des données multidimensionnelles des mesures.
    - l'étude des différentes techniques pour l'exploitation des données des mesures en génie logiciel et les techniques OLAP (Online Analytical Process).
    - identification des contraintes pour la visualisation des cubes OLAP dans un environnement Web.
    - construction des spécifications fonctionnelles, non-fonctionnelles et technologiques pour l'entrepôt des mesures.
    - définition des étapes pour la conception, design, construction du prototype de l'entrepôt.
    - construction et test du prototype de l'entrepôt en collaboration avec un étudiant de maîtrise.

Deuxième étape : utilisation de l'entrepôt des mesures pour les processus et produits de la V&V

- Identification d'un cycle de vie standard pour l'implantation des pratiques de la V&V dans les logiciels critiques :

- Utilisation de la norme IEEE 1012 (entre autres elle propose un cycle de vie pour la V&V);
- Identification des étapes du cycle du développement d'un projet du logiciel critique à partir la documentation fournie par le partenaire industriel Verocel Inc. (Verocel, 2005);
- Identification des étapes du cycle du développement d'un projet du logiciel critique avec la documentation fournie par le partenaire industriel CMC Electronics (CMC Electronics, 2007);

- Instanciation de l'entrepôt développé dans l'étape précédente dans le contexte du logiciel critique :
  - identification des mesures pour chaque étape du développement du logiciel critique.
  - instancier les mesures proposées dans l'entrepôt des mesures de V&V.

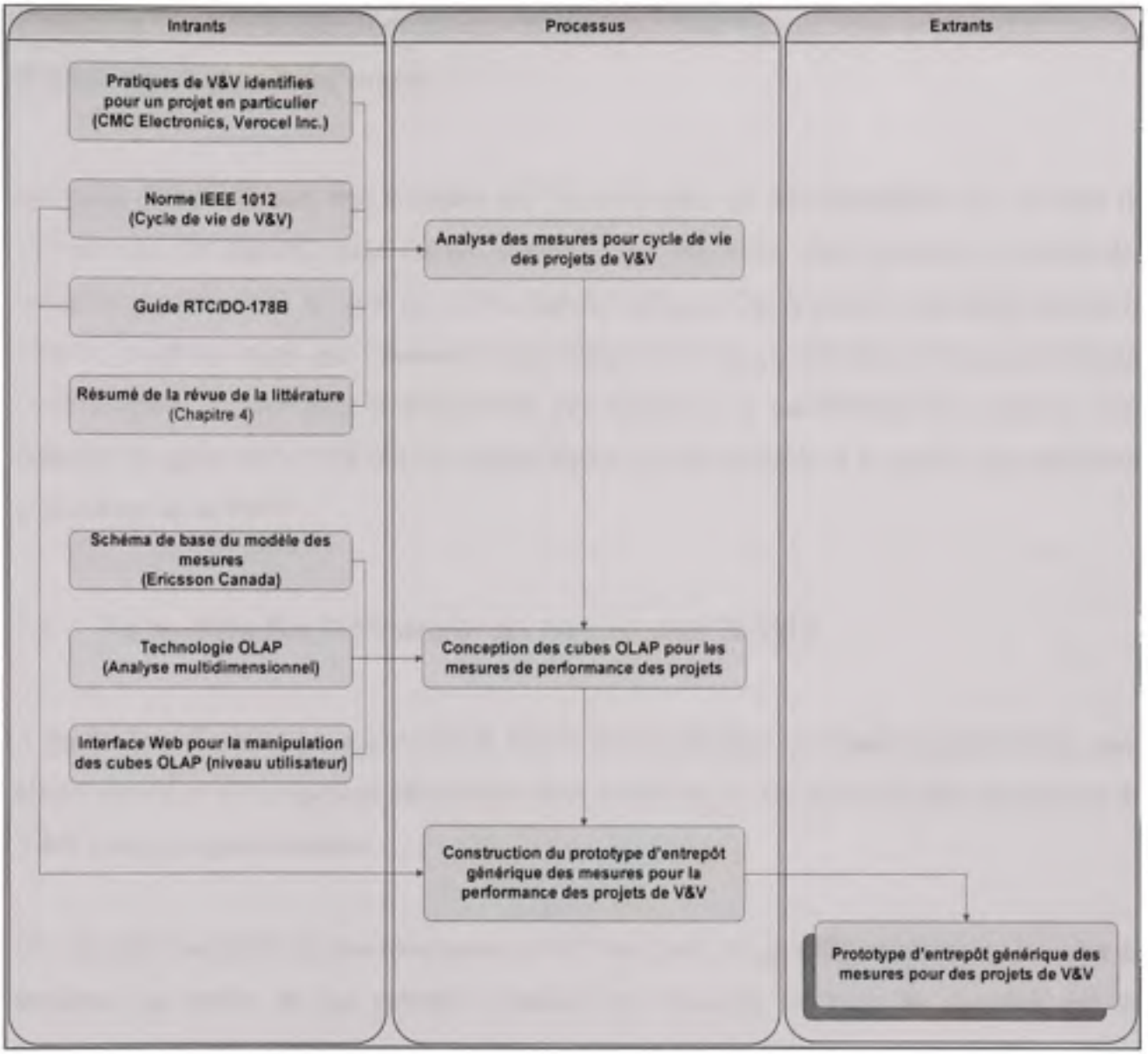

Figure 9.1 ***Démarche pour la conception et réalisation de l'entrepôt des mesures.***

9.3 Besoin d'un entrepôt des mesures de la V&V

L'introduction d'un entrepôt des mesures, dans le cadre de notre proposition, répond au besoin constaté dans l'industrie d'améliorer la qualité dans les pratiques de construction du logiciel, d'améliorer la productivité et la performance des équipes des projets. L'entrepôt des mesures facilite la tâche de prendre de mesures, faire sortir des indicateurs et prendre des décisions dans les étapes primaires du projet. D'ailleurs l'entrepôt devrait également servir

comme une base de données (au niveau des pratiques pour les processus et produits) pour les projets à développer dans l'avenir.

Le guide DO-178B met une emphase sur les processus du développement par le biais de l'évaluation des objectifs pour chaque processus en particulier, ainsi que dans la qualité des livrables produits dans le cycle de vie du logiciel critique. Ces livrables sont exprimés par la liste de livrables exigés par l'autorité compétente lors de la certification du logiciel critique (voir chapitre 2 pour plus d'information par rapport à a certification du logiciel). Les objectifs du guide DO-178B ont un rapport direct avec le contrôle et la qualité des processus et livrables de la V&V.

9.4 La modélisation de l'entrepôt des mesures pour la V&V

À partir du référentiel des processus de V&V développé dans les chapitres précédents, nous allons identifier les exigences nécessaires pour la définition des mesures dans les projets de V&V pour le logiciel critique.

De son côté l'entrepôt des mesures pour la V&V est basé sur un méta-modèle des données de mesures. Le noyau de cet entrepôt contient une structure de base de données qui ne présuppose aucune mesure ou rapport particulier entre elles ; les mesures sont elles-mêmes traitées comme des données. Ces données sont désignées sous le nom de méta-données (i.e. des données qui représentent des données de mesure des produits et des processus de V&V). Pour rencontrer les contraintes d'un environnement dynamique des mesures dans les projets et en particulier les projets du logiciel critique, l'entrepôt des mesures pour la V&V est composé d'une structure générique des données, implémentée par l'entrepôt, avec un niveau élevé de flexibilité. Cette exigence demande alors que les définitions des mesures de V&V et leurs associations soient sauvegardées dans l'entrepôt comme une entité appelé méta-données ou « metadata ». Le principe de méta-données indique un niveau d'abstraction des mesures de V&V plutôt que les mesures spécifiques. L'entité méta-données peut alors fournir la

flexibilité du besoin de différents types de mesures exigées pour avoir une bonne compréhension et quantification du projet.

La caractéristique de l'entrepôt à supporter des mesures de base et des mesures dérivées permet de bâtir différents types de mesures reliées soit aux processus ou produits selon les exigences de la norme IEEE 1012 et en cohérence avec le guide DO-178B. La collection et le stockage des données de mesures sont implémentés à travers un système de base de données reliées aux cubes OLAP. Le système d'analyse de données et des rapports est basé sur les requêtes aux cubes OLAP (Online Analytical Process).

L'entrepôt des mesures pour la V&V est basé sur une structure particulière qui associe les exigences d'information dans les projets avec les entités et les attributs pertinents. Dans le cas de la V&V, ces entités sont liées aux processus et aux produits de V&V ainsi que des plans et des ressources de V&V associés aux projets. Le modèle des mesures qui supporte l'entrepôt décrit comment les attributs de la V&V sont associés et comment les mesures sont converties en indicateurs qui vont faciliter la prise de décision dans les projets.

La norme ISO 15939 explique que le processus de mesures consiste en 4 activités principales, à savoir :
1.- l'établissement des engagements,
2.- la planification,
3.- l'exécution et
4.- l'évaluation des résultats de mesures.

L'entrepôt des mesures de V&V que nous proposons est aligné avec ces 4 processus décrits dans la norme ISO 15939.

La figure 9.2 montre le support de l'entrepôt de mesures à la norme ISO 15939, à savoir :
1.- L'entrepôt des mesures permet d'assigner des utilisateurs (rôles et responsabilités) aux processus des mesures.

2.- L'entrepôt des mesures permet la planification et la définition des mesures pendant tout le cycle de vie du développement logiciel critique.

3.- L'entrepôt des mesures permet l'exécution des mesures basées sur des données extraites des processus et/ou produits de la V&V.

4.- L'entrepôt des mesures permet l'évaluation des résultats des mesures par le biais de la manipulation des cubes OLAP et la création des graphiques à partir de ces cubes.

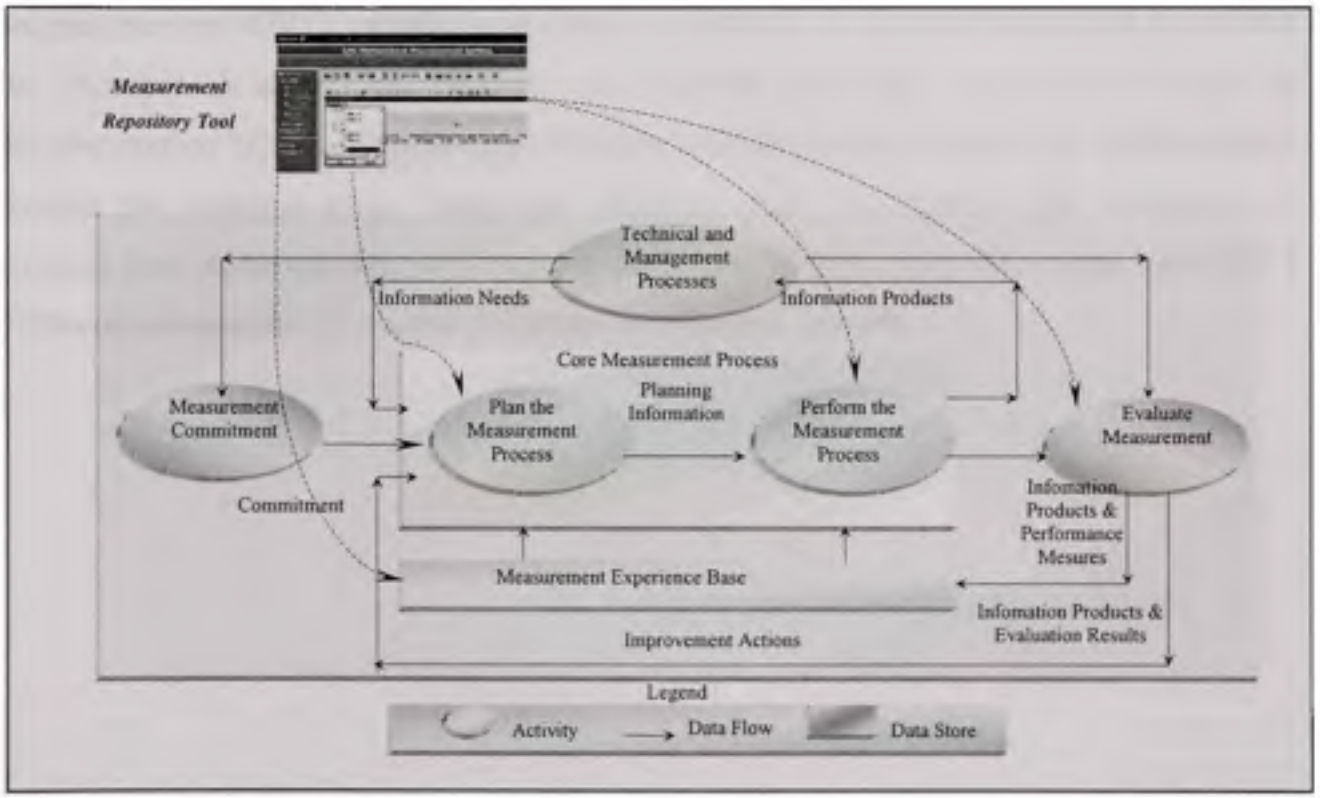

Figure 9.2 ***La norme ISO 15939 et l'entrepôt des mesures pour la V&V.***

9.5 Construction de l'entrepôt des mesures pour la V&V

La figure 9.3 présente le modèle qui a été conçu pour combler les contraintes par rapport à la capacité d'être générique de l'entrepôt. Il contient l'ensemble des classes qui sont énumérées et expliquées dans le tableau 9.1. Également dans le tableau 9.1 nous avons inclus des références pour chaque classe mentionnée dans la figure 9.4.

L'ensemble des relations entre les différentes classes est sauvegardé comme un autre type de classe dans l'entrepôt. Ceci permet de supporter une vue hiérarchique et multidimensionnelle des données. Cette caractéristique essentielle de l'entrepôt est implémentée par les services OLAP tels que : navigation /l'agrégat pour données de mesures associées avec une classe dans un autre niveau (« upper /lower »).

Les utilisateurs ont la capacité de manipuler les cubes OLAP à travers une interface Web programmée en ASP. L'interface comporte un ensemble de fonctionnalités qui permettent par exemple de lancer des requêtes aux données des cubes sans avoir besoin de programmer en SQL. L'interface permet également diverses fonctionnalités additionnelles comme par exemple avoir l'accès aux diverses cubes, sélectionner des indicateurs et changer leur représentation graphique, consulter différentes vues des cubes, travailler à différents niveaux de granularité des données, faire des rapports.

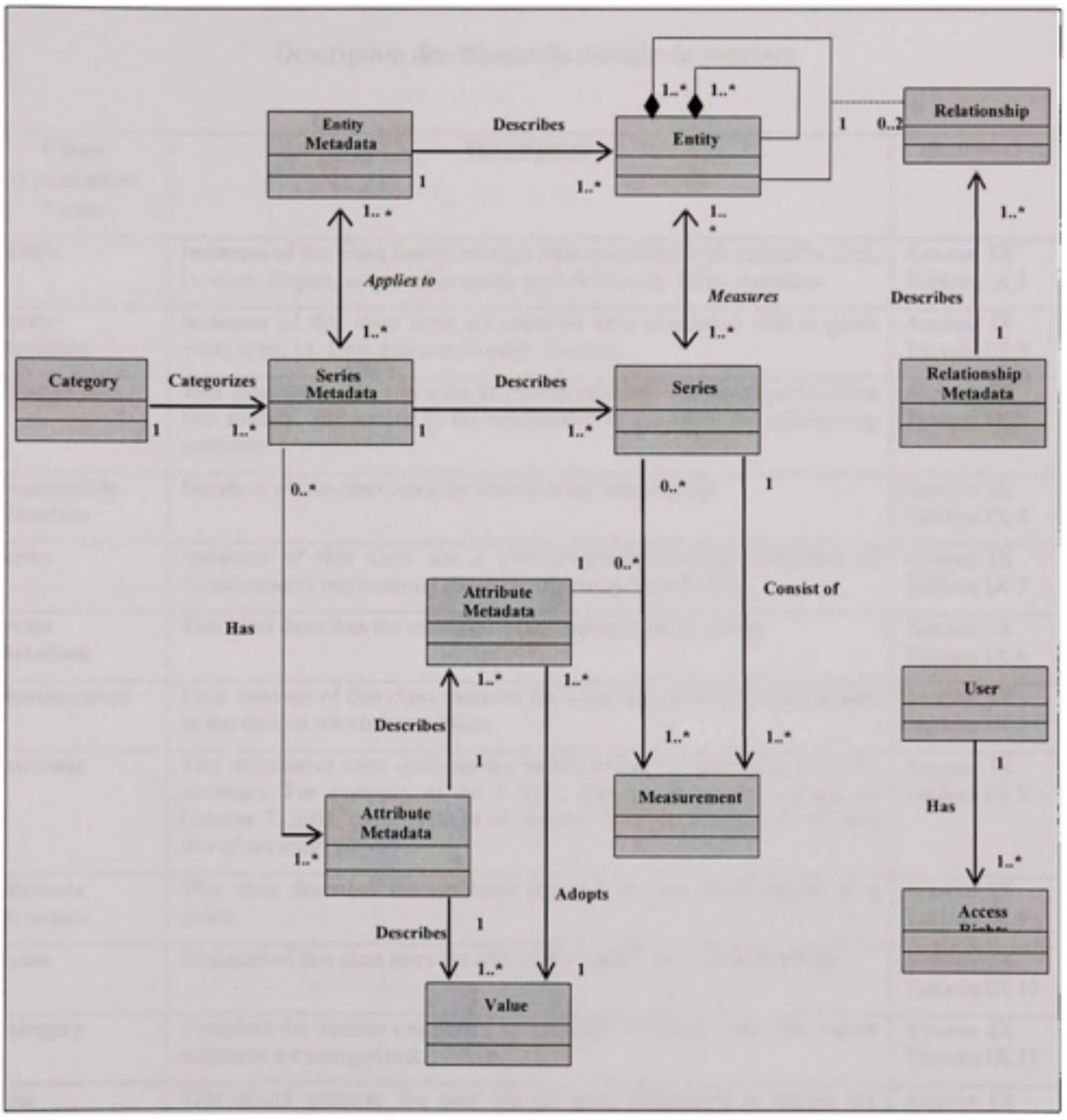

Figure 9.3 ***Diagramme de classes qui supporte le modèle générique des mesures.***

Tableau 9.1

Description des classes du modèle de mesures

Class/ *Association* Name	Description	References
Entity	Instances of this class Entity store all data associated with a specific Unit, Product, Project, or any other entity type defined by entity metadata.	Annexe IX Tableau IX.1
Entity Metadata	Instances of this class store all common data associated with a given entity type, i.e. Unit, Product, Project, Version.	Annexe IX Tableau IX.2
Relationship	This associative class is used to model arbitrary relationships between two entities. The nature of the relationship is given by the relationship metadata.	Annexe IX Tableau IX.3
Relationship Metadata	Instances of this class store the nature of the relationship.	Annexe IX Tableau IX.4
Series	Instances of this class are a chronologically ordered collection of measurements representing the value of a measure over time.	Annexe IX Tableau IX.5
Series Metadata	This class describes the measures being captured by the series.	Annexe IX Tableau IX.6
Measurement	Each instance of this class captures the value of a measurement, as well as the date on which it was taken.	Annexe IX Tableau IX.7
Attribute	This associative class qualifies the measurements according to different attributes. For example, of the 5 TR's (Trouble Reports) recorded on October 7, 2002, three could be of severity "A", one of severity "B" and one of severity "C".	Annexe IX Tableau IX.8
Attribute Metadata	This class describes the attributes that classify the measurement in a series.	Annexe IX Tableau IX.9
Value	Instances of this class store the admissible values for a given attribute.	Annexe IX Tableau IX.10
Category	Describes the various categories, i.e. Quality, Progress, Cost, into which measures are categorized.	Annexe IX Tableau IX.11
User	This object captures the user ids of those authorized to access the repository. Access to this table is restricted to the database administrator.	Annexe IX Tableau IX.12
Access Rights	Indicates the specific object and relationship instances to which a given user has access and what he or she can do with them, i.e. Create, Change, Delete. Access to this table is restricted to the database administrator.	Annexe IX Tableau IX.13
Applies To	This n:n relationship defines the applicable set of measures for each object type.	Annexe IX Tableau IX.14
Measures	This n:n relationship links a specific series to the objects or objects being measured by it.	Annexe IX Tableau IX.15

9.6 Architecture de l'entrepôt générique des mesures

Cette section présente l'architecture générale choisie pour le déploiement des fonctionnalités de l'entrepôt générique des mesures. L'entrepôt est supporté par un moteur analytique OLAP qui fournit les possibilités fonctionnelles suivantes:

- des indicateurs de gestion;

un moteur analytique navigation/l'agrégat des données;

- une interface d'administration;
- un moteur OLAP pour calculer les mesures dérivées à partir des mesures de base, ainsi que l'analyse multidimensionnelle.

La figure 9.4 illustre les caractéristiques de l'architecture de l'entrepôt des mesures que nous avons mentionnées dans le paragraphe précédent.

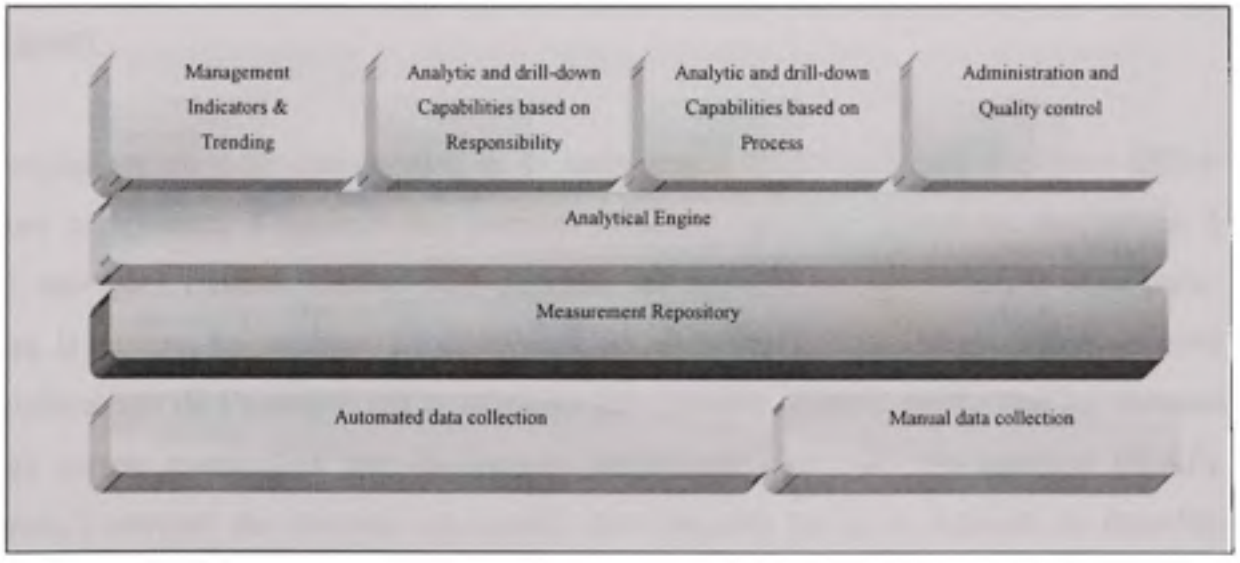

Figure 9.4 ***Architecture de l'entrepôt des mesures pour la V&V.***

9.7 La technologie OLAP et le modèle générique des mesures

La technologie OLAP joue un rôle important dans l'implémentation de l'entrepôt des mesures pour la V&V. La technologie OLAP est utilisée pour rassembler des données des sources multiples dans les projets et, en général, dans l'organisation. Les données sont stockées sous une forme qui facilite l'analyse et la prise de décisions (Ralph Kimball, 2002).

La technologie OLAP est basée sur le principe des cubes d'information avec multiples vues. Les services offerts par OLAP permettent de créer, lancer des requêtes et maintenir les cubes OLAP qui sont des vues matérialisées d'information (i.e., une manière de pré-calcul des sommaires des données de sorte que des demandes puissent être répondues rapidement. Les services OLAP (ex. l'agrégat des données, etc.) fournissent aux utilisateurs la possibilité d'analyser des données à différents niveaux de granularité. Afin d'incorporer la caractéristique multidimensionnelle, l'approche de pivotement de cubes en OLAP a été choisie pour l'affichage et l'ordre dynamique des dimensions multiples des données (Olap Train, 2000).

Des possibilités multidimensionnelles de la technologie OLAP sont utilisées pour définir plusieurs composants d'entrepôt des mesures (classes dans le contexte du diagramme à objets), tels que : l'entité, l'entité méta-données, les agrégations, les séries, la série méta-données, la mesure, les attributs, les catégories, les associations. Nous avons déjà mentionné que l'architecture de l'entrepôt fait le stockage des mesures de base, tandis que les mesures dérivées seront manipulées par "le moteur analytique" (i.e., par les services OLAP). D'ailleurs, l'entrepôt des mesures est capable de manipuler les types suivants de données: nominal, ordinal, intervalle et ratio en concordance avec ISO 15939 (ISO/IEC Std 15939, 2007).

9.8 Le cycle de vie de V&V et l'entrepôt des mesures

Il est important de mentionner que le guide DO-178B n'exige pas un cycle de vie en particulier pour les projets de développement du logiciel critique. D'un autre côté, la gestion de la V&V dans les projets est définie en fonction du cycle de vie de V&V choisi pour ce projet. Alors pour démontrer la capacité de l'entrepôt des mesures à remplir les exigences de la gestion de la V&V dans projets, nous allons montrer comment l'entrepôt des mesures remplit les exigences d'un processus de gestion de la V&V en particulier. Le processus de gestion de la V&V que nous avons choisi est décrit dans la norme IEEE 1012 section 5.1.1.

Afin d'incorporer l'entrepôt des mesures dans le contexte de la V&V nous avons choisi de travailler avec le cycle de vie de V&V proposé par la norme IEEE 1012 (IEEE Std 1012, 2004). Ce cycle de vie facilite la détermination des artefacts impliqués dans chaque étape de la vérification et de la validation du logiciel.

9.8.1 La norme IEEE 1012 et l'entrepôt des mesures

La norme IEEE 1012 est recommandé à la NASA pour utilisation dans des projets dépendants du logiciel. Le manuel du directeur de programme de la « NASA IV&V Facility » (NASA Software IV&V Facility, 2000) mentionne l'utilité de l'IEEE 1012 (IEEE Std 1012, 2004) et d'IEEE 1059 (IEEE Std 1059, 1993) pour la planification et l'exécution des activités de V&V dans leurs projets de logiciel critique. Les processus de logiciel décrits dans la norme IEEE 1012 incluent: gestion, acquisition, approvisionnement, développement, opération, et entretien.

L'entrepôt des mesures fournit l'appui à la gestion des activités de V&V décrites dans IEEE 1012. Étant donné le structure générique du modèle des données des mesures, l'entrepôt serait capable de supporter les mesures recommandées par la norme IEEE 1012 dans le contexte du cycle de vie d'un projet de V&V, à savoir:

- le suivi de l'exécution du plan de V&V du logiciel;

- l'analyse des problèmes découverts pendant l'exécution du plan de V&V;
- le rapport des progrès du processus de V&V;
- l'assurance que les produits répondent à des exigences;
- l'évaluation des résultats;
- la détermination si un tâche est complétée et l'évaluation des résultats pour assurer leur complétude.

La figure 9.5 montre comment l'entrepôt des mesures de V&V interface avec les processus, les activités et les tâches établis dans IEEE 1012. La gestion des activités de V&V surveille et évalue tous les livrables des processus de V&V définis dans le plan de V&V. Avec l'aide de l'entrepôt des mesures de V&V nous pouvons identifier des tendances des données et des risques potentiels dans la gestion de V&V.

Dans les paragraphes suivants nous présentons certaines caractéristiques de l'entrepôt des mesures de V&V qui sont directement liées à l'exécution d'une gestion efficace de V&V dans le logiciel critique.

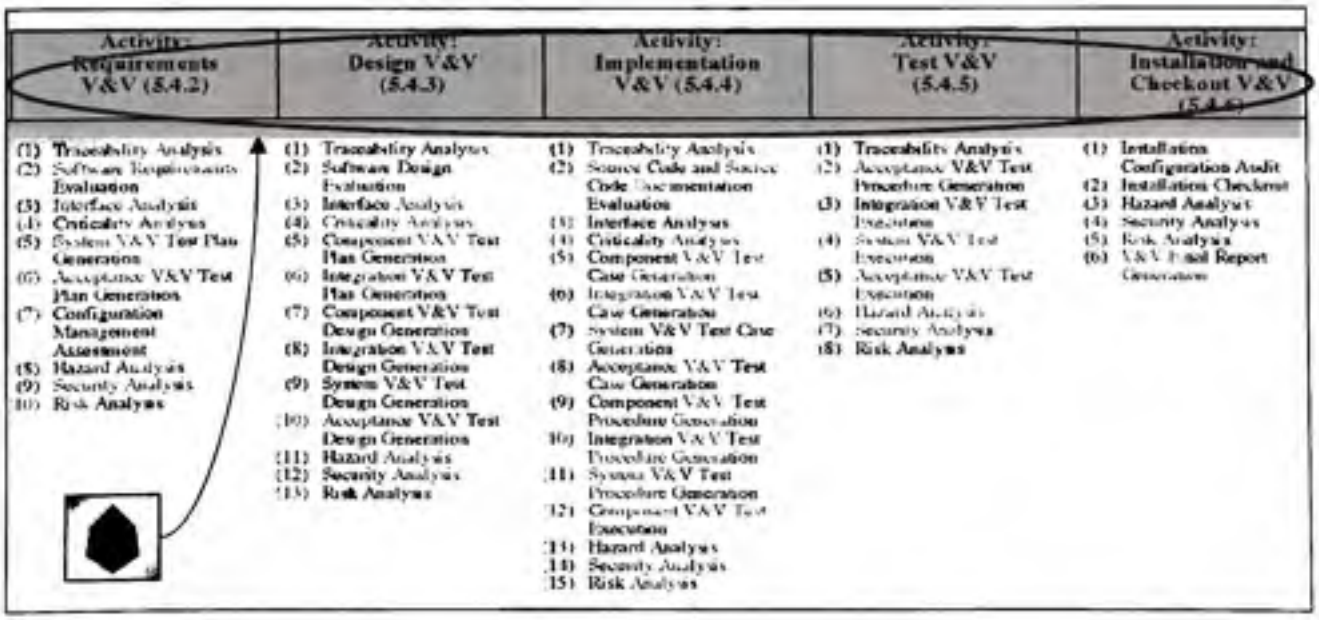

Figure 9.5 L'entrepôt des mesures et le cycle de vie de la V&V selon IEEE 1012.

L'entrepôt des mesures de V&V fournit une compréhension quantitative de l'efficacité de V&V en termes de qualité, fiabilité et la maintenance des projets du logiciel critique. L'entrepôt des mesures de V&V facilite l'évaluation de la qualité de l'exécution des processus, les activités et les tâches par l'établissement des mesures appropriées.

9.9 Planification de la V&V

L'entrepôt des mesures de V&V facilite le suivi des étapes importantes identifiées dans le plan de V&V selon IEEE 1012 (5.1.1 tâche 1).

La planification des mesures des processus et des produits de V&V que l'entrepôt pourrait gérer dans un projet serait par exemple :

- l'efficacité des activités de processus de V&V en termes de temps, de ressources allouées;
- des coûts dépensés dans chaque phase du cycle de vie;
- le temps et le coût pour produire le plan de V&V;
- la complétude des exigences de haut niveau (par exemple mesurée en termes de nombre de changements apportés au document des exigences);
- le pourcentage des défauts éliminés avec les activités de vérification par produit;
- performance des activités de vérification (revues et analyses);
- le pourcentage de temps de reprise, temps moyen entre les défaillances;
- le nombre et la sévérité des défauts par produit livré;
- le nombre des exigences non-traçables;
- le nombre et le type des erreurs détectées au niveau des environnements des tests (unitaires, couverture, robustesse, d'intégration, etc.), etc.

9.9.1 Évaluation des mesures de V&V pour le changement

L'entrepôt des mesures de V&V permuterait la collection et le stockage des données de mesure pour l'évaluation des changements dans le projet (IEEE 1012, 5.1.1 tâche 2).

L'entrepôt des mesures de V&V enregistre ces données de mesures dans une hiérarchie flexible et adaptable, cette caractéristique est offerte par le biais de la technologie OLAP et du modèle générique des données des mesures. Cette hiérarchie est composée des niveaux des associations pour faciliter l'évaluation des changements du logiciel pour les tâches de V&V (par exemple, des corrections des anomalies et des changements de conditions).

Il est important de noter que l'entrepôt des mesures de V&V a été construit selon un ensemble de mesures relatives qui sont généralement applicables dans plusieurs circonstances, indépendamment des besoins spécifiques de l'information de n'importe quelle situation particulière. L'entrepôt des mesures de V&V rassemble des données de mesure de plusieurs tâches de V&V et permet d'établir une comparaison entre elles. L'entrepôt des mesures de V&V offre la possibilité d'établir des évaluations basées sur des données historiques de la mesure V&V, cette caractéristique est offerte par le biais des requêtes aux cubes OLAP, aussi bien que fournir une compréhension de la nature et de l'ampleur de la variation éprouvées dans l'exécution de processus.

9.9.2 La révision et l'amélioration des pratiques de V&V dans les projets

L'entrepôt des mesures de V&V peut faciliter la consolidation de l'effort des activités et tâches de V&V afin d'avoir une idée où réorienter l'effort dans le projet (IEEE 1012, 5.1.1 tâches 3 et 6). L'entrepôt des mesures peut contribuer pour contenir des mesures pour déterminer la valeur de chaque processus d'amélioration en ce qui concerne les objectifs de la qualité et les processus de performance de l'organisation. Les exemples de la mesure incluent : le ratio du nombre des modules du logiciel vérifiés et validés par rapport au total du nombre de modules, etc.

L'entrepôt des mesures de V&V facilite la mesure du coût réel de la V&V par rapport au coût prévu de V&V, l'effort en termes de tâches et le temps pour déployer la V&V dans chaque processus et produit, ces caractéristiques ont été testés dans le projets avec Ericsson Research Canada.

9.9.3 Les indicateurs de la mesure pour la gestion et le support de la V&V

L'entrepôt des mesures de V&V incorpore la possibilité d'établir des indicateurs basés sur des données hiérarchiques de mesure stockées dans le système de base de données du modèle. L'entrepôt des mesures de V&V est conçu pour accepter une définition adaptée aux besoins du projet : par exemple, la définition des alertes d'un déclenchement quand une valeur maximum est atteinte.

L'entrepôt des mesures de V&V peut déterminer quantitativement le statut des processus : il peut surveiller et détecter des changements dans la performance des processus et alors les décideurs peuvent implémenter les actions correctives selon les besoins. L'entrepôt des mesures de V&V offre la possibilité de vérifier la livraison opportune selon le plan, de tous les produits de logiciel nécessaires (IEEE 1012, 5.1.1 tâche 4). L'entrepôt des mesures de V&V offre l'option d'établir des mesures de processus (par exemple, efforts, durée de cycle, et efficacité de déplacement de défaut) et des mesures du produit (par exemple, fiabilité, densité de défauts).

9.9.4 Interface avec des processus de l'organisation

Étant donné la flexibilité conféré par les outils Microsoft (MS-Analysis Services) que nous avons utilisé pour la construction de l'entrepôt, il existe la possibilité d'exporter les données de V&V dans différents formats pour faciliter l'échange avec d'autres processus implémentés dans l'organisation (IEEE 1012, 5.1.1 tâche 5).

9.9.5 Construction de l'entrepôt des mesures

L'entrepôt des mesures de V&V se compose d'une collection de cubes multidimensionnels des données (cubes OLAP). Ces cubes des données contiennent les données d'agrégation sur lesquelles l'analyse multidimensionnelle des données des mesures est basée. Les agrégations

sont les sommaires pré-calculés des données de mesure qui améliorent l'efficacité et le temps de réponse des requêtes d'utilisateur. L'entrepôt des mesures de V&V est basé sur le concept d'un schéma de type étoile. Ce schéma de type étoile représente un modèle multidimensionnel, qui se compose d'une table centrale de faits et plusieurs tables de dimensions. Les tables de faits contiennent les enregistrements qui représentent des mesures (les faits) à être analysées. Chaque table de faits fait référence à une ou plusieurs tables de dimension et chaque table de dimension représente une perspective d'intérêt à mesurer (comme par exemple : des mesures à partir de l'unité, le projet, le produit, etc.). L'annexe VII section « OLAP cube definition using the MS-Analysis Services» présente les détails du design et la construction des cubes OLAP que nous avons accompli pour l'entrepôt de mesures.

9.9.6 La gestion des mesures et les projections

Nous avons démontré lors du test du prototype de l'entrepôt chez Ericsson Research Canada, comment la technologie OLAP fournit la représentation graphique des mesures dans l'entrepôt des mesures de V&V. Cette fonctionnalité est essentielle pour analyser et déterminer le pourquoi de certains comportements qui se produisent tout au long des projets.

L'annexe VII section « OLAP cube visualization using the MS-Analysis Services» présente les détails de la visualisation des cubes OLAP que nous avons développés aux travers d'une interface Web développée en ASP Net. L'interface exécute les cubes OLAP que nous avons définis à travers du langage MDA (Multidimensional Analysis) de Microsoft.

L'annexe X présente comment nous avons construit les cubes OLAP dans l'entrepôt des données des mesures, La section « OLAP Fact table definition », présente comme exemple des scripts écrits en MDA pour la définition pour les mesures « Earned Value » et « Performance Index – PI». Le « Earned Value Measure - EV» (valeur acquise) est une mesure qui représente quelle quantité du projet (ex. du processus du développement) a été

développée réellement, par opposition à combien d'heures ou combien de ressources ont été dépensées. La valeur acquise est une mesure des travaux terminés exprimés en pourcentage du budget prévu pour les travaux terminés. Les mesures type « Performance Index – PI» sont utilisées pour le suivi, la planification, l'estimation, le contrôle de coûts et du temps des projets. Un exemple de ces mesures est : « Cost performance index (CPI) », « schedule performance index (SPI) ».

Il faut mentionner que le prototype de l'entrepôt des mesures pour la V&V a été testé chez Ericsson Research Canada avec succès avec les mesures mentionnées dans le paragraphe précédent. La construction et l'exécution du prototype avec les mesures « EV » et « PI » est montré dans l'annexe X.

L'entrepôt des mesures pour la V&V a fait l'objet d'articles de recherche au niveau international : (Palza, Abran et Fuhrman, 2003), (Abran et Palza, 2003), (Palza, Abran et Fuhrman, 2004), (Palza, 2005).

9.10 Technologie utilisé pour la construction de l'entrepôt des mesures de V&V

Nous avons utilisé les produits suivants de Microsoft pour la construction de l'entrepôt des mesures de V&V :

- MS Windows 2000 Server ;
- MS SQL 2000 Server ;
- MS Analysis Services Enterprise Edition ;
- MS Internet Information Server ;
- ASP technology and Pivot Table Services (PTS).

9.11 Sommaire

Dans ce chapitre nous avons montré notre proposition de l'entrepôt des mesures pour les pratiques de V&V. Nous avons montré la structure de l'entrepôt qui est basé sur un modèle générique des mesures construit sur un diagramme de classes, ainsi que le support technologique que nous avons utilisé pour implémenter ce modèle.

Nous avons expliqué pourquoi la technologie OLAP a été choisie pour implanter le modèle. Nous avons fait un tour des différentes options techniques pour implémenter l'entrepôt des mesures. Également nous avons indiqué, avec une analyse du cycle de vie de V&V proposé par la norme IEEE 1012, comment l'entrepôt des mesures pourrait combler et faciliter la gestion des projets dans le cadre de la V&V. Les différentes phases et mesures pour la gestion des projets pourront être gérées avec l'entrepôt des mesures. Nous avons montré que l'ensemble des caractéristiques accordés par la technologie OLAP faciliterait la collecte, l'exécution, l'évaluation, le stockage des mesures en concordance avec la norme ISO 15939 et le PSM (Practical Software Measurement).

Comme nous avons mentionné nous avons publié plusieurs articles de recherche par rapport à l'entrepôt des mesures en général (par exemple dans le cadre du CMMI (Palza, Abran et Fuhrman, 2003) et l'amélioration des processus) et une en particulier par rapport à l'entrepôt des mesures et la gestion des pratiques de V&V dans les projets (Palza, Abran et Fuhrman, 2004).

CHAPITRE 10

ÉVALUATION DES RÉSULTATS DE RECHERCHE

Pour valider les résultats finaux de notre proposition de recherche, l'approche idéale aurait été d'implanter notre proposition dans une des organisations partenaires, mais cela n'a pas été possible dans la pratique. Nous avons fait des évaluations partielles de chaque livrable important de notre contribution, à savoir : le modèle des pratiques de V&V et l'entrepôt des mesures de V&V.

10.1 Évaluation du référentiel des processus de V&V

Comme nous l'avons mentionné le modèle de processus de V&V est exprimé par un prototype. Pour l'évaluation de ce prototype nous avons utilisé la méthode testé les capacités du prototype selon les besoins des organisations exprimés par ces livrables de travail et des contrats qui ont été accordés dans le cadre de notre travail de recherche. Voici une description des activités réalisées avec nos partenaires au niveau de l'encadrement, la vérification et le support des livrables du modèle des processus de V&V (voir figure 10.1).

CMC Electronics[3] : l'entente consistait à avoir accès aux divers livrables des projets liées à la certification DO-178B du logiciel critique. Les livrables que nous avons eu l'opportunité d'analyser et d'utiliser dans le cadre de notre contribution ont été les suivants:

- Software Verification Plan (SVP) ;
- Software Configuration Management Plan (SCMP) ;
- Software Development Plan (SDP) ;

[3] http://www.cmcelectronics.ca

- Software Quality Assurance Plan (SQAP).

Ces livrables nous ont donné la possibilité de structurer, garder une cohérence et valider le référentiel de V&V avec les contraintes de l'industrie du logiciel critique.

Verocel Inc.[4]: cette entreprise américaine offre des services de consultation dans la certification du logiciel critique avec le guide DO-178B. Des membres de cette entreprise font partie du groupe du développement du guide DO-178B. L'entente que nous avons signée nous a donné accès à la documentation et au code source du logiciel critique d'un projet qui devait être conforme au guide DO-178B. Les livrables auxquels nous avons eu accès nous ont permis de valider le référentiel de V&V depuis une perspective de la construction détaillée (codification) du logiciel critique. Les livrables auxquels nous avons eu accès sont les suivants :

- code source des diverses modules du projet du logiciel critique;
- documentation associée aux composants du logiciel critique.

NASA IV&V Facility[5]: nous avons eu un contrat pour déployer la recherche dans le contexte de la V&V. Le but de ce travail a été d'identifier des artefacts dans le CMMI (Capability Maturity Model Integrated) (SEI, 2006) qui peuvent remplir les exigences de la V&V dans le contexte des projets du logiciel critique développé par NASA. Les exigences de V&V à combler pour le CMMI ont été décrites principalement dans les normes IEEE 1012 et IEEE 1059. Les artefacts de V&V ont été identifiés par niveau de maturité des organisations en concordance avec le CMMI. La recherche développée dans le cadre de cette entente nous a permis d'identifier des artefacts et des pratiques pertinentes pour les incorporer dans le

[4] http://www.verocel.com

[5] www.ivv.nasa.gov

référentiel de V&V proposé dans cette thèse. Nous avons publié des articles de recherche avec nos résultats (voir Annexe I).

Autres références de l'industrie et la recherche : nous avons également identifié des références au niveau de la recherche ainsi que des publications des expériences au niveau industriel afin de bâtir un ensemble des caractéristiques pour le prototype des processus de V&V proposé dans le cadre de cette thèse (voir chapitre 8).

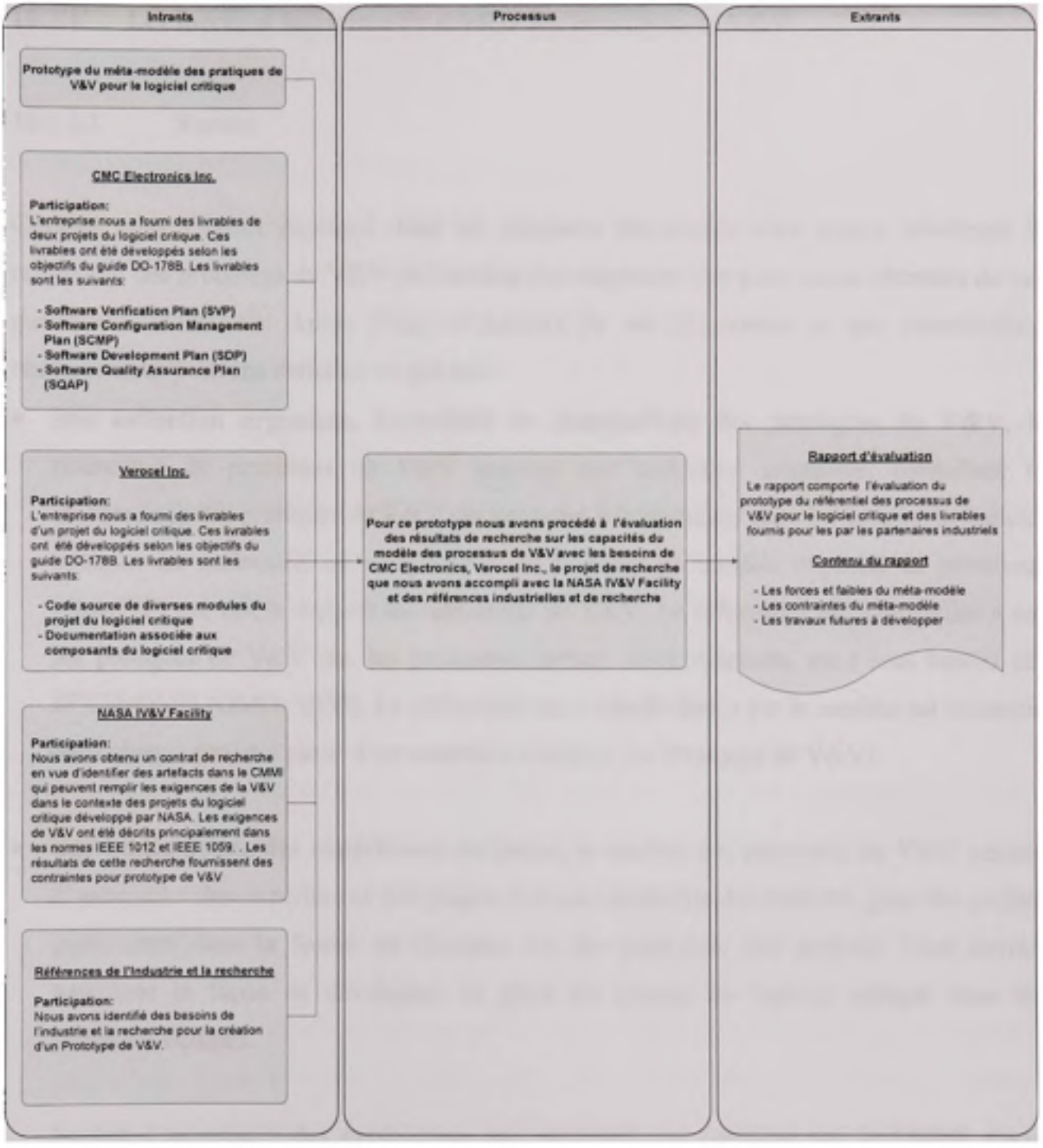

Figure 10.1 ***Évaluation des résultats du modèle des processus de V&V.***

10.1.1 Les forces et faiblesses du modèle des processus de V&V

10.1.1.1 Forces

Comme nous l'avons expliqué dans les chapitres précédents nous avons développé le référentiel des processus de V&V en fonction des exigences que nous avons obtenues de nos partenaires industriels. Après d'une évaluation de ces documents et nos contributions réalisées nous pouvons conclure ce qui suit :

- **une collection organisée, formalisée et standardisée des pratiques de V&V,** le référentiel de processus de V&V propose une collection organisée, formalisée et standardisée des pratiques de V&V qui pourront être réutilisés dans les projets du logiciel critique. Le référentiel est « organisé » car il existe un modèle organisé de processus (OpenUP) qui fait le support du référentiel de V&V. Le référentiel est « formalisé » car les pratiques de V&V (ie. les processus, tâches, rôles, artefacts, etc.) sont basées sur SPEM-OMG (OMG, 2008). Le référentiel est « standardisé » car le modèle est instancié dans chaque projet à partir d'un ensemble standard des pratiques de V&V;

- **une bibliothèque des expériences du passé,** le modèle des processus de V&V permet d'accumuler des expériences des projets réalisés (ensemble des activités pour des projets particuliers dans la forme de librairies ou des plugiciels des projets). Ceci devrait améliorer la façon de développer et gérer les projets du logiciel critique dans les différents équipes.

  Le fait d'accumuler des expériences des processus est demandé par le CMMI. Cette exigence est nécessaire pour avoir le niveau 3 de maturité. Plus précisément le Process Area qui établit cette exigence est : « Organizational Process Definition +IPPD (OPD) », « Specific Practice (SP): 1.5 Establish the Organization's Process Asset Library ». D'ailleurs la norme ISO 15504 demande également un entrepôt de processus pour l'organisation afin d'atteindre le niveau 3 et plus de maturité (le processus est défini);

- **une cohérence avec le guide DO-178B, et les processus unifiés** pour la construction du logiciel critique.

- **une collection systématique des processus qui facilite l'implantation de la V&V** dans les projets, par le biais de la définition des pratiques pertinentes pour la définition et l'exécution du V&V. Cette collection des processus faciliterait également le conformité des exigences au niveau de SG et SP (« Specific Goals and Specific Practices ») des processus de vérification et validation décrites dans CMMI (référence section 3.5.1 du chapitre 3);

- **influence sur le retour de l'investissement,** étant donné que le référentiel de pratiques et des mesures de V&V comporte une amélioration et un meilleur contrôle des pratiques de construction du logiciel critique et de leurs estimations, et par conséquence leur qualité, il est envisagé un impact positif sur la diminution des coûts de la V&V pour la certification des logiciels;

- **l'amélioration des pratiques de V&V déjà existantes dans les organisations**. comme nous l'avons vu, le modèle des processus de V&V proposé dans le cadre de cette thèse incorpore une structure organisée des pratiques (disciplines, activités, rôles, responsabilités, etc.) alignées aux principes du développement agile (car l'OpenUP est est cohérent avec les principes des méthodes agiles), la formalisation des processus type SPEM-OMG et la modélisation Orienté Objets. Alors, ces pratiques pourront constituer un point de départ pour une évaluation des pratiques dans les organisations. D'ailleurs l'entrepôt des mesures de V&V permet de collecter des données des mesures d'une façon systématique pour les différents processus et produits du projet de V&V, donc une façon quantitative d'évaluation des projets;

- **la définition des processus de V&V,** à partir du référentiel des processus de V&V on pourrait identifier des processus pertinents à implanter dans le cadre des exigences particulières des projets;

- **l'évaluation de la pertinence des processus de V&V,** comme nous avons déjà mentionné le modèle des processus de V&V et des mesures pourra être utilisé comme base pour une évaluation des pratiques dans les organisations;

- **la diminution des risques dans la construction du logiciel,** par le biais de l'implantation du référentiel des processus qui permettra de mieux faire l'identification des processus, tâches, rôles, artefacts le suivi dans un projet, le tout contribuant à la diminution des risques d'échec des projets;

- **faciliter la communication entre les différentes équipes du projet de V&V,** étant donné que le modèle des processus de V&V constitue une collection des pratiques de V&V organisées d'une façon structurée et intégrée avec des relations entre eux par le biais du V&V plugiciel, les différents acteurs sont impliqués dans le processus de construction du logiciel et sa certification. D'ailleurs le plugiciel de V&V pourra être publié sur un site Web pour faciliter la communication entre les diverses intervenants du projet;

- **faciliter la formation,** étant donné que la documentation des projets serait facilement disponible par le biais du plugiciel qui sera publié par exemple dans l'intranet de l'organisation, il est à prévoir que la formation serait améliorée.

10.1.1.2 Faiblesses

- Étant donné que le référentiel des pratiques de V&V est basé sur OpenUP qui est un outil du logiciel libre, éventuellement on pourrait trouver que ce n'est pas évident d'avoir des nouvelles améliorations ou corrections pour le produit (exemple : patches, correction des bugs etc.) ou bien il est possible de rencontrer certaines difficultés pour avoir un support pour l'implantation, entre autres.

10.1.2 Contraintes pour l'implantation du référentiel des processus de V&V

Afin de bien déployer le modèle des processus de V&V dans les contexte des projets et au sein d'une organisation, il faut maintenir des bonnes pratiques (Y.Wang and A. Bryant, 2002), (Marc Lacroix et Pierre-N. Robillard, 2007), à savoir :

- **Analyser les besoins de l'organisation et ses objectifs en termes de V&V,** afin de bien cibler des éventuelles possibilités d'amélioration des pratiques et méthodes de V&V au sein des équipes des projets.

- **Identifier les pratiques de V&V à faire évoluer,** soit dans l'organisation ou dans le référentiel.

- **Établir des priorités et un planning,** pour l'implantation du référentiel. Établir une liste des améliorations et des priorités. Valider les améliorations proposées avec des différentes équipes dans l'organisation.

- **Planifier le développement et l'adaptation du** plugiciel **pour les activités de V&V de l'organisation,** prendre en considération du temps pour le développement.

Autres facteurs à considérer :

- **Identifier le facteur humain dans l'implantation**, la bonne communication entre l'équipe en charge d'implanter le modèle des processus de V&V et le personnel de l'organisation où le modèle serait implanté. Il faut chercher l'adhésion du personnel.

- **Le facteur temps,** il faut planifier d'implanter le modèle des processus de V&V dans un délai raisonnable, afin de garder l'intérêt des employés.

- **Maintenir un niveau de documentation assez simple,** afin que la compréhension du modèle des processus de V&V à implanter ne soit pas un obstacle pour la lecture.

- **L'implantation d'un programme de mesures,** dans l'organisation faciliterait l'implantation du référentiel des pratiques de V&V.

10.1.3 Autres types de confirmation de la pertinence du référentiel de la V&V

Comme nous l'avons déjà mentionné dans les chapitres précédents, nous avons publié des articles de recherche qui montrent les différentes contributions du référentiel de V&V. Ces articles ont été évalués et approuvés par des comités d'experts avant d'être publiés (ex. NASA, IEEE et autres). Il est important de mentionner que chaque publication a reçu des commentaires des membres du comité d'acceptation des articles et ces commentaires ont été incorporés dans les améliorations pour le référentiel de V&V. L'annexe I présente une liste des articles publiés.

Nous voudrions également mentionner que nous avons fait des confirmations supplémentaires de la pertinence du référentiel et de l'entrepôt des mesuresl. Ces confirmations ont consisté en ce qui suit :

- **Le besoin d'un plugiciel pour la V&V.** Le Dr. Chris Sibbald and M. Kurt Sand de l'entreprise Telelogic (Chris Sibbald and Kurt Sand, 2007) ont publié un article sur l'Internet qui fait mention expressément du besoin de disposer d'un plugiciel de type OpenUP qui adapterait les principes du guide DO-178B pour les projets du logiciel critique.

- **La structure et de l'organisation du plugiciel pour la V&V.** Nous avons été invités à participer à une conférence offerte par « Osellus[6] » chez « CMC Électronique » à

[6] Osellus (www.osellus.com) est une entreprise canadienne (de l'Ontario) dédié à offrir la réingénierie et à l'implantation des outils pour les processus au sein des organisations. Ils sont contributeurs d'OMG-SPEM

Montréal par rapport à l'implantation d'un Framework de type plugiciel des processus pour la V&V. Nous avons validé nos hypothèses de recherche avec les experts présents à cette réunion. La conférence nous a permis d'un côté de valider la cohérence de notre approche, ie. OpenUP-EPFC et les objectifs du guide DO-178B, et d'un autre côté nous avons eu la possibilité de confronter notre vision de la façon d'organiser et de structurer le plugiciel pour la V&V dans notre recherche.

- **Entretien avec un expert de la construction du logiciel critique.** Nous avons aussi eu la possibilité de discuter avec M. Pierre Labrèche (Labrèche 2008) par rapport à la façon d'implanter la DO-178B dans les projets chez CMC Électronique. Plus particulièrement, nous avons discuté des principes du LLR (« Low Level Requirements »), des HLR (« High Level Requirements »), Architecture du logiciel, entre autres. Cette entrevue nous a permis de valider la pertinence des processus de construction du logiciel critique que nous avons inclus dans le référentiel de la V&V.

- **Besoin de l'entrepôt pour les mesures de la V&V.** Nous avons publié un article par rapport à l'implémentation des mesures de performance dans les projets de V&V (Palza, Abran et Fuhrman, 2004). Nous avons utilisé le cycle de vie de projets de V&V décrit dans la norme IEEE 1012. Dans l'article nous proposons l'utilisation de l'entrepôt de mesures afin de faciliter la gestion de la performance des projets de V&V. L'article a été publié en Allemagne. Voir Annexe I pour plus de détails.

- **Besoin général du référentiel de V&V.** Bertrand et Fuhrman ont publié un article sur la pertinence des pratiques d'OpenUP pour combler les objectifs du guide DO-178B. Nous avons suivi ces conseils et nous sommes allés plus loin dans l'exploration et l'implémentation des pratiques de V&V selon l'exigence du DO-178B, et en conformité avec l'OpenUP. Nous avons analysé, fait la conception et la construction d'un plugiciel pour les exigences du DO-178B basé sur OpenUP.

- **Échanges de commentaires avec des membres de l'entreprise CMC Électronique aux États Unis**. Le Prof. Fuhrman a établi le contact avec CMC Electronique aux États Unis pour l'évaluation des pratiques de la construction du logiciel critique. Ces pratiques ont fait partie des conseils du Prof. Fuhrman pour l'amélioration du référentiel des processus de V&V.

- **Échanges des commentaires avec des membres de l'entreprise CMC Électronique-Montréal, CS Canada-Montréal.** . Le Prof. Fuhrman a établi des communications avec CMC Électronique à Montréal, CS Canada à Montréal dans le cadre des projets du logiciel critique. Les conversations ont servi pour améliorer notre approche de recherche.

10.2 Évaluation de l'entrepôt des mesures pour la V&V

Nous avons expliqué dans les chapitres précédents que l'entrepôt des données de mesures a été développé en prototype et que ce prototype a été testé et approuvé par des représentants de notre partenaire industriel Ericsson Research Canada..

Le test du prototype a été appliqué sur l'évaluation des capacités de l'entrepôt à fournir des mesures au niveau du temps et de l'effort pour les projets, à savoir : « Performance Indicator (PI) » et « Earned Value (EV) ». Également les capacités OLAP ont été testées avec succès (voir chapitre 9 et annexe X).

Il faut mentionner que l'évaluation a été effectuée par le biais de la cohérence avec des exigences de la gestion de la V&V selon la norme IEEE 1012 (voir chapitre 9).

Voici une description des activités réalisées par notre partenaire au niveau de l'encadrement, la vérification et le support de ce livrable (voir aussi figure 10.2):

Ericsson Research Canada[7] : nous avons fait la recherche pour développer un entrepôt générique des mesures pour les projets. Les exigences pour l'entrepôt ainsi qu'un schéma de base des mesures ont été fournies par l'entreprise. Ces intrants ont été définis dans le contrat ainsi que les conditions des extrants de réussite. Nous avons réalisé des publications internationales avec nos résultats de recherche (voir annexe I : Publications de recherche). Dans le cadre de cette thèse nous avons utilisé l'entrepôt des mesures pour le cycle de vie de projets de V&V.

Autres références de l'industrie et de la recherche : nous avons également identifié des références au niveau de la recherche ainsi que des publications des expériences au niveau industriel afin de bâtir un ensemble des caractéristiques pour l'entrepôt des mesures pour des projets de V&V (voir chapitre 9).

---

[7] http://www.ericsson.com

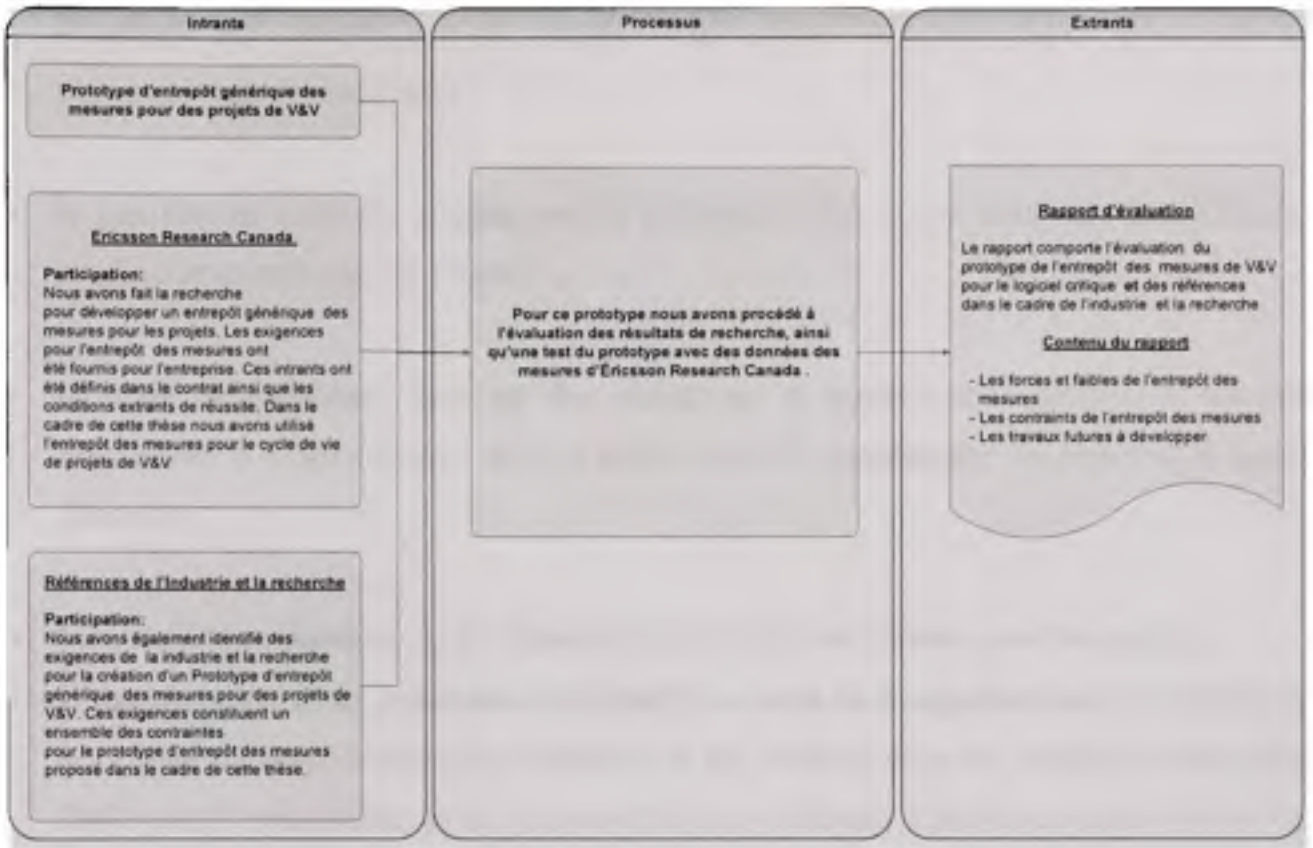

Figure 10.2 ***Évaluation des résultats de l'entrepôt des mesures de V&V.***

10.2.1 Forces de l'entrepôt des mesures pour le V&V

L'entrepôt des mesures permet entre autres :

- **la prise de mesures pour les projets,** d'une façon structurée organisée, centralisée, soit par exemple au niveau projet, département, organisation. Dans (Palza, Abran et Fuhrman, 2003) nous avons analysé comment l'entrepôt des mesures est capable de combler les exigences des différents « Process Areas » du CMMI par rapport à la gestion des projets;

- **la compréhension quantitative de l'état du projet,** en termes de coûts soit par exemple au niveau du produit ou processus. Dans (Palza, Abran et Fuhrman, 2003) nous avons analysé comment l'entrepôt des mesures est capable de combler les exigences des différents « Process Areas » du CMMI par rapport à la gestion quantitative des projets ;

- **la collecte des mesures,** soit par exemple au niveau des différentes phases des projets ou bien au niveau des livrables ;

- **le contrôle et suivi du projet,** par les différents intervenants ainsi que de différents artefacts et composants du projet ;

- **la prise de décisions,** basé sur des indicateurs et représentation graphique des ces indicateurs. Il serait un outil facile à utiliser pour le gestionnaire du projet et la haute direction ;

- **historique des mesures,** La formation d'un historique de mesures pour les projets ;
- **L'implantation d'un processus de maturité au sein de l'organisation,** étant donné la possibilité de quantification des processus et des produits dans les projets, il serait plus facile pour l'organisation de se faire certifier dans différents modèles de génie logiciel ou de qualité comme par exemple CMMI, ISO 9001, entre autres ;

- **la comparaison entre projets,** au niveau de coûts, échéanciers, ressources, etc.;

10.2.2 Faiblesses de l'entrepôt des mesures pour la V&V

- l'application de l'entrepôt se fait à partir des nouveaux projets. Pour les anciens projets il faudrait faire un travail de normalisation des données pour les intégrer dans l'entrepôt (bien sur lorsque l'organisation a accès à ces données);

- la collecte des mesures est manuelle. La possibilité d'intégration avec d'autres outils comme par exemple de type ETL (Extract, Transfer Load) serait envisagé mais cela demanderait un projet particulier pour évaluer la faisabilité ;

- l'entrepôt des mesures a été développé et testé en prototype : il n'a pas été mis en production pour une organisation ;

- l'entrepôt a été testé avec des données de mesures liées à des projets de télécommunications (pas nécessairement des logiciels critiques). Les mesures qui ont été représentées dans les cubes OLAP sont les suivantes : CPI (Cost Performance Index) et le SPI (Schedule Performance Index) ;

- pour utiliser l'entrepôt dans un contexte industriel, il faudrait demander l'autorisation à Ericsson Research Canada ;

- nous n'avons pas de données des mesures des projets de V&V pour établir des analyses ;

10.2.3 Contraintes de l'entrepôt des mesures pour le V&V

- un processus de développement cohérent et répétitif déjà en place serait convenable pour implanter plus facilement l'entrepôt des mesures ;

- avant d'appliquer l'entrepôt des mesures dans les projets il faut se mettre d'accord entre les diverses équipes du logiciel critique, par exemple à quel moment, quelle étape, phase des données des mesures devront être colletées pour les projets ;

- faire la formation pour l'utilisation de l'interface de l'entrepôt et la manipulation des cubes OLAP avec l'outil ;

Critères particulières pour la construction des cubes OLAP pour l'entrepôt des mesures

Les mesures sont traitées dans l'entrepôt à travers des cubes multidimensionnels de type OLAP. Donc, la création de nouvelles mesures et leurs relations devra être définie par la création de cubes OLAP. Dans les paragraphes suivants nous mentionnons certaines

contraintes (critères) à-propos de la création des mesures et de l'implémentation des cubes multidimensionnels des données des mesures :

- le méta modèle, sur lequel l'entrepôt est basé, ne présuppose pas de calculs préalables sur des mesures particulières ou des relations des mesures particulières. Les mesures comme telles sont traitées comme des données.

- les données qui définissent les mesures et les relations qui sont stockées dans l'entrepôt sont appelées métadonnées.

- l'architecture de l'entrepôt contient seulement des mesures de base. Les mesures dérivées sont construites par le « Analytical Engine » (i.e. MS-Analysis Services).

- l'entrepôt est bâti sur des relations de hiérarchie entre les entités, c'est-à-dire sur les mesures associées avec une entité de niveau inférieur ou supérieure : il est possible de les rassembler ou agréger avec la capacité « rolled-up » « drill-down » de l'entrepôt.

CONCLUSION

Nous avons indiqué que nos sociétés actuelles sont devenues de plus en plus dépendantes du logiciel et des systèmes informatiques. Nous avons expliqué comme l'informatique influence notre quotidien et facilite également la globalisation mondiale. On constate que l'informatique occupe progressivement une place qui devient indispensable. En parallèle, les projets informatiques et en particulier ceux du logiciel sont devenus progressivement plus sophistiqués et plus complexes. Les facteurs qui compliquent cette réalisation sont nombreux : la composition hétérogène des logiciels sur une diversité de systèmes, l'exécution distribuée du logiciel, la complexité dans le développement avec des algorithmes de calcul complexe, la multiplicité de sous-traitants avec des méthodologies de développement variées, etc.

Cette croissance de sophistication et de complexité du logiciel augmente les risques, incrémentant les coûts de développement ainsi que les coûts de maintenance. Certains logiciels dits critiques sont plus susceptibles, s'ils ne sont pas d'un certain niveau de qualité en termes de la sûreté et la fiabilité, de mettre en péril la vie humaine et/ou risquent de grands investissements financiers. Nous avons aussi présenté quelques accidents dus à des défauts dans les logiciels.

Ces enjeux nécessitent souvent la conception et le développement des applications critiques avec des méthodes et des processus de vérification et de validation (V&V) du logiciel à un niveau rigoureux. D'ailleurs, ce domaine de la V&V revêt une importance croissante et les besoins sont reconnus, concernant tant les développements de la recherche, que la nécessité d'implanter ces modèles dans l'industrie.

Nous avons expliqué que les méthodes actuelles de V&V des logiciels que l'on retrouve souvent dans l'industrie ne rencontrent pas toujours les caractéristiques nécessaires pour combler les besoins de qualité à un degré rigoureux et encore à un coût acceptable pour les organisations. L'industrie a un besoin d'un référentiel de V&V du logiciel critique qui serait à la fois adapté aux nouveaux modèles et méthodes du génie logiciel. Nous constatons que,

dans plusieurs cas, les normes et modèles existants n'incorporent pas les nouvelles approches du génie logiciel.

Nous avons expliqué dans les chapitres initiaux de cette thèse qu'il existe un manque d'un modèle unifié pour les pratiques de V&V pour le logiciel critique qui soit cohérent avec le guide DO-178B. Ce modèle devrait avoir une collection cohérente et intégrée des processus et des artefacts de V&V (procédures, flux, acteurs, rôles, responsabilités, etc.) en concordance avec les exigences de la certification du guide DO-178B et avec les principes des processus unifiés de développement logiciel (« Unified Process »).

Nous avons également remarqué qu'il existe un manque d'outil flexible (artefact logiciel type plugiciel) pour la formalisation et l'instanciation des processus de V&V dans les projets du logiciel critique. Cet outil devrait faciliter par exemple : (a) la formalisation et l'instanciation des processus, procédures, acteurs, des artefacts etc., de la V&V dans les projets; (b) la meilleure planification des ressources, les processus et les artefacts de V&V dans les projets; (c) le maintien pour chaque projet de la traçabilité entre les processus, les produits et leurs artefacts; (d) le maintien d'un historique des projets, facilement accessible et utilisable; (e) le maintien d'une communication fluide entre les différentes équipes du projet par rapport à ces responsabilités associées; (f) faciliter l'institutionnalisation et la standardisation des pratiques de V&V dans l'organisation par le biais d'un outil et des pratiques uniques pour les projets de logiciel critique.

Nous avons constaté, dans le contexte de la gestion de la performance et du contrôle des projets de V&V, le manque d'un entrepôt basé sur des mesures génériques pour la définition, la collecte, l'évaluation et la représentation des données des mesures des projets du logiciel critique. Un tel entrepôt de mesures devrait permettre : (a) de mieux planifier les coûts des pratiques de V&V à partir des historiques des projets (étant donné qu'il existerait un entrepôt des données de mesures historiques des projets); (b) de mieux faire le suivi des pratiques de V&V dans les phases des projets en termes de ressources, du temps, etc.; (c) d'avoir la

possibilité d'un suivi et contrôle permanent de l'état de l'avancement des activités dans les projets de V&V ; (d) de mieux évaluer, planifier et exécuter la gestion des changements dans les projets ; (e) de faire une quantification et par conséquence une amélioration continue dans la réalisation des projets grâce à la capacité de l'entrepôt de garder la mémoire des projets en terme d'indicateurs des mesures.

Objectifs de la thèse

Les objectifs de recherche que nous avons accomplie et présentée dans le cadre de cette thèse s'inscrivent dans deux livrables, à savoir:

- L'analyse, la conception et l'implémentation d'un référentiel formalisé et unifié des artefacts du logiciel pour l'instanciation des processus de V&V du logiciel critique.
- L'analyse, la conception et l'implémentation d'un entrepôt générique des mesures de V&V pour la prise de décisions dans les projets du logiciel critique.

Analyse préliminaire

- Nous avons présenté une analyse des processus du développement de logiciel critique selon la DO-178B. Cette analyse à été faite à partir des documents que nos partenaires industriels nous ont fournis CMC Electronics (CMC Electronics, 2007), Verocel (Verocel, 2005), Ericsson Research Canada, NASA IV&V Facility. ainsi que des conversations avec des experts dans le domaine (Labrèche 2008).

- Nous avons fait une analyse et une modélisation des processus de logiciel selon les exigences de la norme IEEE 1012 pour la V&V du logiciel, et l'outil de RUP XDE. Les résultats ont été montrés dans le chapitre 6 et dans l'annexe II de cette thèse. Les résultats ont été publiés comme articles de recherche (Fuhrman. Djlive. Palza, 2003).

Contributions de recherche

- Nous avons fait l'analyse et la proposition des nouvelles pratiques et des améliorations des pratiques déjà existantes dans OpenUP afin de combler les exigences du guide DO-178B. Ces nouvelles pratiques et des améliorations dans OpenUP ont été faites pour combler les processus du guide DO-178B dans le contexte suivant :
  - Le processus des exigences ;
  - Le processus du design ;
  - Le processus de codification ;
  - Le processus d'intégration ;
  - Le processus de vérification.

- Nous avons construit un artefact logiciel appelé « Plugin » pour le processus de la vérification du logiciel critique selon les exigences du guide DO-178B. Le plugiciel a été créé à partir de l'analyse des spécifications de conception que nous avons développées pour combler les objectifs du DO-17B. Le plugiciel pour la V&V a été créé à partir du processus unifié « OpenUP » et modélisé avec l'outil « Eclipse Process Framework Composer ».

- L'analyse, le design, et le développement d'un prototype d'entrepôt générique des mesures pour la V&V. Cet entrepôt est basé sur un modèle générique des mesures qui est construit à partir d'un schéma des classes d'objets. Le modèle des mesures a été implémenté avec la technologie OLAP. Nous avons décrit le détail de la conception et développement de l'entrepôt des mesures pour la V&V dans le chapitre 9 et l'annexe X. Un ensemble d'articles de recherche ont été publiés sur l'entrepôt des mesures et leur utilisation pour mesurer la performance des projets dans la V&V (Palza, Abran et Fuhrman, 2004) .

RECOMMANDATIONS

10.2.4 Travaux futurs du modèle des processus de V&V

- Faire des implantations du référentiel de processus de V&V dans plusieurs projets dans l'industrie et éventuellement considérer des nouvelles versions du référentiel et des plugiciels associés.

- Le référentiel des processus de V&V est composé des pratiques pour la construction du logiciel critique qui est applicable dans d'autres secteurs comme par exemple le médical, l'automoteur, le nucléaire, entre autres. En ce moment pour combler les objectifs de sûreté de chaque secteur il serait nécessaire de créer des adaptations du référentiel à chaque norme reliée à la sûreté critique (« Safety Critical ») et reliée au secteur en particulier. Cette adaptation prendrait la forme de la création des plugiciels.

10.2.5 Travaux futurs de l'entrepôt des mesures pour la V&V

- Faire des implantations de l'entrepôt des mesures pour la V&V dans plusieurs projets dans l'industrie et éventuellement considérer des nouvelles versions de l'entrepôt avec des nouveaux cubes OLAP des mesures associées.

- L'entrepôt des mesures pour la V&V est assez flexible pour faire l'analyse multidimensionnelle des données des mesures dans plusieurs autres secteurs de la sûreté critique (« Safety Critical ») comme par exemple le médical, l'automoteur, le nucléaire, entre autres. Il est important de noter que l'entrepôt est aussi utile pour les mesures dans les domaines non reliés à la sûreté. En ce moment pour combler les objectifs des mesures de sûreté dans d'autres secteurs il serait nécessaire de créer des cubes OLAP reliés à chaque secteur en particulier.

ANNEXE I

LISTE DES PUBLICATIONS DE RECHERCHE

1. Fuhrman Christopher, Palza Edgardo, Kim Loan DO, "Optimizing the Planning and Executing IV&V in Mature Organizations", IEEE 28th Annual International Computer Software and Application Conference (COMPSAC-2004), Hong Kong, 28-30 September 2004.

2. Palza Edgardo, Abran Alain, Fuhrman Christopher, "V&V Measurements Management Issues in Safety-Critical Software", 14th IWSM2004, International Workshop on Software Measurement, Germany, November, 8-12, 2004.

3. Belkbebir Y, Bourque P, Doucet M, Laporte C, Robert F, Palza E, "Coverage Analysis and Improvement of Role Definitions of the Bombardier Transportation Software Engineering Process", 17th International Conference "Software & Systems Engineering and their Applications" ICSSEA 2004, Paris.

4. Fuhrman Christopher, Fatime Djlive, Palza Edgardo, "Software Verification and Validation within the (Rational) Unified Process", 28th Annual IEEE/NASA Software Engineering Workshop, Greenbelt, MD, USA 2003.

5. Palza Edgardo Abran Alain, Fuhrman Christopher "Establishing a Generic and Multidimensional Measurement Repository in CMMI context", 28th Annual IEEE/NASA Software Engineering Workshop, Greenbelt, MD, USA 2003.

6. Abran Alain, Palza Edgardo, "Design of a Generic Performance Measurement Repository in Industry" in 13th International Workshop on Software Measurement, Montréal (Québec), September, Canada 2003.

7. Palza Edgardo, "Facilitating Measurement Indicators in Software Improvements Projects", Systems & Software Engineering Review, Faculty of Systems & Computer Engineering, San Marcos University. Lima - Peru 2005.

ANNEXE II

ANALYSE ET MODELISATION DES PROCESSUS DE LA NORME IEEE 1012 VS. LE RUP

Analyse et modélisation des pratiques de la norme IEEE 1012 (IEEE Std 1012, 2004) avec RUP

Cette section présente l'analyse et la modélisation de toutes les activités du processus des tests de la V&V de la norme IEEE 1012. Nous analyserons le dégrée de couverture des pratiques du « Rational Unified Process - (RUP) » afin d'essayer de combler les exigences de V&V telles que décrites dans la norme IEEE 1012.

L'analyse et la modélisation sont réalisées au niveau des activités, rôles, produits de travail tels que décrits dans le RUP. La modélisation de l'analyse de couverture des pratiques de V&V est faite avec l'outil RUP XDE qui a été présenté dans le chapitre 3.

La démarche suivie a été la suivante : Nous avons chacune des activités mentionnées dans la section 5.4.5 – les tests de V&V de la norme IEEE 1012, et nous avons analysé le dégré de couverture en termes des pratiques mentionnées dans les différents processus du RUP. Une fois la réflexion pour l'activité complétée, nous avons modélisé les diverses artefacts et rôles impliqués dans l'activité en particulier.

Il faut noter que nous avons établi une relation de type « contributes » (ou de généralisation) des différents rôles de chaque activité de V&V (ex. V&V Traceability Analyst, V&V Security Analyst, etc.) au rôle principal appelé « V&V Analyst » pour le processus de V&V. Voir le figure II.1.

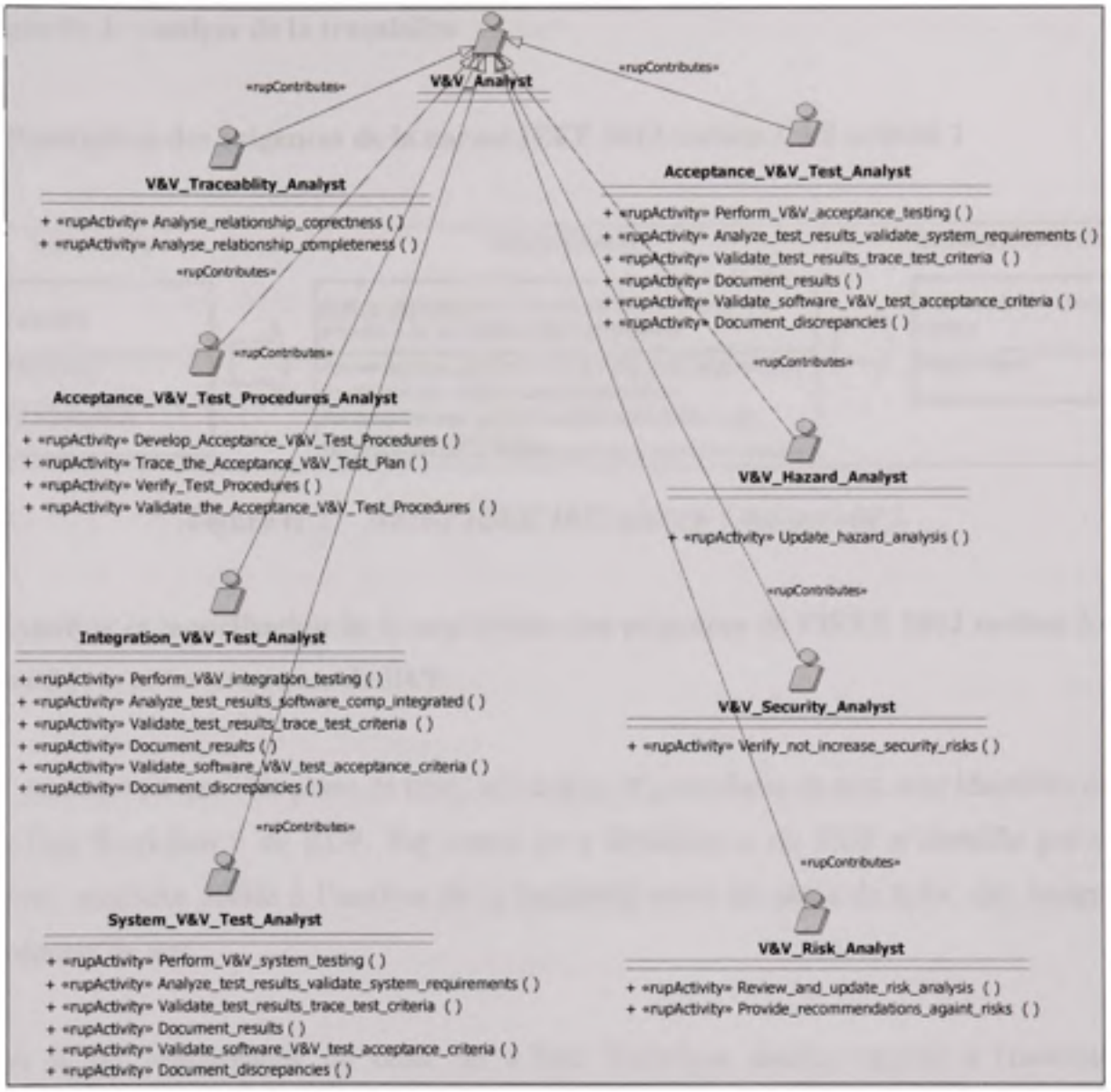

Figure II.1 ***Relation type « Contributes » avec le rôle V&V Analyst.***

1. Activité 1: Analyse de la traçabilité

1.1 Description des exigences de la norme IEEE 1012 section 5.4.5 activité 1

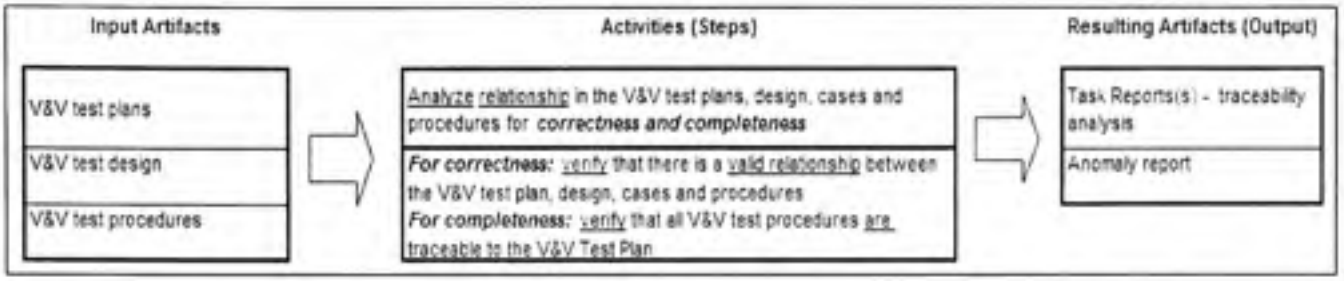

Figure II.2 *Norme IEEE 1012 section 5.4.5 activité 1.*

1.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 1, avec les pratiques de RUP

Les artefacts tel que : les plans de tests, test design et procédures de test, sont identifiés dans le « Test Workflow » de RUP. Par contre ce « Workflow » du RUP n'identifie pas une activité explicite dédiée à l'analyse de la traçabilité entre les plans de tests, test design et procédures de test.

Alors il est recommandé de créer un « Test Workflow detail » appelé « Traceability Analysis » à l'intérieur du « Test Workflow » ou dans un workflow particulier pour la V&V (ex. « V&V Test Workflow »).

Selon le type de V&V à implanter dans le projet les artefacts serons produits complètement par le groupe de V&V (V&V indépendant) ou bien partiellement avec une participation des autres équipes du projet. Dans cette activité ne sont pas consignées des activités particulières reliées à la criticité du logiciel.

Le test workflow detail « Traceability Analysis» doit contenir les activités et pas (« steps ») suivants:

- « Analyse relationship for correctness »

- Verify that there is a valid relationship between the test plan, design, cases and procedures.

- « Analyse relationship for completeness ».
  - Verify that all test procedures are traceable to the Test Plan.

Les livrables de ce « Workflow » sont :

- Task Reports(s) - Traceability analysis.
- Anomaly report.

Tandis que le RUP ne présente pas une référence explicite telle que les exigences de la norme IEEE 1012, il est clair que la notion de traçabilité est incluse dans le RUP, voici quelques exemples:

- « Workflow Detail: Define Evaluation Mission », l'activité: « Activity: Define Assessment and Traceability Needs ».
- «Workflow Detail: Manage Changing Requirements", l'activité: «Activity: Manage Dependencies».
- «Workflow Detail: Achieve Acceptable Mission», l'activité: «Activity: Identify Test Ideas», step: «Maintain traceability relationships».

Ce nouveau « workflow » doit être exécuté soit par un analyste de V&V, dans le cas de la V&V indépendante, ou bien par le rôle « Tester » du RUP ou autre selon l'organisation du projet, voir figure II.3.

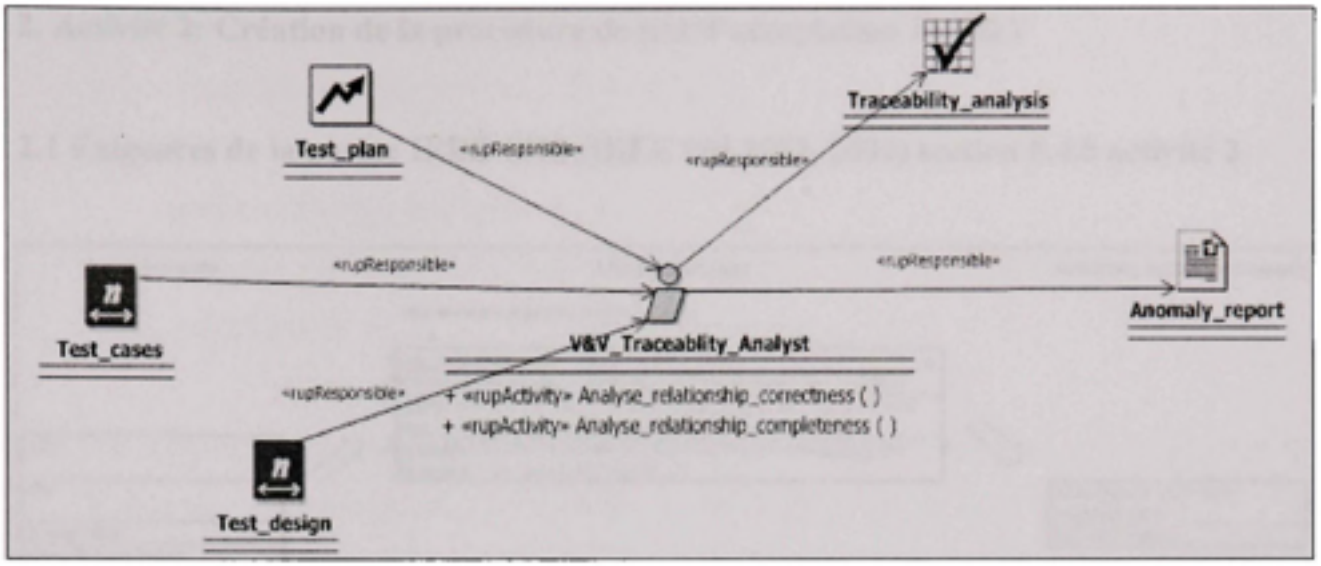

Figure II.3 ***V&V Traceability Analyst Role (IEEE 1012 and RUP).***

2. Activité 2: Création de la procédure de test d'acceptation de V&V

2.1 Exigences de la norme IEEE 1012 (IEEE Std 1012, 2004) section 5.4.5 activité 2

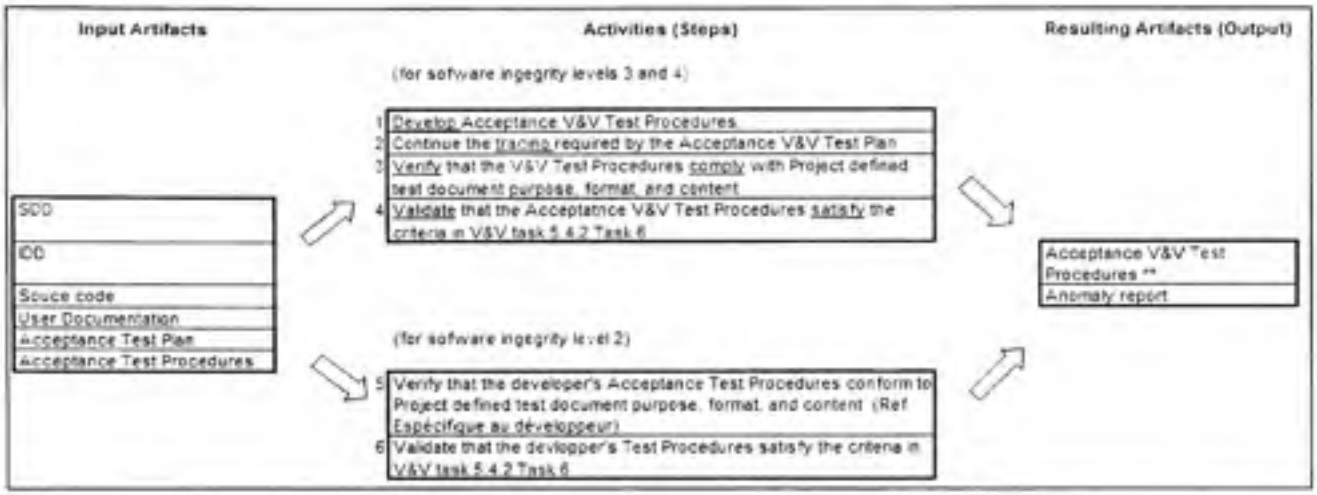

Figure II.4 *Norme IEEE 1012 section 5.4.5 activité 2.*

2.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 2 avec les pratiques de RUP

La norme IEEE 1012 décrit comme entrées des artefacts produits au niveau du système du logiciel, voici la description de quelques uns:

> « **Software design description (SDD):** A representation of software created to facilitate analysis, planning, implementation, and decision-making. The software design description is used as a medium for communicating software design information and may be thought of as a blueprint or model of the system ».
>
> « **Interface design document (IDD):** Documentation that describes the architecture and design interfaces between system and components. These descriptions include control algorithms, protocols, data contents and formats, and performance ».

Dans le « Test Workflow » ou dans un workflow particulier pour la V&V (ex. « V&V Test Workflow »), il faut créer un « workflow detail » appelé « Acceptance V&V Test Procedure Generation». Ce workflow doit prendre les suivantes caractéristiques :

a. Si le niveau de criticité du logiciel est 3 ou 4, il faut exécuter les activités suivantes

1. Develop Acceptance V&V Test Procedures.
2. Trace the Acceptance V&V Test Plan.
3. Verify the V&V Test Procedures to comply with Project defined test document purpose, format, and content.
4. Validate the Acceptance V&V Test Procedures to satisfy the criteria in V&V task 5.4.2 Task 6.

b. Si le niveau de criticité est de 2, le « Acceptance Test Procedures » développé par le développeur en concordance avec RUP, est suffisant pour combler les exigences de la norme IEEE 1012. Par contre il faut s'assurer d'exécuter une vérification des artefacts du développeur selon les exigences décrites pour la criticité du logiciel niveau 2, à savoir :
   1. Verify that the developer's Acceptance Test Procedures conform to Project defined test document purpose, format, and content.
   2. Validate that the developer's Test Procedures satisfy the criteria in V&V task 5.4.2 Task 6.

Des activités du RUP qui pourraient combler les exigences du logiciel avec une criticité niveau 2 sont:

- Au niveau du déploiement du produit:
  - « Workflow Detail: Manage Acceptance Test » et l'activité « Activity: Manage Acceptance Test» de la discipline « Deployment», rôle responsable: « Deployment Manager ».
- Au niveau de l'acceptation du produit :
  - « Workflow Detail: Plan the Project », et l'activité « Activity: Develop Product Acceptance Plan » de la discipline « Project Management », rôle responsable: « Project Manager »

Les livrables de ce « Workflow » sont :

- Acceptance V&V Test Procedures.
- Anomaly report.

Le nouveau « workflow » à incorporer dans le RUP sera utilisé principalement pour la construction des logiciels avec une criticité niveau 3 ou 4. Dans ce moment la V&V doit être exécuté soit pour un analyste de V&V (dans le cas d'une V&V indépendante), ou bien par le rôle « Tester », ou « Deployment Manager », ou « Project Manager » du RUP, selon l'organisation du projet, voir figure II.5.

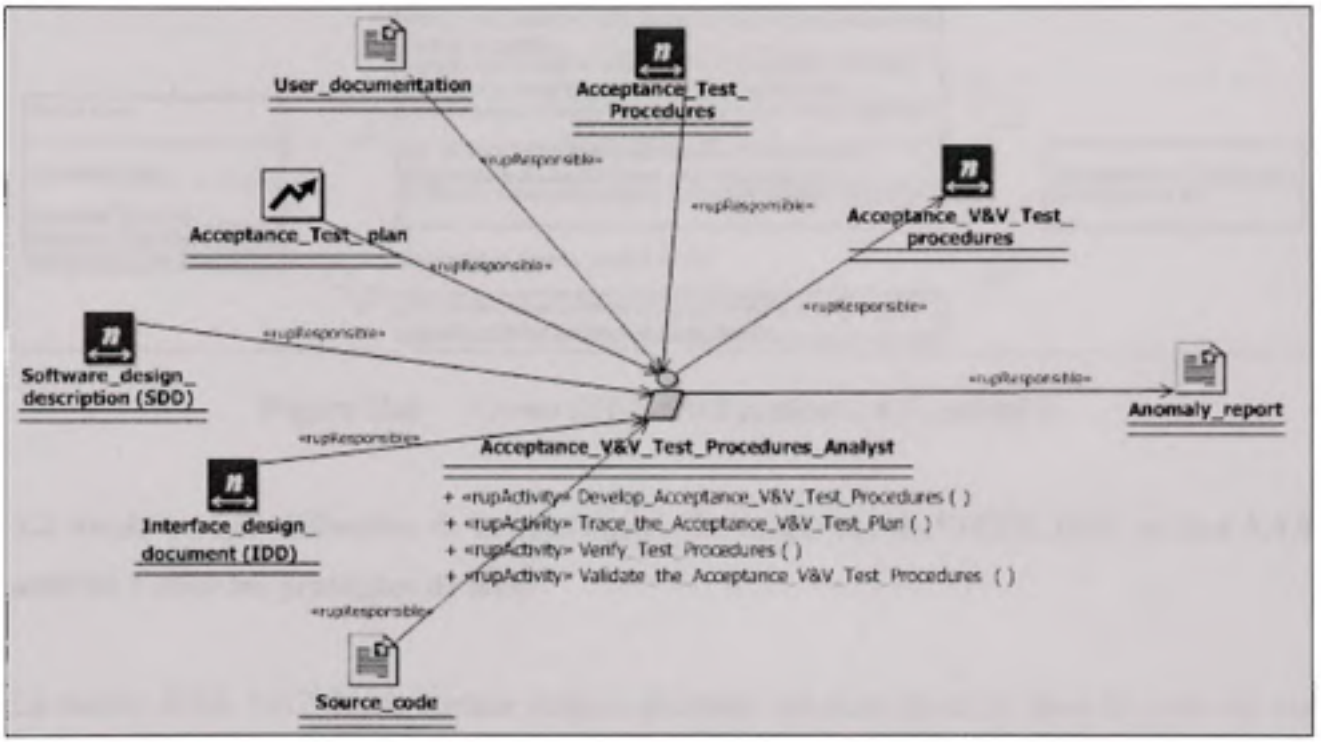

Figure II.5 ***V&V Acceptance V&V Test Procedures Analyst Role (IEEE 1012 and RUP).***

3. Activité 3: Exécution des tests d'intégration de V&V

3.1 Exigences de la norme IEEE 1012 (IEEE Std 1012, 2004) section 5.4.5 activité 3

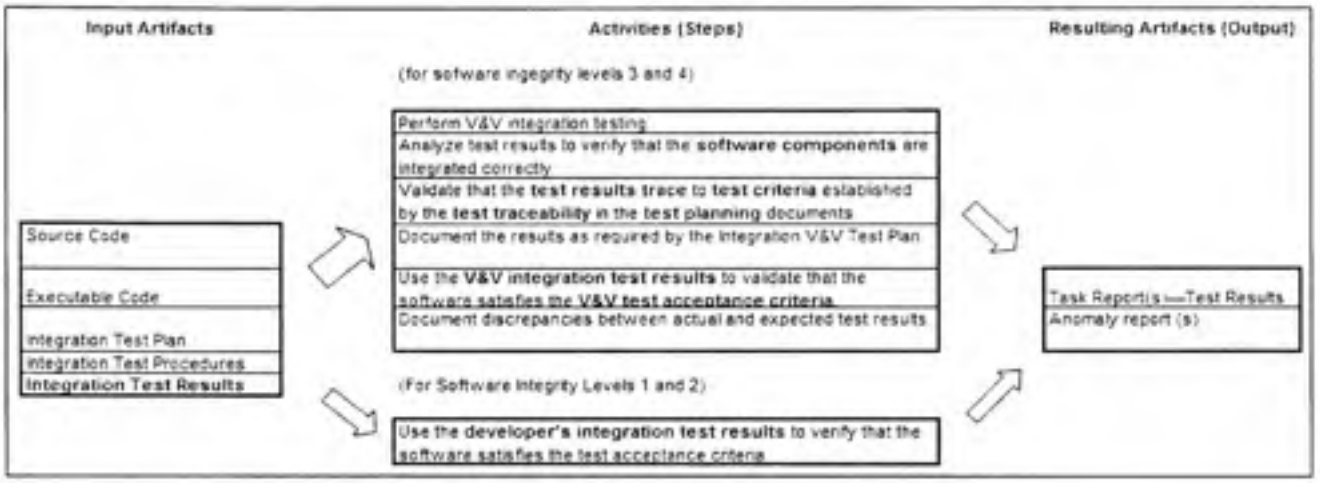

Figure II.6 ***Norme IEEE 1012 section 5.4.5 activité 3.***

3.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 3 avec les pratiques de RUP

La norme IEEE 1012 décrit comme entrées plusieurs artefacts produits dans le cycle de vie du logiciel, à savoir : Code source, code exécutable, plan des tests d'intégration, procédures des tests d'intégration, résultats des tests d'intégration.

Dans le « Test Workflow » ou dans un autre workflow spécifique pour la V&V (ex. « V&V Test Workflow »), il faut créer un « workflow detail » appelé « Integration V&V Test Execution ». Ce workflow doit prendre les suivantes caractéristiques :

a. Si le niveau de criticité du logiciel est 3 ou 4, il faut exécuter les activités suivantes :
    1. Perform V&V integration testing.
    2. Analyze test results to verify that the software components are integrated correctly.
    3. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.

4. Document the results as required by the Integration V&V Test Plan.
5. Use the V&V integration test results to validate that the software satisfies the V&V test acceptance criteria.
6. Document discrepancies between actual and expected test results.

b. Si le niveau de criticité est de 2, le « Integration Test Results» développé par le développeur en concordance avec RUP, est suffisant pour combler les exigences de la norme IEEE 1012. Par contre il faut s'assurer d'exécuter une vérification des artefacts du développeur selon les exigences décrites pour la criticité du logiciel niveau 2, à savoir :
    - Use the developer's integration test results to verify that the software satisfies the test acceptance criteria.

L'activité du RUP qui pourra combler les exigences du logiciel avec une criticité niveau 2 est:

- « Workflow Detail: Integrate the System», de la discipline « Iimplementation », rôle responsable: « System Integrator ».

Les livrables de ce workflow sont :

- Test Results.
- Anomaly report.

Le nouveau « workflow » à incorporer dans le RUP sera utilisé principalement pour la construction des logiciels avec une criticité niveau 3 ou 4. La V&V doit être exécutée soit par un analyste de V&V (dans le cas d'une V&V indépendante), ou bien par le rôle « Tester » ou « System Integrator » du RUP, selon l'organisation du projet. Voir figure II.7.

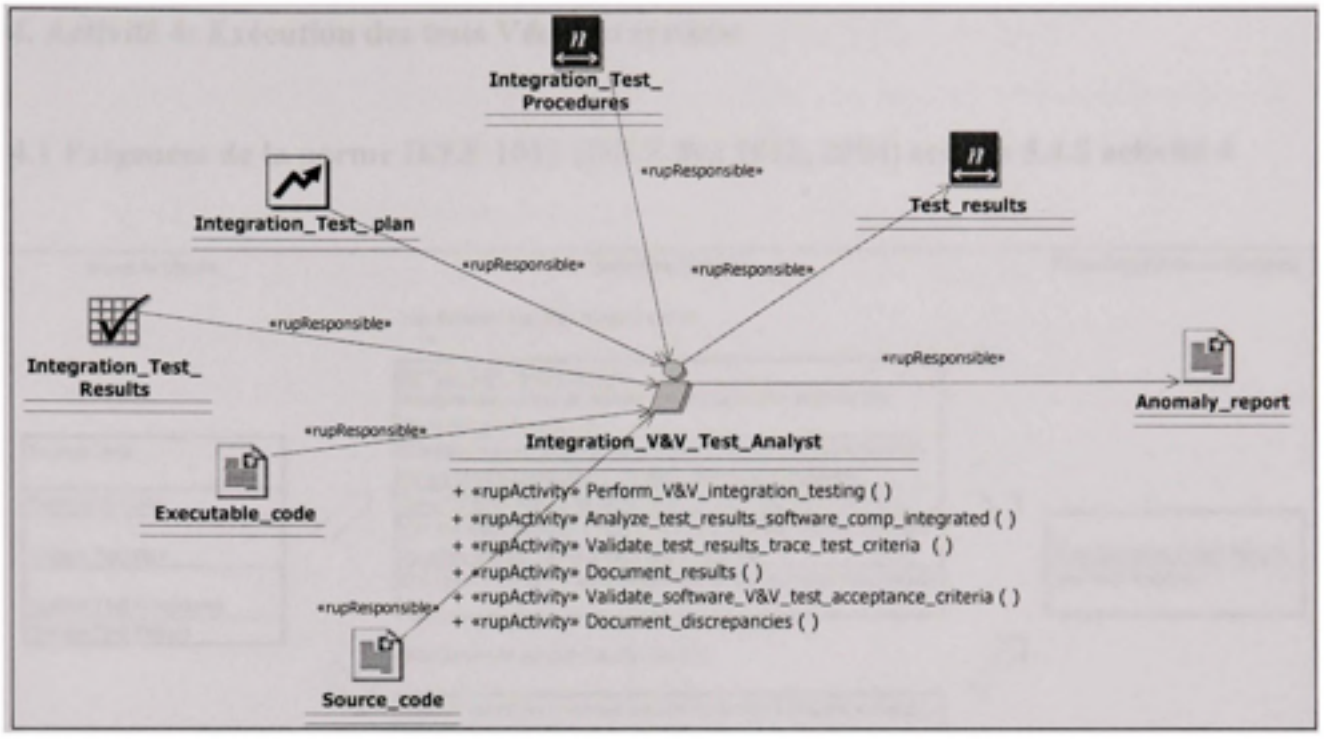

Figure II.7 ***Integration V&V Test Analyst Role (IEEE 1012 and RUP).***

4. Activité 4: Exécution des tests V&V du système

4.1 Exigences de la norme IEEE 1012 (IEEE Std 1012, 2004) section 5.4.5 activité 4

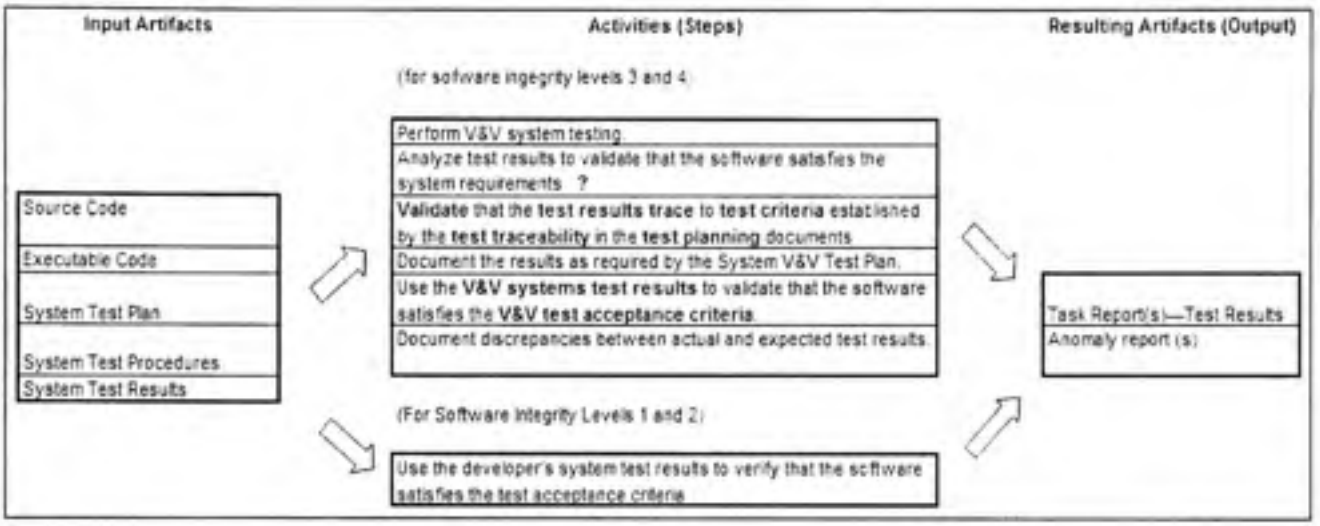

Figure II.8 ***Norme IEEE 1012 section 5.4.5 activité 3.***

4.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 4 avec les pratiques de RUP

La norme IEEE 1012 décrit comme entrées plusieurs artefacts produits dans le cycle de vie du logiciel, à savoir : Code source, code exécutable, plan de test du système, procédures des tests du système, résultats des tests du système.

Dans le « Test Workflow » ou dans un autre workflow spécifique pour la V&V (ex. « V&V Test Workflow »), il faut créer un « workflow detail » appelé « System V&V test execution». Ce workflow doit prendre les suivantes caractéristiques :

a. Si le niveau de criticité du logiciel est 3 ou 4, il faut exécuter les activités suivantes :
    1. Perform V&V system testing.
    2. Analyze test results to validate that the software satisfies the system requirements.
    3. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.

4. Document the results as required by the System V&V Test Plan.
5. Use the V&V systems test results to validate that the software satisfies the V&V test acceptance criteria.
6. Document discrepancies between actual and expected test results.

c. Si le niveau de criticité est de 2 ou 1, le « System Test Results» développé par le développeur en concordance avec RUP, est suffisant pour combler les exigences de la norme IEEE 1012. Par contre il faut s'assurer d'exécuter une vérification des artefacts du développeur selon les exigences décrites pour la criticité du logiciel niveau 1 et 2, à savoir :

   - Use the developer's integration test results to verify that the software satisfies the test acceptance criteria

L'activité du RUP qui pourra combler les exigences du logiciel avec une criticité niveau 2 est:

- « Workflow Detail: Test and Evaluate», de la discipline « Test», l'activité : « Activity: Execute Test Suite », rôle responsable: « Tester».

Les livrables de ce workflow sont :

- Test Results.
- Anomaly report.

Le nouveau « workflow » à incorporer dans le RUP sera utilisé principalement pour la construction des logiciels avec une criticité niveau 3 ou 4. La V&V doit être exécutée soit par un analyste de V&V (dans le cas d'une V&V indépendante), ou bien par le rôle « Tester » du RUP, voir figure II.9.

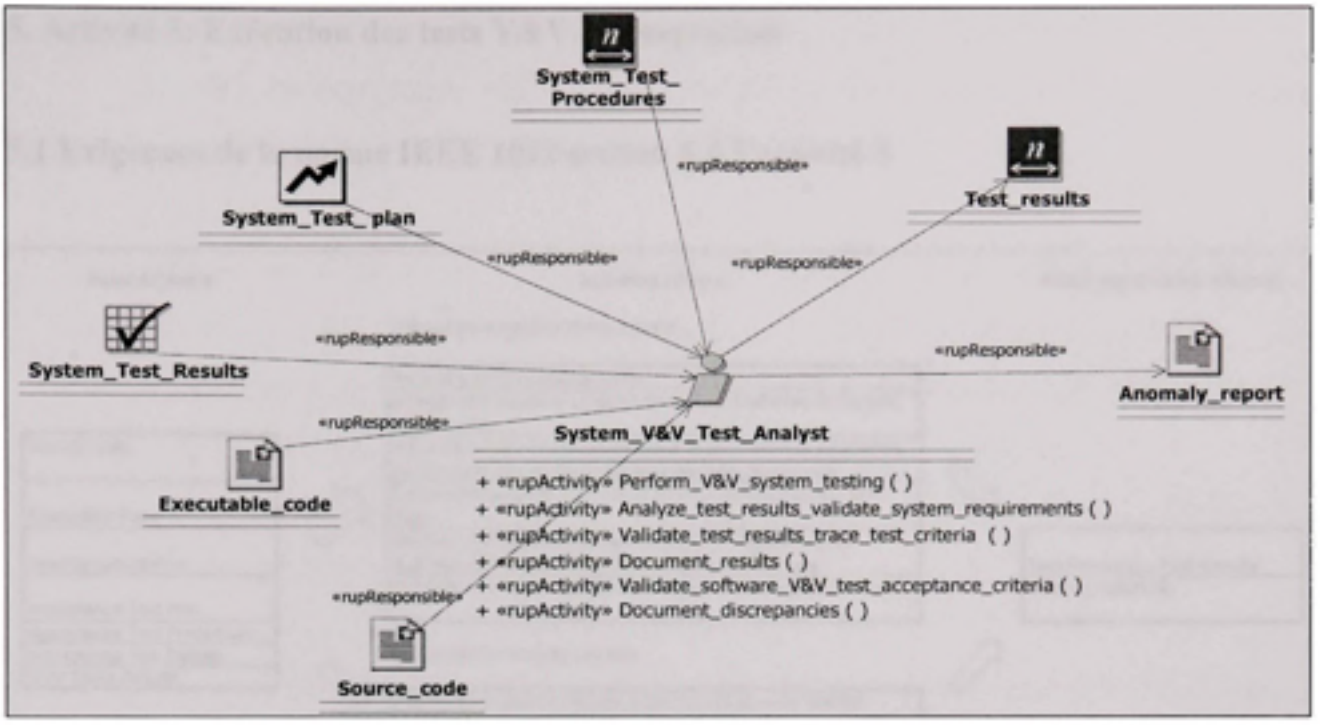

Figure II.9 ***System V&V Test Analyst Role (IEEE 1012 and RUP).***

5. Activité 5: Exécution des tests V&V d'acceptation

5.1 Exigences de la norme IEEE 1012 section 5.4.5 activité 5

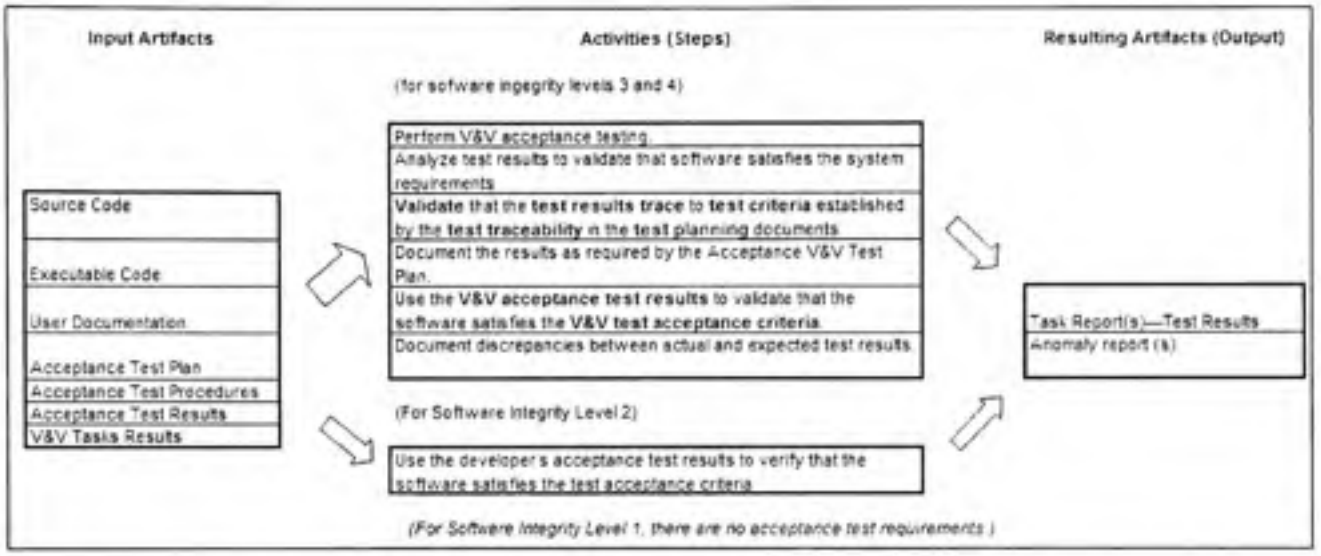

Figure II.10 ***Norme IEEE 1012 section 5.4.5 activité 3.***

5.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 5 avec les pratiques de RUP

La norme IEEE 1012 décrit comme entrées plusieurs artefacts produits dans le cycle de vie du logiciel, à savoir : Code source, code exécutable, plan de test du système, procédures des tests du système, résultats des tests du système.

Dans le « Test Workflow » ou dans un autre workflow spécifique pour la V&V (ex. « V&V Test Workflow »), il faut créer un « workflow detail » appelé « Acceptance V&V test execution». Ce workflow doit prendre les suivantes caractéristiques :

a. Si le niveau de criticité du logiciel est 3 ou 4, il faut exécuter les activités suivantes :
    1. Perform V&V acceptance testing.
    2. Analyze test results to validate that software satisfies the system requirements.
    3. Validate that the test results trace to test criteria established by the test traceability in the test planning documents.

4. Document the results as required by the Acceptance V&V Test Plan.
5. Use the acceptance V&V test results to validate that the software satisfies the V&V test acceptance criteria.

b. Si le niveau de criticité est de 2 ou 1, le « acquirer's acceptance test results» en concordance avec RUP, est suffisant pour combler les exigences de la norme IEEE 1012. Par contre il faut s'assurer d'exécuter une vérification des artefacts du développeur selon les exigences décrites pour la criticité du logiciel niveau 1 et 2, à savoir :
   1. Use the acquirer's acceptance test results to verify that the software satisfies the test acceptance criteria.

L'activité du RUP qui pourra combler les exigences du logiciel avec une criticité niveau 2 est:

- « Workflow Detail: Manage Acceptance Test » et l'activité « Activity: Manage Acceptance Test» de la discipline « Deployment», rôle responsable:: « Deployment Manager ».

Les livrables de ce workflow sont :

- Test Results.
- Anomaly report.

Le nouveau « workflow » à incorporer dans le RUP sera utilisé principalement pour la construction des logiciels avec une criticité niveau 3 ou 4. La V&V doit être exécutée soit par un analyste de V&V (dans le cas d'une V&V indépendante), ou bien par le rôle « Tester » du RUP, voir figure II.11.

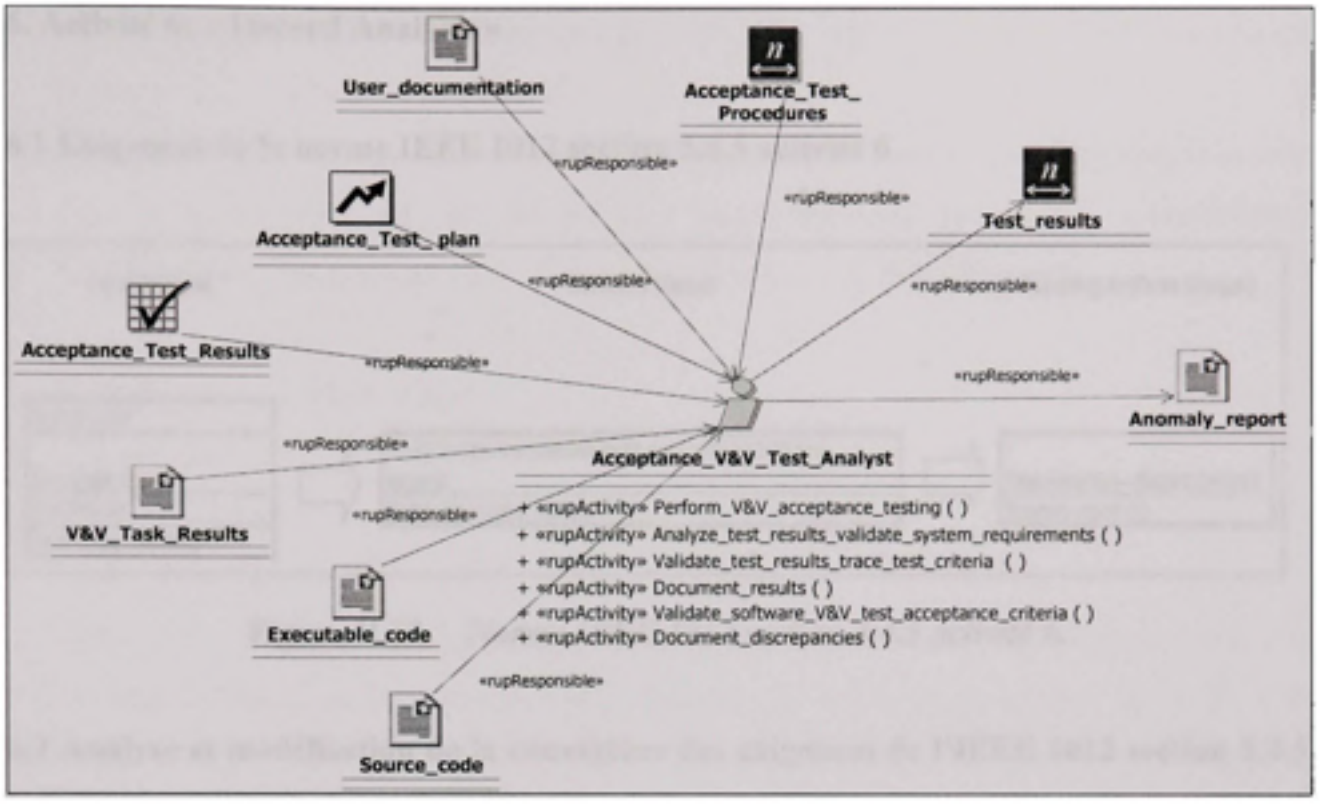

Figure II.11 Acceptance *V&V Test Analyst Role (IEEE 1012 and RUP).*

6. Activité 6: « Hazard Analysis »

6.1 Exigences de la norme IEEE 1012 section 5.4.5 activité 6

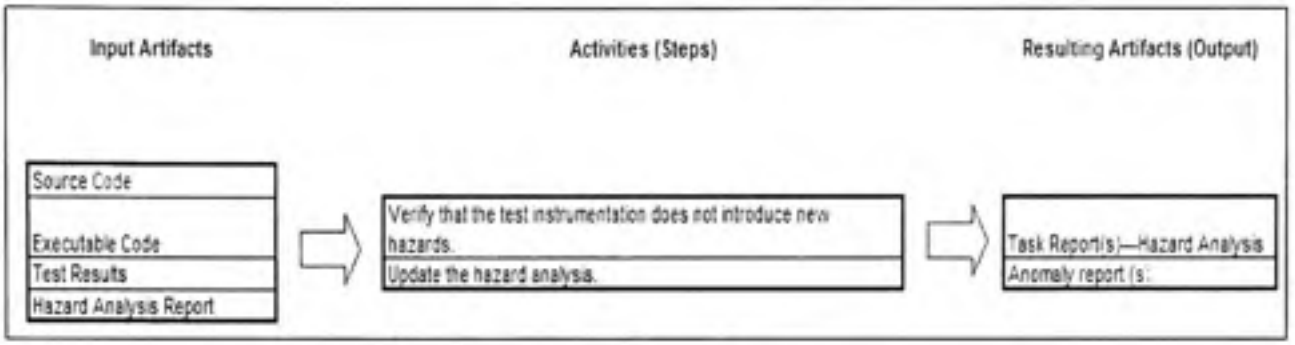

Figure II.12 ***Norme IEEE 1012 section 5.4.5 activité 6.***

6.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 6avec les pratiques de RUP

La norme IEEE 1012 décrit comme entrées plusieurs artefacts produits dans le cycle de vie du logiciel, à savoir : Code source, code exécutable, résultats des tests, Hazard Analysis Report.

Il est clair que pour cette activité la norme IEEE 1012 n'exige pas des activités spécifiques associées à un niveau particulier de criticité du logiciel. Alors, quelques pratiques décrites dans le RUP peuvent remplir les exigences de l'IEEE 1012 pour cette activité.

Les exigences demandées par la norme IEEE 1012 pourraient être couvertes par certaines activités du RUP. Il faut toutefois lors de l'exécution des activités dans le RUP, respecter les activités suivantes de la norme IEEE 1012, à savoir :

a. Verify that the test instrumentation does not introduce new hazards.
b. Update the hazard analysis.

L'activité du RUP qui pourra combler les exigences du logiciel avec une criticité niveau 2 est:

- « Workflow Detail: Test and Evaluate» et l'activité « Activity: Analyze Test Failuret» de la discipline « Test», rôle responsable: « Tester»
- « Workflow Detail: Analyze the Problem» et l'activité « Activity: Develop Requirements Management Plan» de la discipline « Requirements», la guide : « Guidelines: Requirements Management Plan », rôle responsable:: « System Analyst»

Les livrables de ce workflow sont :

- Hazard Analysis.
- Anomaly report.

Les activités de V&V doivent être exécutées soit par un analyste de V&V (dans le cas d'une V&V indépendante), ou bien par le rôle « Tester », ou « System Analyst » du RUP, voir figure II.13.

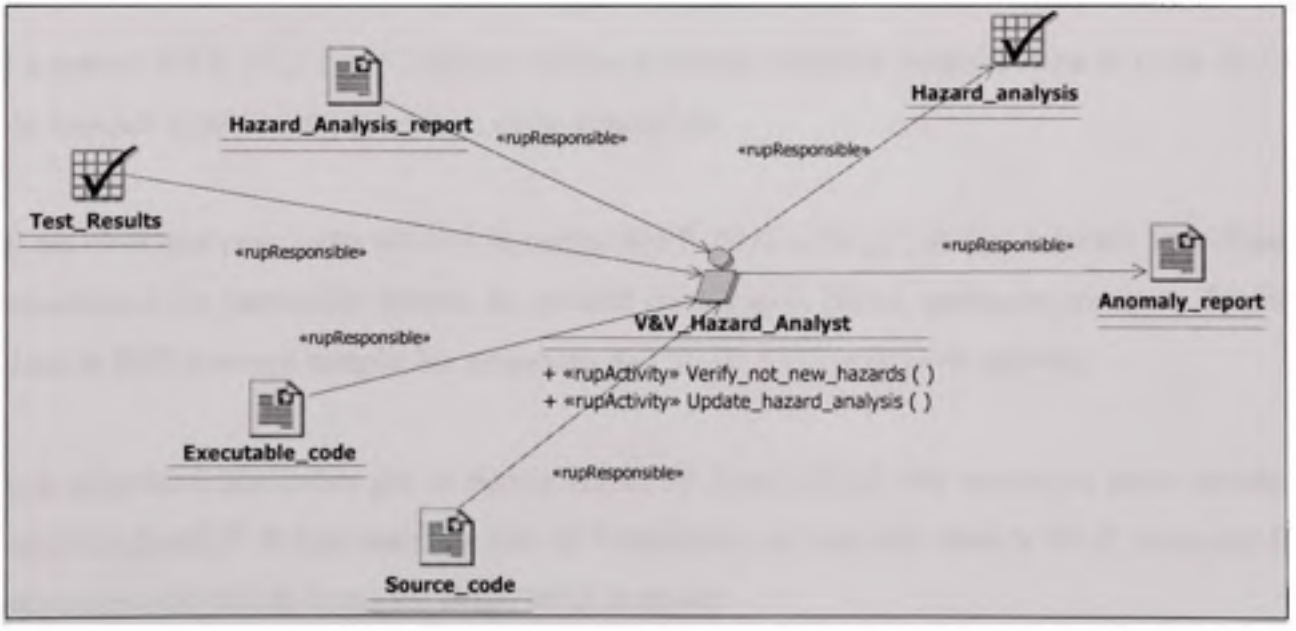

Figure II.13 V&V *Hazard Analyst Role (IEEE 1012 and RUP).*

7. Activité 7: Analyse de la sécurité

7.1 Exigences de la norme IEEE 1012 section 5.4.5 activité 7

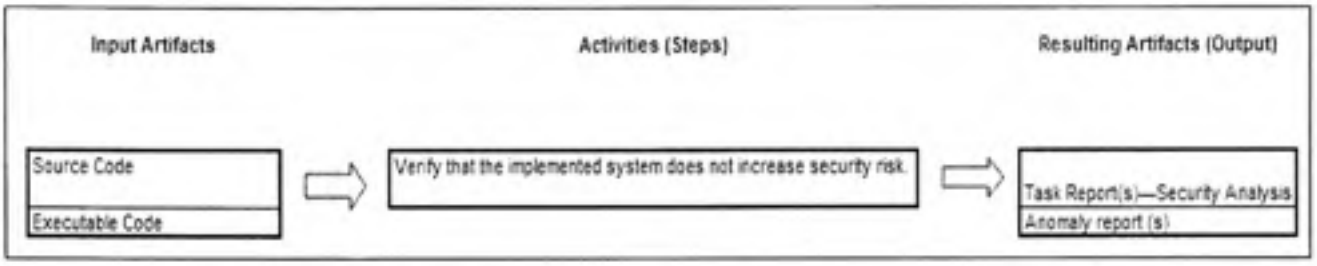

Figure II.14 *Norme IEEE 1012 section 5.4.5 activité 7.*

7.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 7, avec les pratiques de RUP

La norme IEEE 1012 décrit comme entrées plusieurs artefacts produits dans le cycle de vie du logiciel, à savoir : Code source, code exécutable.

Il est clair que pour cette activité la norme IEEE 1012 n'exige pas des activités spécifiques associées à un particulier niveau de criticité du logiciel. Alors, quelques pratiques décrites dans le RUP peuvent remplir les exigences de l'IEEE 1012 pour cette activité.

Les exigences demandes par la norme IEEE 1012 pourraient être couvertes pour certaines activités du RUP. Il faut toutefois lors de l'exécution des activités dans le RUP, respecter les suivantes activités de la norme IEEE 1012, à savoir :

- Verify that the implemented system does not increase security risk.

- L'activité du RUP qui pourra combler les exigences du logiciel avec une criticité niveau 2 est:

- « Workflow Detail: Verify Test Approach» et l'activité « Activity: Define Testability Elements», discipline « Test», rôle responsable: « Tester».

Autres références à la vérification de la sécurité sont incluses :

- « Workflow Detail: Verify Test Approach» et l'activité «Activity: Implement Test», discipline « Test», rôle responsable: « Tester ».
- « Workflow Detail: Identify Business Processes » et l'activité « Activity: Find Business Actors and Use Cases», discipline « Test», rôle responsable: « Business-Process Analyst».

Les livrables de ce workflow sont :

- Security Analysis.
- Anomaly report.

Les activités de V&V doivent être exécutées soit par un analyste de V&V (dans le cas d'une V&V indépendante), ou bien par le rôle « Tester », ou « Business-Process Analyst » du RUP, voir figure II.15.

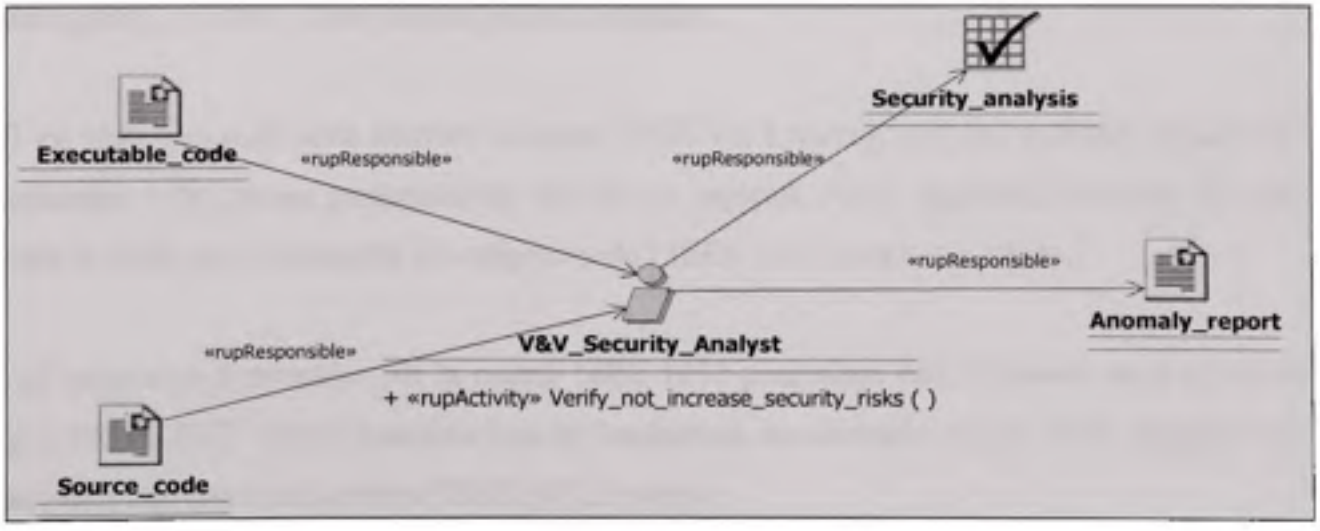

Figure II.15 ***V&V Security Analyst Role (IEEE 1012 and RUP).***

8 Activité 8: Analyse des risques

8.1 Exigences de la norme IEEE 1012 section 5.4.5 activité 8 par rapport à l'analyse des risques

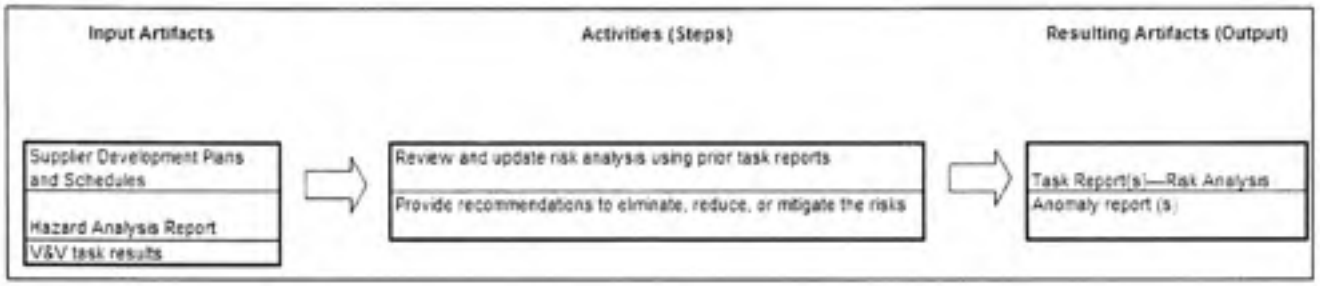

Figure II.16 ***IEEE 1012 section 5.4.5 activité 8.***

8.2 Analyse et modélisation de la couverture des exigences de l'IEEE 1012 section 5.4.5 activité 8, avec les pratiques de RUP

La norme IEEE 1012 décrit comme entrées plusieurs artefacts produits dans le cycle de vie du logiciel, à savoir : Code source, code exécutable.

Il est clair que pour cette activité la norme IEEE 1012 n'exige pas des activités spécifiques associées à un niveau particulier de criticité du logiciel. Alors, quelques pratiques décrites dans le RUP peuvent remplir les exigences de l'IEEE 1012 pour cette activité.

Les exigences demandées par la norme IEEE 1012 pourraient être couvertes pour certaines activités du RUP. Il faut toutefois lors de l'exécution des activités dans le RUP, respecter les activités suivantes de la norme IEEE 1012, à savoir :

a. Review and update risk analysis using prior task reports.
b. Provide recommendations to eliminate, reduce, or mitigate the risks.

L'activité du RUP qui pourra combler les exigences du logiciel avec une criticité niveau 2 est:

- « Workflow Detail: Evaluate Project Scope and Risk », l'activité « Activity: Identify and Assess Risks», discipline « Project Management», rôle responsable: « Project Manager».

Autres références à l'analyse de risques:

- « Workflow Detail: Define Evaluation Mission», l'activité «Activity: Identify Test Motivators», Step « Determine quality risks », discipline « Test», rôle responsable: « Test Manager»

Les livrables de ce workflow sont :

- Security Analysis.
- Anomaly report.

Les activités de V&V doivent être exécutées soit par un analyste de V&V (dans le cas d'une V&V indépendante), ou bien par le rôle « Tester », ou « Project Manager » du RUP, voir figure II.17.

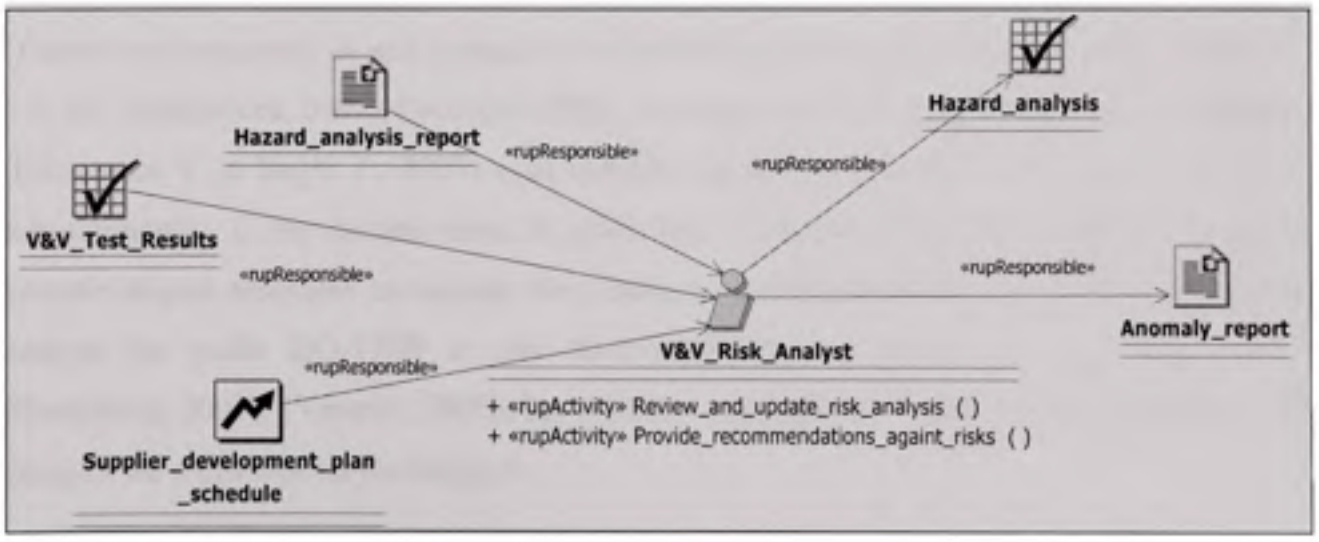

Figure II.17 *V&V Risk Analyst Role (IEEE 1012 and RUP).*

ANNEXE III

ANALYSE ET CONCEPTION DES PRATIQUES DE V&V POUR LE PROCESSUS DE DESIGN DU LOGICIEL CRITIQUE

Introduction

Cette Annexe traite des pratiques (exemple : exécution de tâches, rôles, responsabilités, et artefacts en général) qui doivent être exécutées en vue de vérifier les résultats du processus de design du logiciel critique. Ce processus est focalisé sur le test des aspects du code source. Les objectifs de ce processus font référence au tableau A-4 du guide DO-178B, voir annexe VII, tableau VII.2.

Nous avons discuté dans le chapitre 6 que les « High Level Requirements - HLR» sont décrits dans le guide DO-178B comme les caractéristiques du système allouées au logiciel.

D'après les documents de nos partenaires industriels que nous avons discutés dans le chapitre 6 (CMC Electronics, 2007), (Verocel, 2005), ainsi que dans le livre de Hilderman and Baghai (Hilderman V. et Baghi T., 2007) nous considérons que les exigences de type « Low Level Requirements - LLR» décrites dans le guide DO-178B, pourraient être comblées avec les caractéristiques associées au concept du « Design » mentionné dans l'OpenUP. Après d'une analyse du guide DO-178B et des documents de nos partenaires industriels (CMC Electronics, 2007), (Verocel, 2005), les « LLR » sont appelles aussi « les spécifications du design » ou « des aspects du design ».

Nous avons également constaté, après l'analyse du guide DO-178B et des documents de nos partenaires industriels (CMC Electronics, 2007), (Verocel, 2005) que le concept d'architecture du logiciel selon le guide DO-178B est semblable à celui d'OpenUP.

Les documents de nos partenaires (CMC Electronics, 2007), (Verocel, 2005) expliquent que:

- L'architecture du logiciel « **Software Architecture** » identifie les composants, organise les entités du design (ex. tâches, librairies, etc.), ainsi que leurs interactions.
- Les exigences de haut niveau ou « **High Level Requirements (HLR)** » identifient les exigences du système alloués au logiciel (ou STRATS).
- Les exigences de bas niveau ou « **Low Level Requirements (LLR)** » spécifient des fonctions et comment ces fonctions seront implémentées.

Les prochains tableaux montreront entre autres : l'analyse des objectifs du guide DO-178B versus l'OpenUP, ainsi que la proposition des pratiques de V&V qui constitueront les fondements de la conception du référentiel de V&V.

Tableau III.1

Analyse des résultats de vérification du design : Objectif 01

Définition de l'objectif 01:

"Low-level requirements **comply** with high-level requirements"

Description de l'objectif 01:

"The objective is to ensure that the software low-level requirements **satisfy** the software high-level requirements and that derived requirements and the design basis for their existence are **correctly** defined"

Référence dans le guide DO-178B:

6.3.2a

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification (point de vue analyse/revue/test) de la **conformité** entre les exigences de **haut niveau du logiciel** et **les exigences de bas niveau**. Les exigences dérivées (ce résultat des processus du développement) doivent être aussi bien justifiées.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - En référence à la tâche : « Design the solution » de la discipline « Development ». Cette tâche devrait demander obligatoirement :

     - Le « Step 1: Understand requirement details». Le « Step » fait référence d'un côté d'évaluer les « uses cases » et « supporting requirements specifications », ainsi que clarifier « Ambigous our Missing Information » qui pourrait être détecté dans l'évaluation.

     - Le « Step 8 : Evaluate the design». Le « Step » fait référence à l'utilisation de plusieurs

artefacts type check-lists pour évaluer le design. Pour remplir l'objectif du DO-178B il faut obligatoirement exécuter les artefacts suivants : o La Check-list « Design » o Le Guideline « Analyze the Design » o Le Guideline « Evolve the Design »

Tableau III.2

Analyse des résultats de vérification du design : Objectif 02

Définition de l'objectif 02:
"Low-level requirements are **accurate and consistent**"
Description de l'objectif 02:
"The objective is to ensure that each low-level requirement is **accurate and unambiguous** and that the low-level requirements **do not conflict** with each other"
Référence dans le guide DO-178B:
6.3.2b
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de l'exactitude des exigences de bas niveau. Il parle aussi de la non-ambigüité et du non conflits entre les exigences de bas niveau. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V :
1. Au niveau du rôle: **Selon le type de V&V à implanter dans le projet:** o **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : o **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.). **Note :** Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. Au niveau des disciplines, tâches et des artefacts:

- L'exécution de la tâche : « Design the solution » de la discipline « Development ». Cette tâche devrait d'exécuter obligatoirement :
  - Le « Step 1: Understand requirement details». Le « Step » fait référence d'un côté évaluer les « uses cases » et « supporting requirements specifications » afin de bien interpréter les exigences dans le design. Le « Step » fait aussi mention à l'utilisation de l'artefact suivant :
    - Le Guideline « Analyze the Design », pour bien analyser le design dans le contexte de la non ambiguïté et précision. Cet artefact devrait être utilisé obligatoirement.
  - Le « Step 8 : Evaluate the design» pourrait également être utilisé car il fait référence à l'utilisation de plusieurs artefacts afin de vérifier la qualité et la précision du design. Pour remplir l'objectif du DO-178B il faut obligatoirement exécuter les artefacts suivants :
    - La Check-list « Design »
    - Le Guideline « Evolve the Design »

Tableau III.3

Analyse des résultats de vérification du design : Objectif 03

Définition de l'objectif 03:
"Low-level requirements are **compatible** with target computer."
Description de l'objectif 03:
"The objective is to ensure that **no conflicts** exist between the software requirements and the hardware/software features of the target computer, especially, the use of resources (such as bus loading), system response times, and input/output hardware"
Référence dans le guide DO-178B:
6.3.2c
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification qu'il n'y a pas des conflits entre les exigences les exigences du logiciel et les caractéristiques du matériel, logiciel de l'ordinateur cible. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la

section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendant en dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - L'exécution de la tâche : « Find and Outline Requirements » de la discipline « Requirements». Cette tâche devrait d'exécuter obligatoirement :
     - L'exécution du « Step 7: Have external interfaces that must be considered in the next iteration been identified? », de la Check-list « Supporting Requirements ». Le « Step » inclut la vérification de l'interaction du logiciel avec le matériel, logiciel du système.
   - L'exécution de la tâche : « Detail Requirements » de la discipline « Requirements». Cette tâche devrait d'exécuter obligatoirement :
     - L'exécution du « Step 7: Is the requirement feasible? », de la Check-list « Quality of Good Requirements ». Le « Step » inclut des recommandations pour déterminer si l'exigence est faisable avec la technologie actuelle.
   - L'exécution de la tâche : « Define Vision » de la discipline « Requirements». Cette tâche devrait d'exécuter obligatoirement :
     - L'exécution des plusieurs « Steps» de la Check-list « Vision» afin de combler les exigences de non conflits entre le matériel, logiciel de l'ordinateur cible : ,
       - Does everyone agree on the definition of the system boundaries?
       - Have you sufficiently explored constraints to put on the system?
       - Are the features consistent with constraints that you've identified?

     Observation : Il faudrait faire attention lors de l'exécution de cette tâche de vérifier l'absence des conflits entre les exigences du logiciel et les caractéristiques du matériel/logiciel de l'ordinateur cible, particulièrement dans l'utilisation des ressources telles que « bus loading », « system response times », « input/output hardware ».

Tableau III.4

Analyse des résultats de vérification du design : Objectif 04

Définition de l'objectif 04:

"Low-level requirements are **verifiable.**"

Description de l'objectif 04:

"The objective is to ensure that each low-level requirement **can be verified**"

Référence dans le guide DO-178B:

6.3.2d

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de capacité à être vérifié des exigences de bas niveau en général.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   Nous mentionnerons une tâche qui exécuté de façon obligatoire devrait combler les objectifs du guide DO-178B.

   - La tâche : « Design the solution » de la discipline « Development ». Cette tâche devrait d'exécuter obligatoirement :
     - Le « Step 5 », « Step 6 » de la Check-list « Quality of Good Requirements ». Les « Steps » incluent des recommandations pour déterminer la vérifiabilité et la traçabilité des exigences.

Tableau III.5

Analyse des résultats de vérification du design : Objectif 05

Définition de l'objectif 05:
"Low-level requirements **conform** to standards"
Description de l'objectif 05:
"The objective is to ensure that the Software Design Standards **were followed** during the software design process, and that deviations from the standards **are justified**"
Référence dans le guide DO-178B:
6.3.2e
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la conformité du design aux standards, principalement au « Software Design Standards », et si les déviations ont été justifiées. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V : **1. Au niveau du rôle:** **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : • **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.). **Note :** Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2. **2. Au niveau des disciplines, tâches et des artefacts:** Nous mentionnerons une tâche qui exécuté de façon obligatoire devrait combler les objectifs du guide DO-178B. • L'exécution de la tâche : « Design the solution » de la discipline « Development ». Cette tâche devrait d'exécuter obligatoirement : ▪ Le « Step 2 » de la Check-list « Design». Le « Steps » inclut les vérifications si le standard du design a été suivi.

Tableau III.6

Analyse des résultats de vérification du design : Objectif 06

Définition de l'objectif 06:
"Low-level requirements are **traceable** to high-level requirements"
Description de l'objectif 06:
"The objective is to ensure that the high-level requirements and derived requirements **were developed** into the low-level requirements"
Référence dans le guide DO-178B:
6.3.2f
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification qu'exigences de haut niveau et des exigences dérivés ont été développés dans les exigences de bas niveau. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V :
1. Au niveau du rôle: **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : • **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.). **Note :** Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2. **2. Au niveau des disciplines, tâches et des artefacts:** • La tâche : « Design the solution » de la discipline « Development » (ou bien directement à la discipline « Development »). Cette tâche devrait d'exécuter obligatoirement : o Le « Step 4 : Is the design traceable? » de la check-list « Design » du guideline « Analyze the Design » ou « Evolve le design ».

Tableau III.7

Analyse des résultats de vérification du design : Objectif 07

Définition de l'objectif 07:
"Algorithms are **accurate**"
Description de l'objectif 07:
"The objective is to ensure the **accuracy and behavior** of the proposed algorithms, especially in the area of discontinuities"
Référence dans le guide DO-178B:
6.3.2g
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la précision et comportement des algorithmes particulièrement dans l'area de discontinuités. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V : **1. Au niveau du rôle:** **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : • **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.). **Note :** Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2. **2. Au niveau des disciplines, tâches et des artefacts:** • La tâche : « Design the solution » de la discipline « Development ». Cette tâche devrait d'exécuter obligatoirement : ◦ Le « Step 6 Design internals» de la tâche « Design the solution ». Le « Step » fait mention à la vérification des algorithmes et à l'évaluation de son comportement pendant son cycle de vie. Il faudrait juste mettre l'emphase dans les discontinuités.

Tableau III.8

Analyse des résultats de vérification du design : Objectif 08

Définition de l'objectif 08:

"Software architecture is **compatible** with high-level requirements"

Description de l'objectif 08:

"The objective is to ensure that the software architecture does **not conflict** with the high-level requirements, especially functions that ensure system integrity, for example, partitioning schemes"

Référence dans le guide DO-178B:

6.3.3a

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la compatibilité de l'architecture des exigences de haut niveau.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou architecte** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - L'exécution de la tâche : « Refine the Architecture» de la discipline « Architecture». Cette tâche devrait d'exécuter obligatoirement :

     - Le « Step 6: Validate the architecture » de la tâche « Refine the Architecture». Le « Step » fait mention au support aux exigences de haut niveau.

▪ Le « Step 1: Is the overall structure of the architecture clear? » de la check-list « Architecture Notebook». Le « Step » fait mention à la vérification que toutes les exigences ont été traitées dans les « Architectural Mechanisms » ▪ Le « Step 2: Have the supporting requirements been adequately addressed?» de la check-list « Architecture Notebook». Le « Step » fait mention à la vérification que les exigences on été prises en considération dans l'architecture.

Tableau III.9

Analyse des résultats de vérification du design : Objectif 09

Définition de l'objectif 09:
"Software architecture is **consistent**"
Description de l'objectif 09:
"The objective is to ensure that a **correct relationship** exists between the components of the software architecture. This relationship exists via data flow and control flow"
Référence dans le guide DO-178B:
6.3.3b
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite sur la vérification que l'architecture du logiciel soit consistante (pas des conflits dans le cycle de vie). **Analyse de l'objectif du point de vue d'OpenUP :** Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système. Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V :
1. **Au niveau du rôle:** **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : • **L'utilisation des rôles d'analyste ou architecte** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles,

contraintes du temps, des coûts, etc.).

Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

- L'exécution de la tâche : « Outline the Architecture» de la discipline « Architecture». Cette tâche devrait d'exécuter obligatoirement :
  - Le « Step 9: Verify architectural consistency ». Le « Step » fait mention à la consistance de l'architecture vers les exigences ainsi que leur clarté, et complétude entre autres. Il faudra mettre une emphase dans l'utilisation des flow de control des données.
- L'exécution de la tâche : « Refine the Architecture» de la discipline « Architecture». Cette tâche devrait d'exécuter obligatoirement :
  - Le « Step 1: Is the overall structure of the architecture clear? » de la check-list « Architecture Notebook». Le « Step » fait également mention à la consistance de l'architecture vers les exigences ainsi que leur clarté, et complétude entre autres. Il faudra mettre une emphase dans l'utilisation des flux de contrôle des données.

Tableau III.10

Analyse des résultats de vérification du design : Objectif 10

Definition de l'objectif 10:
"Software architecture is **compatible** with target computer"
Description de l'objectif 10:
"The objective is to ensure that **no conflicts exist**, especially initialization, asynchronous operation, synchronization and interrupts, between the software architecture and the hardware/software features of the target computer"
Référence dans le guide DO-178B:
6.3.3c
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification que l'architecture du logiciel n'as pas des conflits avec le matériel/logiciel de l'ordinateur cible.

Analyse de l'objectif du point de vue d'OpenUP :
Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système. Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V :
1. Au niveau du rôle: **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : • **L'utilisation des rôles d'analyste ou architecte** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.). **Note :** Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2. **2. Au niveau des disciplines, tâches et des artefacts:** • L'exécution de la tâche : « Refine the Architecture» de la discipline « Architecture». Cette tâche devrait d'exécuter obligatoirement : • Le « Step 4: Map the software to the hardware ». Le « Step » fait mention à la identification et l'évaluation de l'environnement cible où l'architecture devrait opérer. **Observation :** Il faudrait faire attention lors de l'exécution de cette tâche, de vérifier les non conflits particulièrement dans les contextes suivants: « initialization, asynchronous operation, synchronization and interrupts »

Tableau III.11

Analyse des résultats de vérification du design : Objectif 11

Définition de l'objectif 11:
"Software architecture is **verifiable**"
Description de l'objectif 11:
"The objective is to ensure that the software architecture **can be verified**, for example, there are no unbounded recursive algorithms"

Référence dans le guide DO-178B:

6.3.3d

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de capacité à être vérifié de l'architecture du logiciel en général.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou architecte** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - L'exécution de la tâche : « Refine the Architecture» de la discipline « Architecture». Cette tâche devrait d'exécuter obligatoirement :

     - Le « Step 4: Define development architecture and test architecture » de la check-list « Architecture Notebook». Le « Step » fait mention à la vérification si l'architecture du développement et du test a été développé.

   - En générale les activités de vérification de l'architecture sont décrites dans les « Steps » des tâches « Refine the Architecture» et « Outiline Architecture» de la discipline « Architecture».
     Par exemple la tâche « Refine the Architecture» fait mention à la revue de:
     - Identify architectural goals
     - Identify architecturally significant requirements
     - Identify constraints on the architecture
     - Verify architectural consistency

     Par exemple la tâche « Outiline the Architecture» fait mention à la revue de:
     - Refine architectural mechanisms
     - Define development architecture and test architecture
     - Validate the architecture
     - Communicate decisions

     Observation: Dans le contexte des algorithmes il faudrait mettre l'emphase de vérifier sur « no unbounded recursive algorithms »

Tableau III.12

Analyse des résultats de vérification du design : Objectif 12

Définition de l'objectif 12:

"Software architecture is **conform** to standards"

Description de l'objectif 12:

"The objective is to ensure that the Software Design **Standards were followed** during the software design process and that **deviations to the standards are justified**, especially complexity restrictions and design constructs that would **not comply** with the system safety objectives"

Référence dans le guide DO-178B:

6.3.3e

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la **conformité** entre l'architecture du logiciel et les standards, comme par exemple le Software Design Standard. Toute déviation des standards doive être justifiée.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou architecte** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - L'exécution de la tâche : « Outline the Architecture» de la discipline « Architecture». Cette tâche devrait d'exécuter obligatoirement :

     - Le « Step 5: In general, does the architecture seem sensible? » de la check-list « Architecture Notebook». Dans le « Step » existe une question expresse si le guidelines du design ont été suivis et si l'architecture est cohérent avec les décisions documentés.

Observation : Lors de l'exécution de cette tâche, il est important de mettre l'emphase sur la conformité aux standards du design, ainsi que sur la justification à toute déviation de ces standards.

Tableau III.13

Analyse des résultats de vérification du design : Objectif 13

Définition de l'objectif 13:
"Software partitioning **integrity** is confirmed"
Description de l'objectif 13:
"The objective is to ensure that partitioning breaches **are prevented or isolated**"
Référence dans le guide DO-178B:
6.3.3f
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification que les possibles « trous » dans le « partitioning » ont été prévus ou isolés Voici le concept de « Partitioning » selon le guide DO-178B : **Partitioning** Partitioning is a technique for providing isolation between functionally independent software components to contain and or isolate faults and potentially reduce the effort of the software verification process. If protection by partitioning is provided, the software level for each partitioned component may be determined using the most severe failure condition category associated with that component. ***Guidance for partitioning includes:*** a. These aspects of the system should be considered when designing partitioning protection to determine their potential for violating that protection: (1) Hardware resources: processors, memory devices, I/O devices, interrupts, and timers. (2) Control coupling: vulnerability to external access. (3) Data coupling: shared or overlaying data, including stacks and processor registers. (4) Failure modes of hardware devices associated with the protection mechanisms. b. The software life cycle processes should address the partitioning design considerations, including the extent and scope of interactions permitted between the partitioned components, and whether the protection is implemented by hardware or by a combination of hardware and software. c. If the partitioning protection involves software, then that software should be assigned the software level corresponding to the highest level of the partitioned software components. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Référence dans OpenUP:

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou architecte** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - L'exécution de la tâche : « Implement the solution » de la discipline « Desing». Cette tâche devrait d'exécuter obligatoirement :

     - Le « Step 11: Packages and Organization » de la check-list « Design». Dans le « Step » existe une question expresse si le page « partition » est logique et consistent.

   - L'exécution de la tâche : « Refine Architecture » de la discipline « Architecture». Cette tâche devrait d'exécuter obligatoirement :

     - Le « Step 1: Is the overall structure of the architecture clear? », point 5 «Is subsystem and package partitioning and layering logically consistent? » de la check-list « Architecture Notebook».

     Observation : Lors de l'exécution de cette tâche, Il est important mettre l'emphase sur le guideline pour le correct « partitioning » inclus dans la définition, à savoir :

     a. These aspects of the system should be considered when designing partitioning protection to determine their potential for violating that protection:
        (1) Hardware resources: processors, memory devices, I/O devices, interrupts, and timers.
        (2) Control coupling: vulnerability to external access.
        (3) Data coupling: shared or overlaying data, including stacks and processor registers.
        (4) Failure modes of hardware devices associated with the protection mechanisms.
     b. The software life cycle processes should address the partitioning design considerations, including the extent and scope of interactions permitted between the partitioned components, and whether the protection is implemented by hardware or by a combination of hardware and software.
     c. If the partitioning protection involves software, then that software should be assigned the software level corresponding to the highest level of the partitioned software components.

ANNEXE IV

ANALYSE ET CONCEPTION DES PRATIQUES DE V&V POUR LE PROCESSUS DE CODIFICATION ET INTÉGRATION DU LOGICIEL CRITIQUE

Introduction

Cette Annexe traite sur les pratiques (exemple : exécution de tâches, rôles, responsabilités, et artefacts en général) qui doivent être exécutées en vue de vérifier les résultats du processus codification et d'intégration du logiciel critique. Ce processus est focalisé sur le test des aspects du code source. Les objectifs de ce processus font référence au tableau A-5 du guide DO-178B, voir annexe VII, tableau VII.3.

D'après les documents de nos partenaires industriels que nous avons discutés dans le chapitre 6, nous allons considérer que les exigences du « Low Level Requirements » décrites dans le guide DO-178B, sont comblées avec les caractéristiques associées au concept du « Design » mentionné dans l'OpenUP.

Les prochains tableaux montreront entre autres l'analyse des objectifs du guide DO-178B versus l'OpenUP, ainsi que la proposition des pratiques de V&V qui constitueront les fondements de la conception du référentiel de V&V.

Tableau IV.1

Analyse des résultats de vérif. de codification et intégration : Obj. 01

Définition de l'objectif 01:
"Source code **complies** with low-level requirements"
Description de l'objectif 01:
"The objective is to ensure that the Source Code is **accurate and complete** with respect to the software low-level requirements, and that no Source Code **implements an undocumented** function"
Référence dans le guide DO-178B:
6.3.4a
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la **conformité** entre le code source et les exigences de bas niveau. Il faut vérifier que le code source est précise, et complète respect des exigences de bas niveau et qui le code n'implémente pas fonctions non documentés dans les exigences. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V : **1. Au niveau du rôle:** **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : • **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.). **Note :** Les rôles devront utiliser les disciplines, tâches, artefacts, produits (Work Products) proposés dans le point 2. **2. Au niveau des disciplines, tâches et des artefacts:** • En référence à la tâche : « Implement the solution » de la discipline « Development ». Cette tâche devrait demander obligatoirement: ◦ Le « Step 5: Evaluate the Implementation». Le « Step » fait référence à l'évaluation du code à fin de déterminer celui-ci rempli les fonctions. **Observation :** Une référence à vérifier le « Step 3 : Transform Design into Implementation» de la même tâche serait nécessaire afin de remplir les exigences, d'une façon plus précise, le guide DO-178B au sujet

de la conformité du code source et le LLR (Low Level Requirements).

- La discipline « Development » fait référence de la check-list « Implementation ». Cette Check-list devrait demander obligatoirement:
  - Le « Step 1: Does the implementation conform to the architecture and design? ». Le « Step » inclut des recommandations vérifier si le code source remplit les exigences du design (LLR).

Tableau IV.2

Analyse des résultats de vérif. de codification et intégration : Obj. 02

Définition de l'objectif 02:
"Source code **complies** with software architecture"
Description de l'objectif 02:
"The objective is to ensure that the Source Code **matches** the data flow and control flow defined in the software architecture"
Référence dans le guide DO-178B:
6.3.4b
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la **conformité** entre le code source et l'architecture du logiciel. Il faut vérifier que le code source coïncide avec le « data flow » et le « control flow » défini dans l'architecture du logiciel. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V :
1. Au niveau du rôle: **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien : • **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - La discipline « Development » fait référence de la check-list « Implementation ». Cette Check-list devrait demander obligatoirement:

     - L'exécution du « Step 1: Does the implementation conform to the architecture and design? ». Le « Step » inclut des recommandations vérifier si le code source rempli les exigences de l'architecture du logiciel.

     Observation : Il est important de mettre une emphase sur la vérification des aspects du « data flow » et les « control flow » dans le code source.

Tableau IV.3

Analyse des résultats de vérif. de codification et intégration : Obj. 03

Définition de l'objectif 03:
"Source code is **verifiable**"
Description de l'objectif 03:
"The objective is to ensure the Source Code does not contain statements and structures **that cannot be verified** and that the code **does not have to be altered** to test it."
Référence dans le guide DO-178B:
6.3.4c
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de capacité à être vérifié (testé, analysé, révisé) en générale du code source. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - La discipline « Development » fait référence de la check-list « Implementation ». Cette Check-list devrait demander obligatoirement:
     - Le « Step 2: Is the implementation testable? ». Le « Step » inclut des recommandations si le code source a répond à un comportement attendu.
     - Le « Step 3: Is the implementation correct?». Le « Step » inclut une référence si le code a passé les tests du développeur.
   - En référence à la tâche : « Implement the solution » de la discipline « Development ». Cette tâche devrait demander obligatoirement:
     - Le « Step 5: Evaluate the Implementation». Le « Step » fait référence à l'inspection du code, l'utilisation des outils pour vérifier les erreurs d'implémentation (ex. due à des compilateurs), entre autres.
   - En référence à la tâche : « Run Developper Tests» de la discipline « Development ». Cette tâche devrait demander obligatoirement:
     - Le « Step 1» à « Step 3», afin de vérifier que l'implémentation du code fonctionne comme a été spécifié. Les « Steps » sont :
       - Run Developer Tests
       - Evaluate test execution
       - Respond to test results

   Observation : Lors de l'exécution de ces tâches il faut faire attention aux « statements » et « structures » qui ne sont pas capables être vérifiées ou à des modifications du code pour faire les tests.

Tableau IV.4

Analyse des résultats de vérif. de codification et intégration : Obj. 04

Définition de l'objectif 04:
"Source code **conforms** to standards"

Description de l'objectif 04:

"The objective is to ensure that the Software Code **Standards were followed** during the development of the code, especially **complexity restrictions and code constraints** that would be **consistent with the system safety objectives**. Complexity includes **the degree of coupling** between software components, the nesting levels for control structures, and the complexity of logical or numeric expressions, **This analysis** also ensures that deviations to the **standards are justified**"

Référence dans le guide DO-178B:

6.3.4d

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la **conformité** entre le code source et les standards. Il faut vérifier que le code source est conforme aux contraints et restrictions de complexité qui sont imposés pour les objectifs de sûreté. Les déviations aux standards doivent être justifies

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - La discipline « Development » fait référence de la check-list « Implementation ». Cette Check-list devrait demander obligatoirement:
     - L'exécution du « Step 2: Is the implementation understandable? ». Le « Step » demande parmi autres la vérification si le code rempli les exigences de la documentation.
   - En référence à la tâche : « Implement the solution » de la discipline « Development ». Cette tâche devrait demander obligatoirement:
     - L'exécution du « Step 5: Evaluate the Implementation». Le « Step » fait référence par exemple à l'évaluation du code par rapport au excessive « coupling » et « circular dependencies ».
     - L'exécution du « Step 6 : Communicate significant decisions ». Le « Step » fait référence à la communication des changements dans le code et le besoin d'un « change request »si nécessaire.

   Observation : Lors de l'exécution de ces tâches, une référence explicite doit être incluse par rapport aux restrictions et contraintes de complexité du code (Complexité en termes de couplage, structures de contrôle, expressions numériques et logiques).

Tableau IV.5

Analyse des résultats de vérif. de codification et intégration : Obj. 05

Définition de l'objectif 05:

"Source code is **traceable** to low-level requirements"

Description de l'objectif 05:

"The objective is to ensure that the software low-level requirements **were developed** into Source Code"

Référence dans le guide DO-178B:

6.3.4e

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la **traçabilité** entre le code source et les exigences de bas niveau.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - La discipline « Development » fait référence de la check-list « Implementation ». Cette Check-list devrait demander obligatoirement:

     - Le « Step 1: Does the implementation conform to the architecture and design? ». Le « Step » inclut des recommandations vérifier si le code source rempli les exigences du design du logiciel.

   Observation : Il est important de mettre une emphase sur le besoin de créer une tâche explicit par rapport à la traçabilité entre le code source et les LLR.

Tableau IV.6

Analyse des résultats de vérif. de codification et intégration : Obj. 06

Définition de l'objectif 06:

"Source code is **accurate and consistent**"

Description de l'objectif 06:

"The objective is to **determine the correctness and consistency** of the Source Code, including ***stack usage, fixed point arithmetic overflow*** *and* ***resolution, resource contention, worst-case execution timing, exception handling, use of uninitialized variables or constants, unused variables or constants, and data corruption due to task or interrupt conflicts"***

Référence dans le guide DO-178B:

6.3.4f

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la détermination de l'exactitude (précision) et consistance (pas des conflits) du code source.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Référence dans OpenUP:

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - En référence à la tâche : « Implement Developper Tests » de la discipline « Development ». Cette tâche devrait demander obligatoirement :
     - Le « Step 1» à le « Step 7», à savoir :
       - Refine scope and identify the test(s)
       - Write the test setup

- Define the expected results
- Write the test logic
- Define the test response
- Write clean-up code
- Test the test

- **Observation :** Les « Steps » précédents devront être adaptés afin de remplir les exigences décrites pour l'objectif 06 du DO-178B, à savoir :
  - Stack usage,
  - Fixed point arithmetic overflow and resolution,
  - Resource contention,
  - Worst-case execution timing,
  - Exception handling,
  - Use of uninitialized variables or constants,
  - Unused variables or constants,
  - Data corruption due to task or interrupt conflicts.

Tableau IV.7

Analyse des résultats de vérif. de codification et intégration : Obj. 07

Définition de l'objectif 07:
"Output of the software module integration process is **complete and correct**"
Description de l'objectif 07:
"The objective is to ensure that the results of the integration process are **complete and correct**. This could be performed by a detailed examination of the linking and loading data and memory map. The topics should include: a. Incorrect hardware addresses. b. Memory overlaps. c. Missing software components."
Référence dans le guide DO-178B:
6.3.5
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination de la complétude et de l'exactitude des résultats de l'intégration du logiciel. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou développeur** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**
   - En référence à la tâche : « Run Developper Tests» de la discipline « Development ». Cette tâche devrait demander obligatoirement:
     - Le « Step 1» à « Step 3», afin de vérifier que l'implémentation du code fonctionne comme a été spécifié. Les « Steps » sont :
       - Run Developer Tests
       - Evaluate test execution
       - Respond to test results

     Observation : Lors de l'exécution de cette tâche, une référence explicite doit être incluse par rapport au « the linking and loading data and memory map» et les sujets à couvrir doivent inclure :
       a. Incorrect hardware addresses.
       b. Memory overlaps.
       c. Missing software components

ANNEXE V

ANALYSE ET CONCEPTION DES PRATIQUES DE V&V POUR LE PROCESSUS D'INTÉGRATION DU LOGICIEL CRITIQUE

Introduction

Cette Annexe traite des pratiques (exemple : exécution de tâches, rôles, responsabilités, et artefacts en général) qui doivent être exécutées en vue de vérifier les résultats du processus d'intégration du logiciel critique. Ce processus est focalisé sur le test d'intégration du code exécutable. Les tests unitaires ou du développeur, basés sur les tests des morceaux du code pour certains fonctionnalités du logiciel, ne sont pas considérés dans ce processus. Les objectifs de ce processus font référence au tableau A-6 du guide DO-178B, voir annexe VII, , tableau VII.4.

Le code exécutable dans le cas d'OpenUP entre autres pourrait être traduit comme le développement d'un artefact « Build », malgré que cet artefact contienne plus d'éléments que juste le code exécutable. D'ailleurs il faut noter que ce processus est concentré sur la vérification par le biais du test, et non par l'analyse et la revue.

Les prochains tableaux montreront entre autres : l'analyse des objectifs du guide DO-178B versus l'OpenUP, ainsi que la proposition des pratiques de V&V qui constitueront les fondements de la conception du référentiel de V&V.

Tableau V.1

Analyse des résultats de Test d'intégration : Objectif 01

Définition de l'objectif 01:

"**Executable object** code **complies** with **high-level requirements**"

Description de l'objectif 01:

Objective 6.4.2.1:

The objective of **normal range test cases** is to **demonstrate** the ability of the software to **respond to normal inputs and conditions**. Normal range tests cases include:

a. Real and integer input variables should be exercised using valid equivalence classes and boundary values
b. For time-related functions, such as filters, integrators and delays, multiple iterations of the code should be performed to check the characteristics of the function in context
c. For state transitions, test cases should be developed to exercise the transitions possible during normal operation
d. For software requirements expressed by logic equations, the normal range test cases should verify the variable usage and the Boolean operators

Objective 6.4.3 :

Requirements-based testing methods consist of methods for requirements-based hardware/software integration testing. With the exception of hardware/software integration testing, these methods do not prescribe a specific test environment or strategy. Guidance includes:

a. Requirements-based Hardware/Software Integration Testing: **This testing method** should concentrate on **error sources associated with the software operating within the target computer environment, and on the high-level functionality**. The objective of requirements-based hardware/software integration testing is to ensure that the software in the target computer will satisfy the high-level requirements. Typical errors revealed by this testing method include:
   - Incorrect interrupt handling.
   - Failure to satisfy execution time requirements.
   - Incorrect software response to hardware transients or hardware failures, for example,
   - Start-up sequencing, transient input loads and input power transients.
   - Data bus and other resource contention problems, for example, memory mapping.
   - Inability of built-in test to detect failures.
   - Errors in hardware/software interfaces.
   - Incorrect behaviour of feedback loops.
   - Incorrect control of memory management hardware or other hardware devices under
   - Software control.
   - Stack overflow.
   - Incorrect operation of mechanism(s) used to confirm the correctness and Compatibility of field-loadable software.
   - Violations of software partitioning.

b. Requirements-Based Software Integration Testing: **This testing method** should concentrate on the inter-relationships between the **software requirements, and on the implementation of requirements by the software architecture**. The objective of requirements-based software

integration testing is to ensure that the software components interact correctly with each other and satisfy the software requirements and software architecture. This method may be performed by expanding the scope of requirements through successive integration of code components with a corresponding expansion of the scope of the test cases. Typical errors revealed by this testing method include:

- Incorrect initialization of variables and constants.
- Parameter passing errors.
- Data corruption, especially global data.
- Inadequate end-to-end numerical resolution.
- Incorrect sequencing of events and operations

c. Requirements-Based Low-Level Testing: **This testing** method should concentrate on demonstrating that **each software component complies with its low-level requirements**. The objective of requirements-based low-level testing is to ensure that the software components satisfy their low-level requirements. Typical errors revealed by this testing method include:

- Failure of an algorithm to satisfy a software requirement.
- Incorrect loop operations.
- Incorrect logic decisions.
- Failure to process correctly legitimate combinations of input conditions.
- Incorrect responses to missing or corrupted input data.
- Incorrect handling of exceptions, such as arithmetic faults or violations of array limits.
- Incorrect computation sequence.
- Inadequate algorithm precision, accuracy or performance.

Référence dans le guide DO-178B:

6.4.2.1 et 6.4.3

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la détermination de la conformité du code exécutable et les exigences de haut niveau.

Il faut noter que les tests doivent être exécutés tel que décrits dans les objectifs de la DO-178B et en gardant l'objectif suivant « **Executable object** code **complies** with **high-level requirements** ». Autrement dit en gardant la conformité entre le code exécutable et les exigences de haut niveau.

Analyse de l'objectif du point de vue d'OpenUP :

Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système.
Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products)

proposés dans le point 2.

2. Au niveau des disciplines, tâches et des artefacts:

Il faut exécuter les tâches suivantes mais en respectant les observations consignés plus bas dans cette section.

- En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
  - Les « Step 1» à « Step 5»:
    - Review the requirements to be tested
    - Identify relevant Test Cases
    - Outline the Test Cases
    - Identify test data needs
    - Share and evaluate the Test Cases
- En référence à la tâche : « Implement Tests Scripts» de la discipline « Test». Cette tâche devrait demander obligatoirement:
  - Les « Step 1» à « Step 6»:
    - Select Test Cases to implement
    - Design the Test Script
    - Implement the executable Test Script
    - Define specific test data
    - Verify Test Script implementation
    - Share and evaluate Test Scripts
- En référence à la tâche : « Run Tests» de la discipline « Test». Cette tâche est adressée au niveau des tests de l'application logiciel. La tâche devrait demander obligatoirement:
  - Les « Step 1» à « Step 5:
    - Review work items completed in the build
    - Select Test Scripts
    - Execute Test Scripts against the build
    - Analyze and communicate test results
    - Provide feedback to the team

Observation 1: Les «Normal test cases» doivent satisfaire l'objectif 6.4.2.1, à savoir:

- Real and integer input variables should be exercised using valid equivalence classes and boundary values
- For time-related functions, such as filters, integrators and delays, multiple iterations of the code should be performed to check the characteristics of the function in context
- For state transitions, test cases should be developed to exercise the transitions possible during normal operation
- For software requirements expressed by logic equations, the normal range test cases should verify the variable usage and the Boolean operators

Observation 2: Les «Requirements-based Testing» doivent satisfaire les objectifs a, b et c du point 6.4.3, à savoir:

a. Testing should concentrate on error sources associated with the software operating within the target computer environment, and on the high-level functionality.
b. Testing should concentrate on the inter-relationships between the software requirements, and on the implementation of requirements by the software architecture.
c. Testing method should concentrate on demonstrating that each software component complies with its low-level requirements

Observation 3: Les «Requirements-based Testing» doivent garder en considération les types d'erreurs mentionnés dans les objectifs a, b et c du point 6.4.3 mentionné plus haut dans ce tableau.

Tableau V.2

Analyse des résultats de Test d'intégration : Objectif 02

Definition de l'objectif 02:

"**Executable** object code is **robust** with **high-level requirements**"

Description de l'objectif 02:

Objective 6.4.2.2:

The objective of **robustness test cases** is to demonstrate the **ability of the software** to respond **to abnormal inputs and conditions**. Robustness test cases include:

- Real and integer should be exercised using equivalence class selection of invalid values.
- System initialization should be exercised during abnormal conditions
- The possible failure modes of the incoming data should be determined, especially complex, digital data strings from an external system
- For loops where the loop count is a computed value, test cases should be developed to attempt to compute out-of-range loop count values, and thus demonstrate the robustness of the loop-related code.
- A check should be made to ensure that protection mechanisms for exceeded frame times respond correctly
- For time-related functions, such as filters, integrators and delays, test cases should be developed for arithmetic overflow protection mechanisms
- For state transitions, tests cases should be developed to provoke transitions that are not allowed by the software requirements.

Objective 6.4.3: (voir description dans le tableau 32, section Description de l'objectif 1)

Référence dans le guide DO-178B:

6.4.2.2 et 6.4.3

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la détermination de la robustesse du code exécutable et les exigences de haut niveau.

Il faut noter que les tests doivent être exécutés tel que décrits dans les objectifs de la DO-178B et en gardant l'objectif suivant « **Executable** object code is **robust** with **high-level requirements**». Autrement dit en gardant l'objectif que le code exécutable doit agir tel que prévu avec les exigences de haut niveau, dans les cas des conditions ou des données anormales.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet :

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts :**

   Il faut exécuter les tâches suivantes mais en respectant les observations consignés plus bas dans cette section.

   - En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 5»:
       - Review the requirements to be tested
       - Identify relevant Test Cases
       - Outline the Test Cases
       - Identify test data needs
       - Share and evaluate the Test Cases
   - En référence à la tâche : « Implement Tests Scripts» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 6»:
       - Select Test Cases to implement
       - Design the Test Script
       - Implement the executable Test Script
       - Define specific test data
       - Verify Test Script implementation
       - Share and evaluate Test Scripts
   - En référence à la tâche : « Run Tests» de la discipline « Test». Cette tâche est adressée au niveau des tests de l'application logiciel. La tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 5»:
       - Review work items completed in the build
       - Select Test Scripts
       - Execute Test Scripts against the build
       - Analyze and communicate test results
       - Provide feedback to the team

     Observation 1: Les «robustness test cases» doivent satisfaire l'objectif 6.4.2.2, à savoir:

       - Real and integer should be exercised using equivalence class selection of invalid values.
       - System initialization should be exercised during abnormal conditions
       - The possible failure modes of the incoming data should be determined, especially complex, digital data strings from an external system
       - For loops where the loop count is a computed value, test cases should be developed to attempt to compute out-of-range loop count values, and thus demonstrate the robustness of the loop-related code.
       - A check should be made to ensure that protection mechanisms for exceeded frame

times respond correctly o For time-related functions, such as filters, integrators and delays, test cases should be developed for arithmetic overflow protection mechanisms o For state transitions, tests cases should be developed to provoke transitions that are not allowed by the software requirements. **Observation 2:** Les «Requirements-based Testing» doivent satisfaire les objectifs a, b et c du point 6.4.3, à savoir: a. Testing should concentrate on error sources associated with the software operating within the target computer environment, and on the high-level functionality. b. Testing should concentrate on the inter-relationships between the software requirements, and on the implementation of requirements by the software architecture. c. Testing method should concentrate on demonstrating that each software component complies with its low-level requirements **Observation 3:** Les «Requirements-based Testing» doivent garder en considération les types d'erreurs mentionnés dans les objectifs a, b et c du point 6.4.

Tableau V.3

Analyse des résultats de Test d'intégration : Objectif 03

Définition de l'objectif 03:
"**Executable object code complies** with **low-level requirements**"
Description de l'objectif 03:
6.4.2.1: (voir description dans le tableau 32, section Description de l'objectif 1) **6.4.3:** (voir description dans le tableau 32, section Description de l'objectif 1)
Référence dans le guide DO-178B:
6.4.2.1 et 6.4.3
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination de la conformité du code exécutable et les exigences de bas niveau. Il faut noter que les tests doivent être exécutés tel que décrits dans les objectifs de la DO-178B et en gardant l'objectif suivant « **Executable object code complies** with **low-level requirements**». Autrement dit en gardant la conformité entre le code exécutable et les exigences de bas niveau. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Référence dans OpenUP:

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   Il faut exécuter les tâches suivantes mais en respectant les observations consignés plus bas dans cette section.

   - En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 5»:
       - Review the requirements to be tested
       - Identify relevant Test Cases
       - Outline the Test Cases
       - Identify test data needs
       - Share and evaluate the Test Cases
   - En référence à la tâche : « Implement Tests Scripts» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 6»:
       - Select Test Cases to implement
       - Design the Test Script
       - Implement the executable Test Script
       - Define specific test data
       - Verify Test Script implementation
       - Share and evaluate Test Scripts
   - En référence à la tâche : « Run Tests» de la discipline « Test». Cette tâche est adressée au niveau des tests de l'application logiciel. La tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 5»:
       - Review work items completed in the build
       - Select Test Scripts
       - Execute Test Scripts against the build
       - Analyze and communicate test results
       - Provide feedback to the team

     Observation 1: Les «Normal test cases» doivent satisfaire l'objectif 6.4.2.1, à savoir:

       - Real and integer input variables should be exercised using valid equivalence classes and boundary values
       - For time-related functions, such as filters, integrators and delays, multiple iterations of the code should be performed to check the characteristics of the function in context
       - For state transitions, test cases should be developed to exercise the transitions possible during normal operation
       - For software requirements expressed by logic equations, the normal range test cases should verify the variable usage and the Boolean operators

Observation 2: Les «Requirements-based Testing» doivent satisfaire les objectifs a, b et c du point 6.4.3, à savoir:

a. Testing should concentrate on error sources associated with the software operating within the target computer environment, and on the high-level functionality.
b. Testing should concentrate on the inter-relationships between the software requirements, and on the implementation of requirements by the software architecture.
c. Testing method should concentrate on demonstrating that each software component complies with its low-level requirements

Observation 3: Les «Requirements-based Testing» doivent garder en considération les types d'erreurs mentionnés dans les objectifs a, b et c du point 6.4.3.

Tableau V.4

Analyse des résultats de Test d'intégration : Objectif 04

Définition de l'objectif 04:
"Executable object code **is robust** with low-level requirements"
Description de l'objectif 04:
6.4.2.2: (voir description dans le tableau 33, section Description de l'objectif 2) 6.4.3: (voir description dans le tableau 32, section Description de l'objectif 1)
Référence dans le guide DO-178B:
6.4.2.2 et 6.4.3
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination de la robustesse du code exécutable et les exigences de bas niveau. Il faut noter que les tests doivent être exécutés tel que décrits dans les objectifs de la DO-178B et en gardant l'objectif suivant « **Executable** object code is **robust** with **low-level requirements**». Autrement dit en gardant l'objectif que le code exécutable doit agir tel que prévu avec les exigences de bas niveau, dans les cas des conditions ou des données anormales. **Analyse de l'objectif du point de vue d'OpenUP :** Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Référence dans OpenUP:
1. Au niveau du rôle: **Selon le type de V&V à implanter dans le projet:**

- **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
- **L'utilisation des rôles d'analyste ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. Au niveau des disciplines, tâches et des artefacts:

Il faut exécuter les tâches suivantes mais en respectant les observations consignés plus bas dans cette section.

- En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
  - Les « Step 1» à « Step 5»:
    - Review the requirements to be tested
    - Identify relevant Test Cases
    - Outline the Test Cases
    - Identify test data needs
    - Share and evaluate the Test Cases
- En référence à la tâche : « Implement Tests Scripts» de la discipline « Test». Cette tâche devrait demander obligatoirement:
  - Les « Step 1» à « Step 6»:
    - Select Test Cases to implement
    - Design the Test Script
    - Implement the executable Test Script
    - Define specific test data
    - Verify Test Script implementation
    - Share and evaluate Test Scripts
- En référence à la tâche : « Run Tests» de la discipline « Test». Cette tâche est adressée au niveau des tests de l'application logiciel. La tâche devrait demander obligatoirement:
  - Les « Step 1» à « Step 5:
    - Review work items completed in the build
    - Select Test Scripts
    - Execute Test Scripts against the build
    - Analyze and communicate test results
    - Provide feedback to the team

Observation 1: Les «Normal test cases» doivent satisfaire l'objectif 6.4.2.1, à savoir:

- Real and integer input variables should be exercised using valid equivalence classes and boundary values
- For time-related functions, such as filters, integrators and delays, multiple iterations of the code should be performed to check the characteristics of the function in context
- For state transitions, test cases should be developed to exercise the transitions possible during normal operation
- For software requirements expressed by logic equations, the normal range test cases should verify the variable usage and the Boolean operators

Observation 2: Les «Requirements-based Testing» doivent satisfaire les objectifs a, b et c du point 6.4.3, à savoir:

a. Testing should concentrate on error sources associated with the software operating within the target computer environment, and on the high-level functionality.

b. Testing should concentrate on the inter-relationships between the software requirements, and on the implementation of requirements by the software architecture.
c. Testing method should concentrate on demonstrating that each software component complies with its low-level requirements

Observation 3: Les «Requirements-based Testing» doivent garder en considération les types d'erreurs mentionnés dans les objectifs a, b et c du point 6.4.3.

Tableau V.5

Analyse des résultats de Test d'intégration : Objectif 05

Définition de l'objectif 05:
"Executable object code is **compatible** with target computer"
Description de l'objectif 05:
Objective 6.4.3a a. Requirements-based Hardware/Software Integration Testing: **This testing method** should concentrate on **error sources associated with the software operating within the target computer environment, and on the high-level functionality**. The objective of requirements-based hardware/software integration testing is to ensure that the software in the target computer will satisfy the high-level requirements. Typical errors revealed by this testing method include: o Incorrect interrupt handling. o Failure to satisfy execution time requirements. o Incorrect software response to hardware transients or hardware failures, for example, o Start-up sequencing, transient input loads and input power transients. o Data bus and other resource contention problems, for example, memory mapping. o Inability of built-in test to detect failures. o Errors in hardware/software interfaces. o Incorrect behaviour of feedback loops. o Incorrect control of memory management hardware or other hardware devices under o Software control. o Stack overflow. o Incorrect operation of mechanism(s) used to confirm the correctness and Compatibility of field-loadable software. o Violations of software partitioning.
Référence dans le guide DO-178B:
6.4.3a
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination de la compatibilité du code exécutable et l'ordinateur cible.

Il faut noter que les tests doivent être exécutés tel que décrits dans les objectifs de la DO-178B et en gardant l'objectif suivant « **Executable object code** is **compatible** with **target computer**». Autrement dit en gardant la compatibilité entre le code exécutable et l'ordinateur cible où le logiciel doit s'exécuter.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   Il faut exécuter les tâches suivantes mais en respectant les observations consignés plus bas dans cette section.

   - En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - L'exécution des « Step 1» à « Step 5»:
       - Review the requirements to be tested
       - Identify relevant Test Cases
       - Outline the Test Cases
       - Identify test data needs
       - Share and evaluate the Test Cases
   - En référence à la tâche : « Implement Tests Scripts» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - L'exécution des « Step 1» à « Step 6»:
       - Select Test Cases to implement
       - Design the Test Script
       - Implement the executable Test Script
       - Define specific test data
       - Verify Test Script implementation
       - Share and evaluate Test Scripts
   - En référence à la tâche : « Run Tests» de la discipline « Test». Cette tâche est adressée au niveau des tests de l'application logiciel. La tâche devrait demander obligatoirement:
     - L'exécution des « Step 1» à « Step 5:
       - Review work items completed in the build
       - Select Test Scripts
       - Execute Test Scripts against the build
       - Analyze and communicate test results

- Provide feedback to the team

Observation 1: Les «Requirements-based Hardware/Software Integration Testing» doit garder en considération les types d'erreurs mentionnés dans cet objectif.

ANNEXE VI

ANALYSE ET CONCEPTION DES PRATIQUES DE V&V POUR LE PROCESSUS DE VÉRIFICATION DU LOGICIEL CRITIQUE

Introduction

Cette Annexe traite sur les pratiques (exemple : exécution de tâches, rôles, responsabilités, et artefacts en général) qui doivent être exécutées en vue de vérifier les résultats du processus de vérification du logiciel critique. Les objectifs de ce processus font référence au tableau A-7 du guide DO-178B, voir annexe VII, tableau VII.5.

Il faut noter également que cette annexe fait référence aux objectifs du DO-178B qui sont directement reliés à la vérification de la structure du logiciel. Ce type de vérification porte sur l'analyse des résultats et non aux tests. L'analyse de structure du logiciel a pour but d'analyser l'ensemble des tests implémentés pour le logiciel et déterminer si toutes les possibilités ont été couvertes, et s'il n'a pas des exigences qui n'ont pas été implémentées et testes (Hilderman V. et Baghi T., 2007).

Voilà quelques concepts du guide DO-178B (RTCA/DO-178B, 1992) qui sont nécessaires pour l'exécution de l'analyse de la structure du logiciel:

> « **Condition**: A condition is a leaf-level Boolean expression (it cannot be broken down into a simpler Boolean expression) ».
>
> «**Decision:** A Boolean expression composed of conditions and zero or more Boolean operators. A decision without a Boolean operator is a condition. If a condition appears more than once in a decision, each occurrence is a distinct condition ».

Les tableaux prochains présenteront entre autres l'analyse des objectifs du guide DO-178B versus l'OpenUP, ainsi que la proposition des pratiques de V&V qui constitueront les fondements de la conception du référentiel de V&V.

Tableau VI.1

Analyse des résultats de la vérif. des processus de vérification : Obj. 01

Définition de l'objectif 01:

"The test procedures are **correct**"

Description de l'objectif 01:

"The objective is to verify that the **test cases** were **accurately** developed into **test procedures** and **expected results**"

Référence dans le guide DO-178B:

6.3.6b

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la détermination de l'exactitude comme établi des tests cases/procédures.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts :**

   - En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 5», afin de vérifier que l'implémentation du code fonctionne comme il a été spécifié. Les « Steps » sont :
       - Review the requirements to be tested
       - Identify relevant Test Cases
       - Outline the Test Cases
       - Identify test data needs
       - Share and evaluate the Test Cases

- En référence à la tâche : « Implement Tests Scripts» de la discipline « Test». Cette tâche devrait demander obligatoirement:
  - Les « Step 1» à « Step 6», afin de vérifier que les Tests Cases ont été implementés comme Test Procédures. Les « Steps » sont :
    - Select Test Cases to implement
    - Design the Test Script
    - Implement the executable Test Script
    - Define specific test data
    - Verify Test Script implementation
    - Share and evaluate Test Scripts

- La discipline « Test» fait référence à la check-list « Test Case». Cette Check-list devrait demander obligatoirement l'exécution de touts les « Steps » décrits dans l'artefact.

- La discipline « Test» fait référence à la check-list « Test Scripts». Cette Check-list devrait demander obligatoirement l'exécution de touts les « Steps » décrits dans l'artefact.

Tableau VI.2

Analyse des résultats de la vérif. des processus de vérification: Obj. 02

Définition de l'objectif 02:
"The **test results** are **correct** and **discrepancies explained**"
Description de l'objectif 02:
"The objective is to ensure that **the test results are correct** and that **discrepancies between actual and expected results are explained**".
Référence dans le guide DO-178B :
6.3.6c
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination de l'exactitude comme établi des tests cases/procédures. **Analyse de l'objectif du point de vue d'OpenUP :** Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système. Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle :**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste, développeur ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - En référence à la tâche : « Run Developer Tests» de la discipline « Development». Cette tâche est adressée au niveau des tests du développeur. La tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 3», afin de vérifier que les tests sont corrects et que les désaccords sont résolues. Les « Steps » sont :
       - Run Developer Tests
       - Evaluate test execution
       - Respond to test results Provide feedback to the team
   - En référence à la tâche : « Run Tests» de la discipline « Test». Cette tâche est adressée au niveau des tests de l'application logiciel. La tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 5», afin de vérifier que les tests sont corrects et que les désaccords sont résolues. Les « Steps » sont :
       - Review work items completed in the build
       - Select Test Scripts
       - Execute Test Scripts against the build
       - Analyze and communicate test results
       - Provide feedback to the team

Tableau VI.3

Analyse des résultats de la vérif. des processus de vérification : Obj. 03

Definition de l'objectif 03 :

"The **test coverage** of **high-level requirements** is achieved"

Description de l'objectif 03 :

Objective 6.4.4.1

"The objective of this analysis is to determine how well the **requirements-based testing** verified the **implementation of the software requirements**. This analysis may reveal the need for additional requirements-based test cases. The requirements-based test coverage analysis should show that:

a. Test cases exist for each software requirement.
b. Test cases satisfy the criteria of normal and robustness testing as defined in paragraph 6.4.2 (reference le DO-178B)."

Objective 6.4.2 : Requirements-Based Test Case Selection
Requirements-based testing is emphasized because this strategy has been found to be the most effective at revealing errors. Guidance for requirements-based test case selection includes:

a. To implement the software testing objectives, two categories of test cases should be included: normal range test cases and robustness (abnormal range) test cases.
b. The specific test cases should be developed from the software requirements and the error sources inherent in the software development process.

Référence dans le guide DO-178B :

6.4.4.1

Analyse de l'objectif :

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la détermination que les tests de couverture sur les exigences de haut niveau ont été accomplies.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste, développeur ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   - En référence à la tâche : « Implement Developper Tests » de la discipline « Development ». Cette tâche devrait demander obligatoirement :
     - L'exécution du « Step 1» à le « Step 7», à savoir :
       - Refine scope and identify the test(s)
       - Write the test setup
       - Define the expected results
       - Write the test logic
       - Define the test response
       - Write clean-up code
       - Test the test
   - En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - L'exécution des « Step 1» à « Step 5», afin de vérifier que l'implémentation du code fonctionne

comme a été spécifié. Les « Steps » sont :

- Review the requirements to be tested
- Identify relevant Test Cases
- Outline the Test Cases
- Identify test data needs
- Share and evaluate the Test Cases

Observation : Les « Steps » précédents devront être adaptés afin de remplir les exigences décrites pour l'objectif 03, à savoir :

- Test cases exist for each software requirement.
- Test cases satisfy the criteria of normal and robustness testing as defined in paragraph 6.4.2 (reference le DO-178B)."

Tableau VI.4

Analyse des résultats de la vérif. des processus de vérification : Obj. 04

Definition de l'objectif 04:
"The **test coverage** of **low-level requirements** is achieved"
Description de l'objectif 04:
Objective 6.4.4.1 "The objective of this analysis is to determine how well the **requirements-based testing** verified the implementation of the software requirements. This analysis may reveal the need for additional requirements-based test cases. The requirements-based test coverage analysis should show that: a. Test cases exist for each software requirement. b. Test cases satisfy the criteria of normal and robustness testing as defined in paragraph 6.4.2 (RTCA/DO-178B)." ***Objective 6.4.2 : Requirements-Based Test Case Selection*** Requirements-based testing is emphasized because this strategy has been found to be the most effective at revealing errors. Guidance for requirements-based test case selection includes: c. To implement the software testing objectives, two categories of test cases should be included: normal range test cases and robustness (abnormal range) test cases. d. The specific test cases should be developed from the software requirements and the error sources inherent in the software development process.
Référence dans le guide DO-178B:
6.4.4.1
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination que les tests de couverture sur les exigences de bas niveau ont été accomplies.

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste, développeur ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   Observation 1 : Étant donné que n'existe pas une référence dans l'OpenUP d'une tâche destinée au test du design (ou Low Level Requirements), il est nécessaire d'exécuter les tâches suivantes avec une emphase dans le design ou « Low Level Requirements ».

   - En référence à la tâche : « Implement Developper Tests » de la discipline « Development ». Cette tâche devrait demander obligatoirement :
     - Les « Step 1» à le « Step 7», à savoir :
       - Refine scope and identify the test(s)
       - Write the test setup
       - Define the expected results
       - Write the test logic
       - Define the test response
       - Write clean-up code
       - Test the test
   - En référence à la tâche : « Create Tests Cases» de la discipline « Test». Cette tâche devrait demander obligatoirement:
     - Les « Step 1» à « Step 5», afin de vérifier que l'implémentation du code fonctionne comme a été spécifié. Les « Steps » sont :
       - Review the requirements to be tested (ie. Low Level Requirements)
       - Identify relevant Test Cases
       - Outline the Test Cases
       - Identify test data needs
       - Share and evaluate the Test Cases

   Observation 2 : Les « Steps » précédents devront être adaptés afin de remplir les exigences décrites pour l'objectif 03, à savoir :
     - Test cases exist for each software requirement.
     - Test cases satisfy the criteria of normal and robustness testing as defined in paragraph 6.4.2 (reference le DO-178B)."

Tableau VI.5

Analyse des résultats de la vérif. des processus de vérification : Obj. 05

Définition de l'objectif 05:
"The **test coverage** of **software structure** (***modified condition/decision coverage***) is achieved"
Description de l'objectif 05:
"The objective of this analysis is to determine which **code structure** was not exercised by the requirements-based test procedures. The requirements-based test cases may not have completely exercised the code structure, so structural coverage analysis is performed and additional verification produced to provide structural coverage. Guidance includes: a. The analysis should confirm the degree of structural coverage appropriate to the software level. b. The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code. c. The analysis should confirm the data coupling and control coupling between the code components".
Référence dans le guide DO-178B:
6.4.4.2
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination que les tests de couverture type « MC/DC » ont été accomplies. Voici la définition du MC/DC décrite dans le DO-178B : *« **Modified Condition/Decision Coverage:** Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken on all possible outcomes at least once, and each condition has been shown to affect that decision outcome independently. A condition is shown to affect a decision's outcome independently by varying just that decision while holding fixed all other possible conditions ».* ***Il faut savoir que:*** MCDC est une technique des tests de couverture structurelle composé de critères principalement concernés par l'exécution de la logique booléenne. Les tests MCDC fournissent un grand nombre des tests exhaustives à partir des expressions booléennes (Chilenski John Joseph et Miller Steven P, 1994). **Analyse de l'objectif du point de vue d'OpenUP :** Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système.

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste, développeur ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   Observation: Étant donné qu'il n'existe pas une référence dans l'OpenUP pour cet objectif de la DO-178B, il faut créer une tâche explicite dans l'OpenUP. Cette tâche doit être exécutée à l'intérieur de la discipline test de l'OpenUP. Bien sûr la tâche pourrait faire partie d'une discipline appelée V&V.

   - La nouvelle tâche appelée « Run MC/DC Analysis» devrait s'exécuter après la tâche : « Run Tests » de la discipline « Test ». Il est possible également d'exécuter la tâche comme un « Step » de la tâche « Run Test ».
   - La tâche « Run MC/DC Analysis» doit s'assurer que toute la structure du code a été incluse dans les procédures des tests selon les exigences. L'analyse doit comprendre :
     a. The analysis should confirm the degree of structural coverage appropriate to the software level.
     b. The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code.
     c. The analysis should confirm the data coupling and control coupling between the code components".

Tableau VI.6

Analyse des résultats de la vérif. des processus de vérification : Obj. 06

Définition de l'objectif 06 :
"The **test coverage** of **software structure** (***decision coverage***) is achieved"
Description de l'objectif 06 :
"The objective of this analysis is to determine which code structure was not exercised by the requirements-based test procedures. The requirements-based test cases may not have completely exercised the code structure, so structural coverage analysis is performed and additional verification produced to provide structural coverage. Guidance includes: a. The analysis should confirm the degree of structural coverage appropriate to the software level. b. The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code".
Référence dans le guide DO-178B:
6.4.4.2a et 6.4.4.2b
Analyse de l'objectif:
Explication de l'objectif du guide DO-178B : Cet objectif traite de la vérification de la détermination que les tests de couverture type « Decision Coverage » ont été accomplies. Voici la définition du « Decision Coverage » décrite dans le DO-178B : *« **Decision Coverage:** Every point of entry and exit in the program has been invoked at least once, and every decision in the program has taken all possible outcomes at least once».* **Analyse de l'objectif du point de vue d'OpenUP :** Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système. Il est Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.
Proposition des pratiques pour la conception du référentiel de V&V :
1. Au niveau du rôle: **Selon le type de V&V à implanter dans le projet:** • **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :

- **L'utilisation des rôles d'analyste, développeur ou tester** tel que défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

Observation: Étant donné qu'il n'existe pas une référence dans l'OpenUP pour cet objectif de la DO-178B, il faut créer une tâche explicite dans l'OpenUP. Cette tâche doit être exécutée à l'intérieur de la discipline test de l'OpenUP. Bien sûr la tâche pourrait faire partie d'une discipline appelée V&V.

- La nouvelle tâche appelée « Run Decision Coverage Analysis » devrait s'exécuter après la tâche : « Run Tests » de la discipline « Test ». Il est possible également d'exécuter la tâche comme un « Step » de la tâche « Run Test ».
- La tâche « Run Decision Coverage Analysis» doit s'assurer que toute la structure du code a été comprise dans les procédures des tests selon les exigences.

Observation : L'analyse doit comprendre :

a. The analysis should confirm the degree of structural coverage appropriate to the software level.
b. The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code.

Tableau VI.6

Analyse des résultats de la vérif. des processus de vérification: Obj. 07

Definition de l'objectif 07 :
"The **test coverage** of **software structure** (***statement coverage***) is achieved"
Description de l'objectif 07 :
"The objective of this analysis is to determine which code structure was not exercised by the requirements-based test procedures. The requirements-based test cases may not have completely exercised the code structure, so structural coverage analysis is performed and additional verification produced to provide structural coverage. Guidance includes: a. The analysis should confirm the degree of structural coverage appropriate to the software level. b. The structural coverage analysis may be performed on the Source Code unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code".

Référence dans le guide DO-178B :

6.4.4.2a et 6.4.4.2b

Analyse de l'objectif :

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la détermination que les tests de couverture type « Statement Coverage » ont été accomplies.

Voici la définition du « Statement Coverage » décrite dans le DO-178B :

*« **Statement Coverage:** Every statement in the program has been executed at least once ».*

Analyse de l'objectif du point de vue d'OpenUP :

Il n'existe pas une référence explicite à la conformité entre les exigences de haut niveau et du système.

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste, développeur ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. **Au niveau des disciplines, tâches et des artefacts:**

   Observation: Étant donné que n'existe pas une référence dans l'OpenUP pour cet objectif de la DO-178B, il faut créer une tâche explicite dans l'OpenUP. Cette tâche doit être exécutée à l'intérieur de la discipline test de l'OpenUP. Bien sûr la tâche pourrait faire partie d'une discipline appelé V&V.

   - La nouvelle tâche appelé « Run statement coverage Analysis » devrait s'exécuter après la tâche : « Run Tests » de la discipline « Test ». Il est possible également d'exécuter la tâche comme un « Step » de la tâche « Run Test ».
   - La tâche « Run Statement Coverage Analysis» doit s'assurer que toute la structure du code a été comprise dans les procédures des tests selon les exigences.

     Observation : L'analyse doit comprendre :

     a. The analysis should confirm the degree of structural coverage appropriate to the software level.
     b. The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code.

Tableau VI.8

Analyse des résultats de la vérif. des processus de vérification : Obj. 08

Definition de l'objectif 08 :

"The **test coverage** of **software structure** (***data coupling and control coupling***) is **achieved**"

Description de l'objectif 08 :

"The objective of this analysis is to determine which code structure was not exercised by the requirements-based test procedures. The requirements-based test cases may not have completely exercised the code structure, so structural coverage analysis is performed and additional verification produced to provide structural coverage. Guidance includes:

c. The analysis should confirm the data coupling and control coupling between the code components".

Référence dans le guide DO-178B :

6.4.4.2c

Analyse de l'objectif:

Explication de l'objectif du guide DO-178B :

Cet objectif traite de la vérification de la détermination que les tests de couverture type « Data coupling and Control Coupling » ont été accomplies.

Voici les définitions extraites du DO-178B :

« **Data coupling:** The dependence of a software component on data not exclusively under the control of that software component ».

« **Control coupling:** The manner or degree by which one software component influences the execution of another software component ».

Analyse de l'objectif du point de vue d'OpenUP :

Il est nécessaire d'améliorer les pratiques d'OpenUP par l'incorporation des exigences particulières pour cet objectif, tel que décrit par le guide DO-178B. Nous présenterons les améliorations pertinentes dans le la section « Proposition des pratiques pour la conception du référentiel de V&V » plus bas dans ce tableau.

Proposition des pratiques pour la conception du référentiel de V&V :

1. **Au niveau du rôle:**

   Selon le type de V&V à implanter dans le projet:

   - **Création d'un V&V Analyste (ou groupe),** qui serait responsable d'appliquer la vérification de façon indépendante et dépendamment du niveau de criticité du logiciel. Ou bien :
   - **L'utilisation des rôles d'analyste, développeur ou tester** tel qui est défini dans l'OpenUP. Ces rôles seront adaptés selon la réalité particulière de chaque projet (ex. niveau de criticité, ressources disponibles, contraintes du temps, des coûts, etc.).

   Note : Les rôles devront utiliser les disciplines, tâches, artefacts, produits de travail (Work Products) proposés dans le point 2.

2. Au niveau des disciplines, tâches et des artefacts:

- En référence à la tâche : « Implement the solution » de la discipline « Development ». Cette tâche devrait demander obligatoirement:
  - Le du « Step 5: Evaluate the Implementation». Le « Step » fait référence par exemple à l'évaluation du code par rapport au excessive « coupling » et « circular dependencies ».
- En référence à la tâche : « Design the solution » de la discipline « Development ». Cette tâche devrait demander obligatoirement:
  - Le du « Step 6: Are the design elements modular? de la checklist « Design». Le « Step » fait référence à la vérification de la cohésion et couplage des éléments du design.

Observation : Lors de l'exécution de cette tâche il faut mettre emphase sur les aspects du « data coupling » et « control coupling » dans la structure des composants logiciels. Il faut bien instancier la définition des ces aspects dans les tâches de vérification.

ANNEXE VII

TABLEAUX DES DE VERIFICATION DES RESUSLTATS DES PROCESUS DE CONSTRUCTION DU LOGICIEL CRITIQUE SEON LE DO-178B

L'ensemble des tableaux présentés dans cette annexe montrent les objectifs du guide DO-178B par niveau de criticité du logiciel (« Applicability by SW Level »), ainsi que par le niveau d'indépendance exigé par l'objectif.

Les tableaux font également mention des livrables de sortie (gabarits) qui décrivent les objectif comblés. Il est inclut aussi dans les tableaux une référence à la section du guide DO-178B où le gabarit est décrit. Une référence au contrôle de la configuration avec les catégories 1 ou 2 (CC1 et CC2) est aussi présentée.

Tableau VII.1

Vérification des résultats du processus des exigences selon le DO-178B

Table A-3

Verification Of Outputs of Software Requirements Process

	Objective		Applicability by SW Level				Output		Control Category by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Software high-level requirements comply with system requirements.	6.3.1a	●	●	○	○	Software Verification Results	11.14	②	②	②	②
2	High-level requirements are accurate and consistent.	6.3.1b	●	●	○	○	Software Verification Results	11.14	②	②	②	②
3	High-level requirements are compatible with target computer.	6.3.1c	○	○			Software Verification Results	11.14	②	②		
4	High-level requirements are verifiable.	6.3.1d	○	○	○		Software Verification Results	11.14	②	②	②	
5	High-level requirements conform to standards.	6.3.1e	○	○	○		Software Verification Results	11.14	②	②	②	
6	High-level requirements are traceable to system requirements.	6.3.1f	○	○	○	○	Software Verification Results	11.14	②	②	②	②
7	Algorithms are accurate.	6.3.1g	●	●	○		Software Verification Results	11.14	②	②	②	

LEGEND:		
	●	The objective should be satisfied with independence.
	○	The objective should be satisfied.
	Blank	Satisfaction of objective is at applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

Tableau VII.2

Vérification des résultats du processus de design selon le DO-178B

Table A-4

Verification Of Outputs of Software Design Process

	Objective		Applicability by SW Level				Output		Control Category by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Low-level requirements comply with high-level requirements.	6.3.2a	●	●	○		Software Verification Results	11.14	②	②	②	
2	Low-level requirements are accurate and consistent.	6.3.2b	●	●	○		Software Verification Results	11.14	②	②	②	
3	Low level requirements are compatible with target computer.	6.3.2c	○	○			Software Verification Results	11.14	②	②		
4	Low-level requirements are verifiable.	6.3.2d	○	○			Software Verification Results	11.14	②	②		
5	Low-level requirements conform to standards.	6.3.2e	○	○	○		Software Verification Results	11.14	②	②	②	
6	Low-level requirements are traceable to high-level requirements.	6.3.2f	○	○	○		Software Verification Results	11.14	②	②	②	
7	Algorithms are accurate.	6.3.2g	●	●	○		Software Verification Results	11.14	②	②	②	
8	Software architecture is compatible with high-level requirements.	6.3.3a	●	○	○		Software Verification Results	11.14	②	②	②	
9	Software architecture is consistent.	6.3.3b	●	○	○		Software Verification Results	11.14	②	②	②	
10	Software architecture is compatible with target computer.	6.3.3c	○	○			Software Verification Results	11.14	②	②		
11	Software architecture is verifiable.	6.3.3d	○	○			Software Verification Results	11.14	②	②		
12	Software architecture conforms to standards.	6.3.3e	○	○	○		Software Verification Results	11.14	②	②	②	
13	Software partitioning integrity is confirmed.	6.3.3f	●	○	○	○	Software Verification Results	11.14	②	②	②	②

LEGEND:

●	The objective should be satisfied with independence.
○	The objective should be satisfied.
Blank	Satisfaction of objective is at applicant's discretion.
①	Data satisfies the objectives of Control Category 1 (CC1).
②	Data satisfies the objectives of Control Category 2 (CC2).

Tableau VII.3

Vérification des résultats du processus de codification et d'intégration selon le DO-178B

Table A-5

Verification Of Outputs of Software Coding & Integration Processes

	Objective		Applicability by SW Level				Output		Control Category by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Source Code complies with low-level requirements.	6.3.4a	●	●	○		Software Verification Results	11.14	②	②	②	
2	Source Code complies with software architecture.	6.3.4b	●	○	○		Software Verification Results	11.14	②	②	②	
3	Source Code is verifiable.	6.3.4c	○	○			Software Verification Results	11.14	②	②		
4	Source Code conforms to standards.	6.3.4d	○	○	○		Software Verification Results	11.14	②	②	②	
5	Source Code is traceable to low-level requirements.	6.3.4e	○	○	○		Software Verification Results	11.14	②	②	②	
6	Source Code is accurate and consistent.	6.3.4f	●	○	○		Software Verification Results	11.14	②	②	②	
7	Output of software integration process is complete and correct.	6.3.5	○	○	○		Software Verification Results	11.14	②	②	②	

LEGEND:	●	The objective should be satisfied with independence.
	○	The objective should be satisfied.
	Blank	Satisfaction of objective is at applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

Tableau VII.4

Vérification des résultats du processus d'intégration selon le DO-178B

Table A-6

Testing Of Outputs of Integration Process

	Objective		Applicability by SW Level				Output		Control Category by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Executable Object Code complies with high-level requirements.	6.4.2.1 6.4.3	○	○	○	○	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	② ②
2	Executable Object Code is robust with high-level requirements.	6.4.2.2 6.4.3	○	○	○	○	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	② ②
3	Executable Object Code complies with low-level requirements.	6.4.2.1 6.4.3	●	●	○		Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	
4	Executable Object Code is robust with low-level requirements.	6.4.2.2 6.4.3	●	○	○		Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	
5	Executable Object Code is compatible with target computer.	6.4.3a	○	○	○	○	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	② ②

LEGEND:		
	●	The objective should be satisfied with independence.
	○	The objective should be satisfied.
	Blank	Satisfaction of objective is at applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

Tableau VII.5

Vérification des résultats de la vérification du processus de vérification selon le DO-178B

Table A-7

Verification Of Verification Process Results

	Objective		Applicability by SW Level				Output		Control Category by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Test procedures are correct.	6.3.6b	●	○	○		Software Verification Cases and Procedures	11.13	②	②	②	
2	Test results are correct and discrepancies explained.	6.3.6c	●	○	○		Software Verification Results	11.14	②	②	②	
3	Test coverage of high-level requirements is achieved.	6.4.4.*	●	○	○	○	Software Verification Results	11.14	②	②	②	②
4	Test coverage of low-level requirements is achieved.	6.4.4.1	●	○	○		Software Verification Results	11.14	②	②	②	
5	Test coverage of software structure (modified condition/decision) is achieved.	6.4.4.2	●				Software Verification Results	11.14	②			
6	Test coverage of software structure (decision coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●			Software Verification Results	11.14	②	②		
7	Test coverage of software structure (statement coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●	○		Software Verification Results	11.14	②	②	②	
8	Test coverage of software structure (data coupling and control coupling) is achieved.	6.4.4.2c	●	●	○		Software Verification Results	11.14	②	②	②	

LEGEND:	●	The objective should be satisfied with independence.
	○	The objective should be satisfied.
	Blank	Satisfaction of objective is at applicant's discretion.
	①	Data satisfies the objectives of Control Category 1 (CC1).
	②	Data satisfies the objectives of Control Category 2 (CC2).

ANNEXE VIII

DEFINITION DES EXIGENCES DE L'ENTREPÔT DES MESURES

This appendix contains several features that we established for implementing in the Performance Measurement Repository Tool.

Tableau VIII.1

Features of the Measurement Repository

Feature	Description
1. The tool shall offer a generic platform for storage and retrieval of metric information.	This is the purpose of the tool (Measurement Repository Tool). Any tool that constrains us to a set of predefined metrics or a predefined measurement process is not compliant. The reason for this very stringent condition is that even IF we were going to find a tool with pre-established metrics that today satisfy all our need; this might not be the case for tomorrow's needs.
2. The tool shall be capable of handling the following type of scales: - Nominal - Ordinal - Interval - Ratio	Measurement data is usually expressed in one of these four types of scales. See ISO 15939 for a definition of the scales. By handling or support we mean, storing, retrieving, filtering, performing arithmetic and statistic operations, and displaying as applicable to the type of data.
3. The tool shall support base and derived measures.	A base measure is a measure whose value is defined through the measurement of an attribute. A derived measure is a measured whose value is obtained by applying a number of mathematical or logical operations to other derived measures or to base measures.
4. Tool shall support the definition, query and display of the metrics meta-data, such as an alias, a description, target values, etc.	Self-documentation.

5. The tool shall support aggregation across user definable hierarchies.	Different management levels needs to have access to consolidated information and from different perspectives. As a minimum, the tool shall be capable of aggregating along products, departments and projects.
6. The tool shall support drill-down capabilities.	This allows for the navigation of the aggregation structure with the purpose of arriving to the source of the data.
7. The tool shall support time series data.	Time sequence data is fundamental to understand the evolution of a metric and to forecast results.
8. The tool shall support weekly, monthly and quarterly calendars.	These are typical data capture and reporting frequencies.
9. The tool shall support user-defined calendars.	These will allow the capture and reporting of data based on milestones and tollgates.
10. The tool shall allo w metric specific calendars.	Different metrics have different data collection periods.
11. To ol shall provide the following checks on the data inputted: - Type verification - Range verification - List of values verification - Missing (non-reported) values.	Assure data integrity.
12. The tool shall support the following data entry modes: - Manual - Import - Automatic data capture	We will face the three types of circumstances. i.e., low volume sporadic data will be entered manually, data from spreadsheet or similar applications will be entered via imported files with a common format and high-volume high-frequency data such as that coming from time reporting will have to be automated.
13. The tool shall provide as minimum the following operations among measurements: - Mathematical: +, - , * ,/, **, Abs, Sign, Sqr, Log, Sum - Statistics: Avg, Min, Max, Std, Median - Logic: <, >, =, <>, And, Or, Not - Date: Period offset, date arithmetic	To implement derived metrics

14. The tool shall provide the capability to define business rules for each metric such as when the rule is broken the metric is highlighted, an alarm raised and interested subscribers notified.	To implement management by exception.
15. The tool shall provide a macro writing or scripting functionality that allows the extension or automation of common operations.	To support customisation.
16. The tool shall be capable of presenting the metrics in text based reports as well as in charts.	Data presentation
17. The tool shall support the following type of charts: - Histograms - Line charts - Scatter plots - Pie charts - Bubble charts - Kiviat (radar) charts	Data presentation
18. The tool shall support the disp lay of up to 8 different data series in a single chart.	Data presentation. Useful for comparison purposes.
19. To ol shall support the display of multiple measures simultaneously.	Most of the time to understand what is going on it is necessary to look at two or more measurements at the same time. For example the display of histograms and line data in a single chart.
20. The tool shall support the customization of the following chart attributes: - Background color - Series color - Axis scales	Data presentation
21. The tool shall be capable of operating in subsets of the data.	Data analysis
22. The tool shall provide browsi ng capabilities based on the aggregation structures and on time frames.	This capability helps the user to search the right measurements by displaying them in a structured way. Supports the idea of "I'll know when I see it".
23. Shall support query capabilities for metrics stored in the tool's repository.	This capability helps the user to search the right measurements by selecting a number of criteria to be satisfied by the information searched. Supports the idea of "I know what I need".
24. The look and feel of the i nterface shall be consistent with the style of the underlying Windows Manager.	Consistency of use is critical in tool acceptance and productivity increase.

25. The tool shall provide a consis tent model based in a point and click approach. Mandatory use of the keyboard shall be limited to data inputs.	Consistency of use is critical in tool acceptance and productivity increase.
26. The tool shall provide ke yboard shortcuts for requesting operations.	Some experienced users prefer this approach over the point and click interface.
27. The interface s hall be context sensitive.	This minimizes the needs for switching tool modes and the number of commands to memorize. Examples of context sensitive actions are: given a method artifact, navigate to the appropriate decomposition, bring in the appropriate annotation form, etc.
28. The tool shall support the personalization of the application concerning file locations, display colors, tool bars and font sizes.	Tool acceptability.
29. Response t imes shall be consistent with user expectations. This mean that a single operation shall take almost not time and a complicated summarization will take somewhat longer, but still be compatible with the needs of the individual.	A tool that does not provide adequate response times impairs productivity and results in a frustrated user.
30. The tool shall support the following capabilities: copy, cut, paste, resize, redo and undo operations for graphical and textual artefacts. This capability shall be available within and across metrics and also across external applications.	These capabilities contribute to the overall usability of the tool.
31. Shall provide a WYSIWYG interface.	A tool that forces the user to deal with two or more representations of the same artefact impairs productivity and results in a frustrated user.
32. Inputted data shal l reflect immediately and automatically on the current display. User initiated refresh operations shall not be required as part of the normal operation but only after massive changes.	A tool that forces the user to constantly issue a "refresh" command of some kind to see the results of what he/she just finished doing impairs productivity and results in a frustrated user.

33. Shall provide feedback to eac h user initiated action.	A tool that does not provide immediately feedback to every user initiated action, results in the user "clicking" or reinitiating the action many times, usually with disappointing results.
34. Operations that may result in the potential lost of data, i.e. exiting without saving, deletes and rewrites, shall ask for user confirmation.	These capabilities contribute to the overall usability of the tool.
35. Shall provide co mprehensive on-line help consistent with the style guidelines of the underlying Windows Manager.	These capabilities contribute to the overall usability of the tool.
36. Shall provide adequate on -line and printed documentation, i.e.: tutorial, user and administrator guide, and reference manual.	These capabilities contribute to the overall usability of the tool.
37. A qua lified metrics administrator shall be able to learn how to define metrics, print reports and check models on its own in not more than 16 hours.	This is an indicator of the overall usability of the tool and a big contributor towards low operating costs.
38. The tool shall provide an auto save capability from which is possible to recover the work done during the last session after a system crash.	These capabilities contribute to the overall usability of the tool.
39. Shall be possible to defi ne templates for metrics definitions.	Ease of use and consistency
40. The tool shall provide the capability to define templates for reports that could be invoked in demand.	Ease of use and consistency
41. The tool shall p rovide personalized views, which include common referenced metrics and queries.	Ease of use and consistency
42. The tool shall provide role -based views, which include common referenced metrics and queries.	Ease of use and consistency
43. The tool shall pr ovide a wizard to guide a novice user through some simple tasks.	Ease of use and consistency

44. Metrics fro m across the organization shall be made accessible from a common access point.	This is necessary to support cooperative work among members of the team.
45. The tool shall support the definition of multiple logical databases.	To support the development of independent measurement systems within each unit.
46. The tool shall be able to grant different access privileges, i.e. read, write, baselining, etc. to different users.	Variable access privileges are necessary to ensure that only the right people can update the data, preventing others for inadvertly damaging it.
47. The tool shall provide an audit track indicating who changed what when.	These capabilities help to retrace changes made to the metrics and keep it under control. This provides a very basic CM mechanism applicable in a day-to-day basis.
48. The tool shall allo w concurrent access to its repository without loss or corruption of data.	The integrity of the data shall be assured in the presence of multiple updates.
49. Conc urrent access control shall not lock out other users from working on different metrics.	To shorten delivery times work must proceed in parallel. A locking mechanism that prevents this affects project performance..
50. The rep ository implementation shall not impose a practical limit on the number of metrics.	It ought to be possible to use the tool in projects of any size.
51. The s ystem shall not impose a practical limit on the number of users that it can support.	It ought to be possible to use the tool in projects/units of any size.
52. The tool shall provide import/export capabilities of the information stored in the repository through ASCII delimited files.	These capabilities contribute to the overall productivity by allowing the manipulation of repository data by spreadsheets, word processors and other widely available tools.
53. The tool shall provide export capabilities in popular graphics formats like: bit maps, WMF, Giff and JPEG formats.	These capabilities contribute to the overall productivity by allowing the incorporation of charts in web pages and other documents.
54. The tool shall provide an API that allows access to the data stored in the repository.	This capability allows for data integration with other tools in the support environment.
55. The tool shall provide an API that allows it to be called or call other applications.	This capability allows for control integration with other tools in the support environment.

56. The tool shall support, as applicable, the following output formats: PF, RTF, ASCII and HTML.	This capability allows for data integration with the document generation system.
57. The tool shall be available on Wintel-ESDE platforms.	Current & future operating environment.
58. The tool shall be capable to publish reports to printers as well as to web pages.	Current & future operating environment.
59. The tool shall provide access to the stored data thru the IE browser.	Current & future operating environment.
60. Shall t he tool require an external database management system, it shall be one of the following: Oracle, Sybase, SQL Server.	Current & future operating environment.
61. All tool ad ministration shall be performed from a common point.	This capability determines to the overall system integrity by limiting the number of persons who can setup and update project level structures and data.
62. Acce ss to administration utilities shall be restricted to the "Tool Administrator".	System integrity.
63. Backup and recovery shall be provided by the tool or through standard host mechanisms.	System integrity.
64. To ol upgrades shall be backwards compatible.	Minimize rework when a tool change occurs.
65. The tool securit y mechanism shall integrate with security environment.	Ease of administration.

ANNEXE IX

DESCRIPTION DES ENTITÉS DU MODÈLE MULTIDIMENTIONEL DE L'ENTREPÔT DES MESURES

1. Entity

Instances of this entity Entity store all data associated with a specific Unit, Product, Project or any other entity type defined by entity metadata. See Figure IX.1, and Table IX.1.

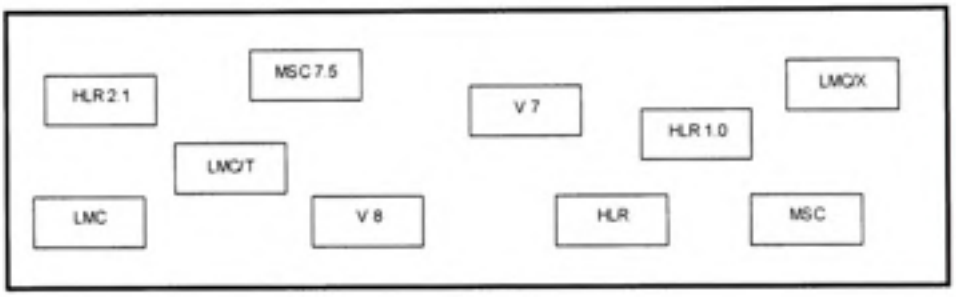

Figure IX.1 ***Entity Instances.***

Table IX.1

Entity

Attributes	Description
Entity id	Primary key
Entity Type	Link to entity metadata. Implements the relationship "describes".
Entity Name	Name of the entity being measured
Entity Parent	Link to the parent entity. Implements the "aggregates" relationship
Security Key	Value used to grant access to a particular user or group of users.

2. Entity Metadata

Instances of this entity store all common data associated with a given entity type, i.e. Unit, Product, or Project. See Figure IX.2, and Table IX.2.

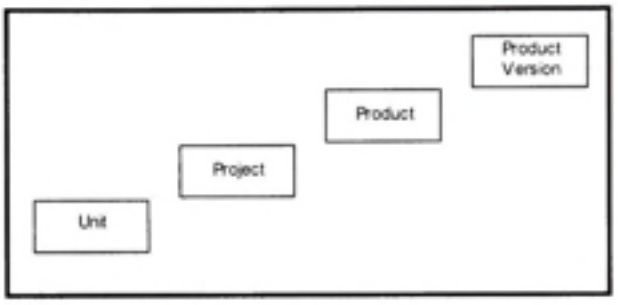

Figure IX.2 ***Entity Types.***

Table IX.2

Entity Metadata

Attributes	Description
Entity Type	Primary key
Description	Description (i.e. Project, Department, Product) corresponding to an entity of type Entity Type
Security Key	Value used to grant access to a particular user or group of users.

3. Relationship

This associative entity is used to model arbitrary relationships between two entities. The nature of the relationship is given by the relationship metadata. See Figure IX.3 and Table IX.3.

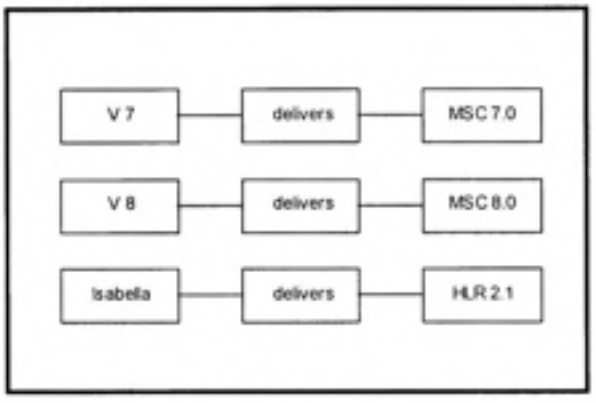

Figure IX.3 ***Relationship between siblings.***

Table IX.3

Relationship

Attributes	Description
Relationship id	Primary key
Entity Name X	Link to one the entities participating in the relation
Entity Name Y	Link to one the entities participating in the relation
Relationship Type	Points to the table describing the nature of the relation. Implements the relationship "describes".
Security Key	Value used to grant access to a particular user or group of users.

4. Relationship Metadata

Hierarchy among entity instances supported by the aggregates relationship Instances of this class store the nature of the relationship. See Figure IX.4, and Table IX.4.

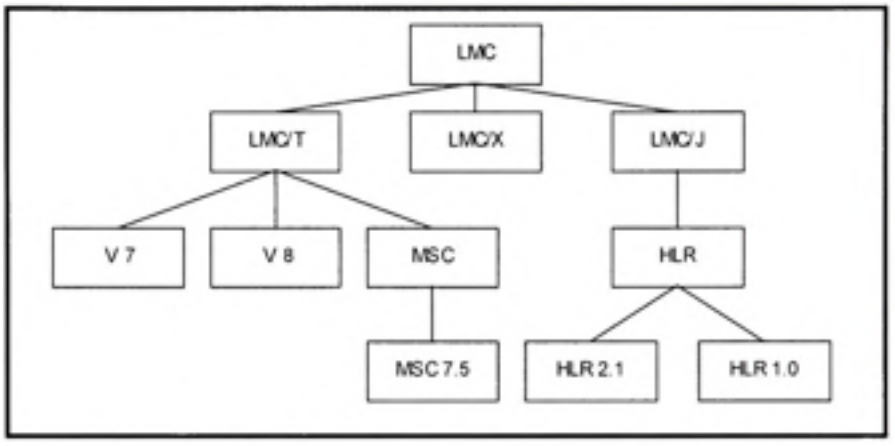

Figure IX.4 Hierarchy among entity instances supported by the aggregates relationship.

Table IX.4

Relationship Metadata

Attributes	Description
Relationship Type	Primary key
Description	Description of relationship between two entities
Security Key	Value used to grant access to a particular user or group of users.

5. Series

It is a chronologically ordered collection of measurement representing the value of a metric over time. See Table IX.5.

Table IX.5

Series

Attributes	Description
Series id	Primary key
Series Type	Defines the metric
Security Key	Value used to grant access to a particular user or group of users.

6. Series Metadata

This entity describes the metric being captured by the series. See Figure IX.5, and Table IX.6.

Table IX.6

Series Metadata

Attributes	Description
Series Type	Primary key
Category Type	Points to the category to which the metric belongs. Implements the relationship "categorized"
Series Description	Describes the metric captured by series. i.e. Open TR, BCWP, etc.
Security Key	Value used to grant access to a particular user or group of users.

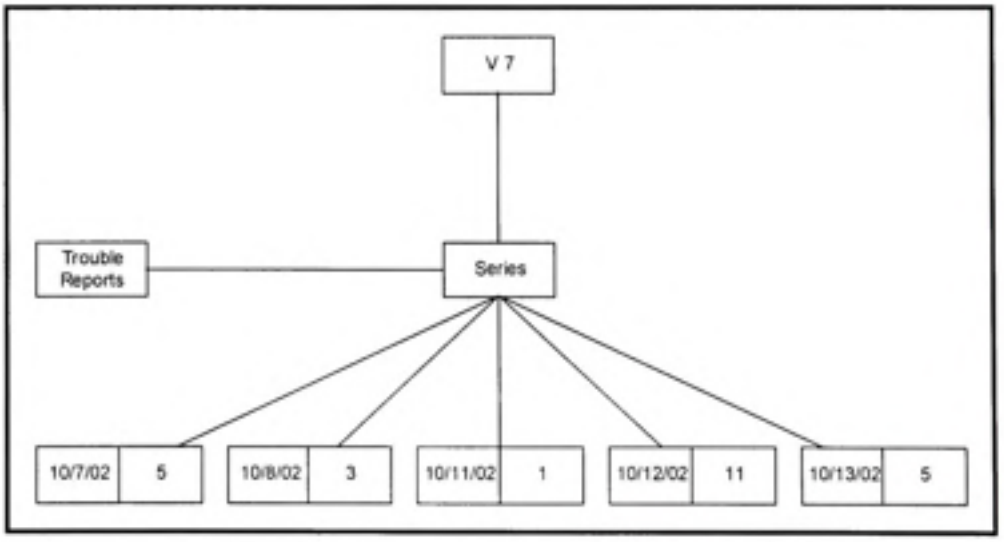

Figure IX.5 ***Chronologically ordered measurements.***

7. Measurement

Each instance of this entity captures the value of a measurement as well as the data in which it was taken. See Table IX.7.

Table IX.7

Measurement

Attributes	Description
Measurement id	Primary key
Series id	Identifies the series to which the measurement belongs. Implement the relationship "consist of"
Time Stamp	Date at which the measurement was made
Value	Value of the measurement supports ordinal, interval and ratio scales. In the case of nominal scales the value is 1
Date of First Entry	Date when the measurement was first entered into the repository
Date of Las Entry	Date when the measurement was last entered. i.e. modified
Version Number	Number of times the value was changed
Security Key	Value used to grant access to a particular user or group of users.

8. Attribute

This associative entity qualifies the measurements according to different attributes. For example, of the 5 TRs recorded on October 7, 2002 three could be severity "A", one severity "B" and the one severity "C". See Table IX.8.

Table IX.8

Attributes

Attributes	Description
Attribute id	Primary key
Series id	Link to the series to which the value being qualified belongs.
Measurement id	Points to the measurement being qualified
Value id	Points to the value the attribute adopts for a particular measurement. Implements the relationship "adopts"
Security Key	Value used to grant access to a particular user or group of users.

9. Attribute Metadata

This entity describes the attributes that classify the measurement in a series. See Table IX.9.

Table IX.9

Attribut Metadata

Attributes	Description
Attribute id	Primary key
Description	Defines the metric
Security Key	Value used to grant access to a particular user or group of users.

10. Value

Instances of this entity store the admissible values for a given attribute. See Figure IX.6, and Table IX.10.

Table IX.10

Value

Attributes	Description
Value id	Primary key
Attribute id	Points to the table describing the nature of the relation. Implements the relationship "describes".
Security Key	Value used to grant access to a particular user or group of users.

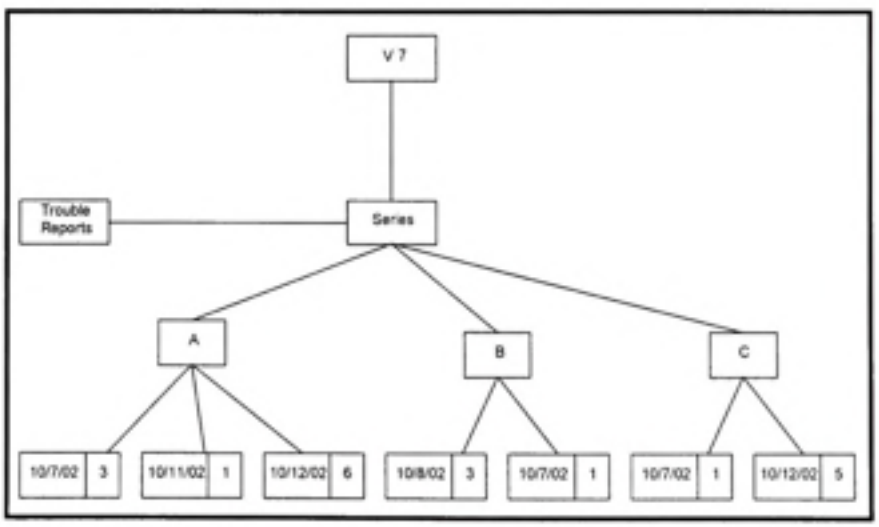

Figure IX.6 ***Measurement qualification.***

11. Category

Describes the different categories, i.e. Quality, Progress, Cost into which metrics are categorized. See figure IX.7, and Table IX.11.

Table IX.11

Category

Attributes	Description
Category id	Primary key
Category Description	Describes the category
Security Key	Value used to grant access to a particular user or group of users.

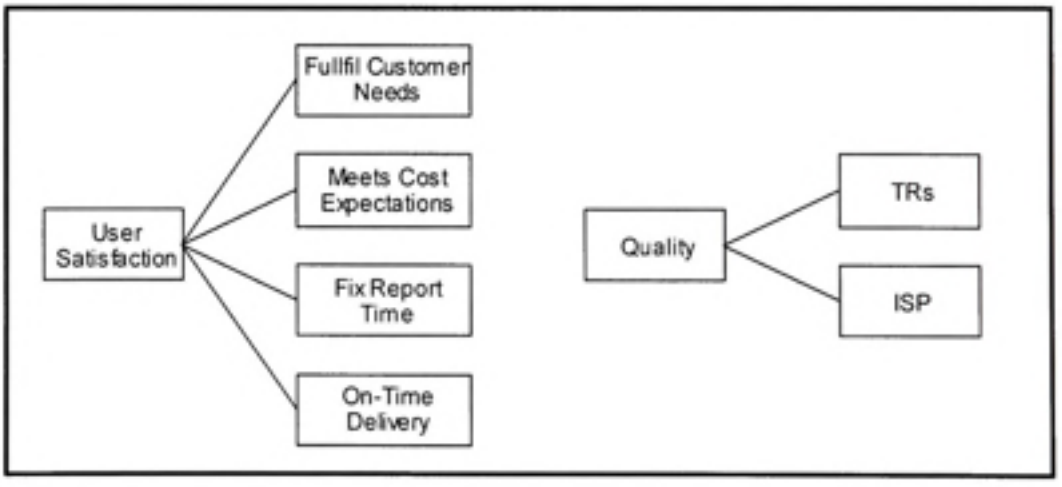

Figure IX.7 ***Categories.***

12. User

This entity captures the user-id, i.e. lmcedpa, etc. of the persons authorized to access the repository. Access to this table is restricted to the database administrator. See Table IX.12.

Table IX.12

User

Attributes	Description
User Number	Primary key
User Id	Identifies the user
Organization	Organization to which the user belongs
Security Key	Value used to grant access to a particular user or group of users.

13. Access Rights

Indicates the specific entities and relationship instances to which a given user has access and what he or she can do with them, i.e. Create, Change, Delete. Access to this table is restricted to the database administrator. See Figure IX.8, and Table IX.13.

Table IX.13

Access Right

Attributes	Description
Access id	Primary key
Table Id	Identifies the database table to which the right apply
Privileges	Permissions granted

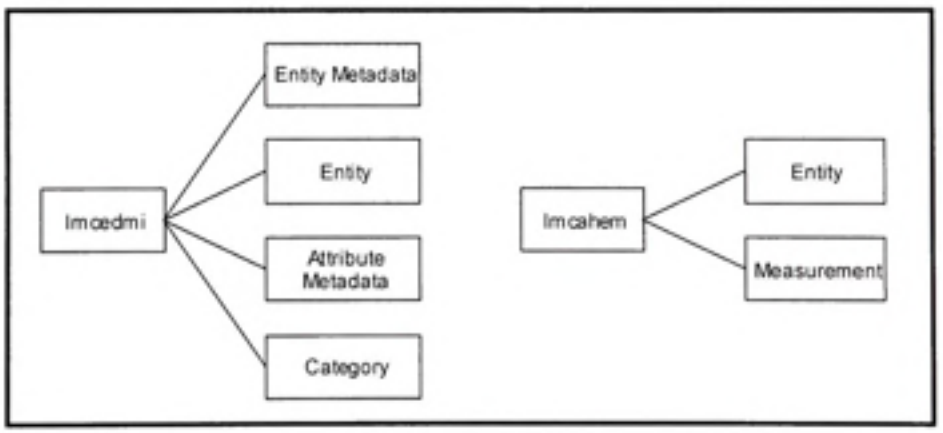

Figure IX.8 ***Security mechanisms.***

14. Applies To

This n:n relationship defines the applicable set of metrics for each entity type. See Table IX.14.

Table IX.14

Applies To Relationship

Attributes	Description
Entity id	Points to the entity metadata. Primary key
Series Type	Points to the series metadata. Primary key
Security Key	Value used to grant access to a particular user or group of users.

15. Measures

This n:n relationship links a specific series to the entity or entities being measured by it. See Table IX.15.

Table IX.15

Mesures Relationship

Attributes	Description
Entity Name	Link to the entity being measured. Primary key
Series id	Lint to series. Primary key
Security Key	Value used to grant access to a particular user or group of users.

ANNEXE X

DESIGN ET CONSTRUCTION DE L'ENTREPÔT DES MESURES

1. Designing the data-mart

1.1. Use Case diagrams

In this section we present the more relevant uses cases included in the Performance **Measurement Repository construction.**

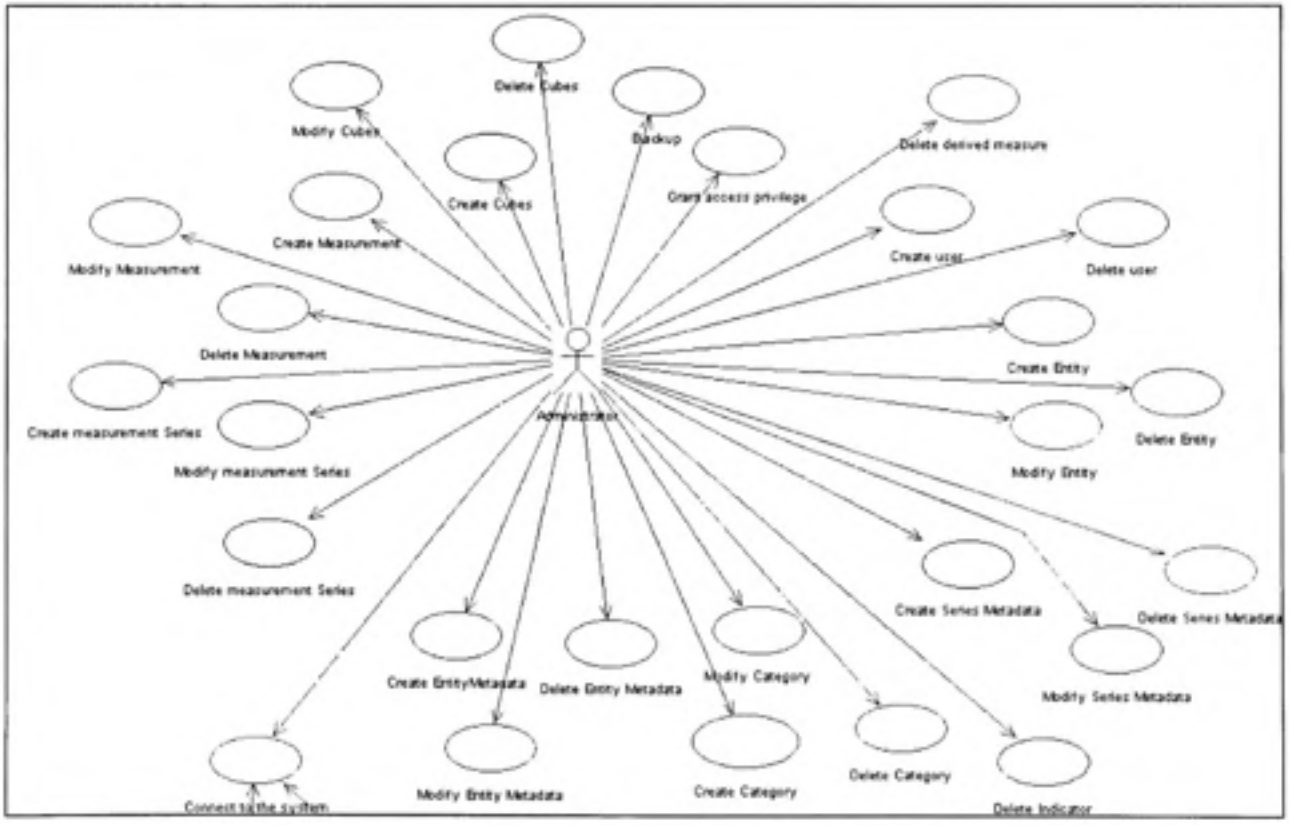

Figure X.1 ***Uses case Administrator.***

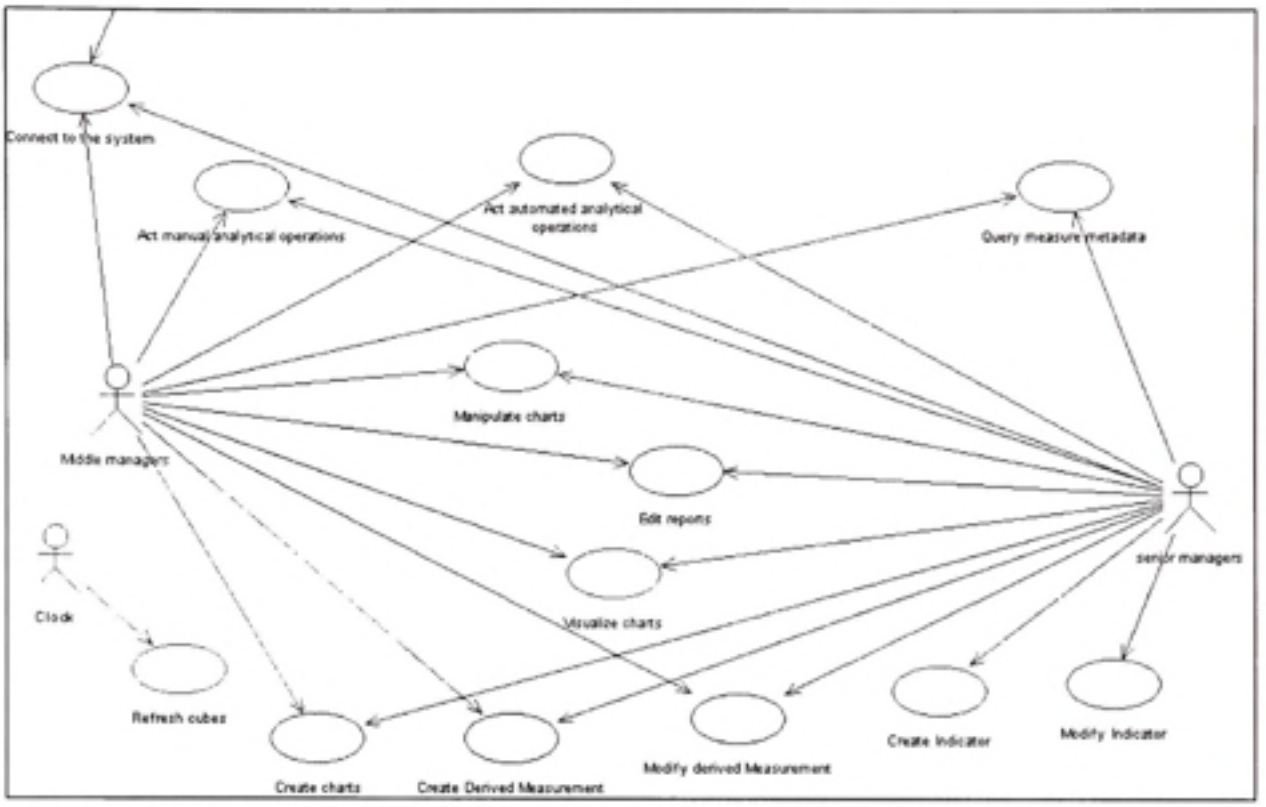

Figure X.2 ***Uses case Managers – Clock.***

1.2. Sequence diagrams

In this section we present the most relevant sequence diagrams for the Performance Measurement Repository.

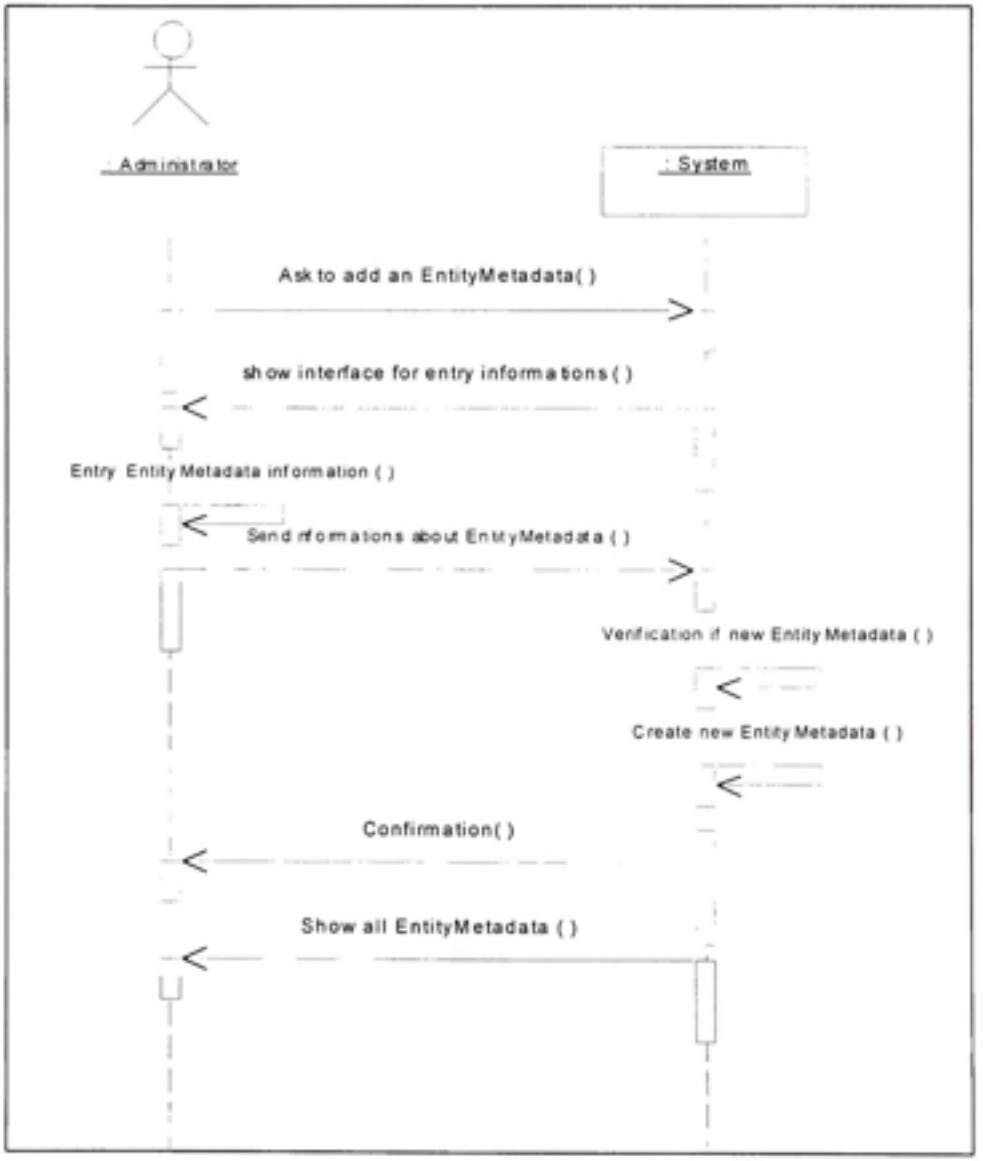

Figure X.3 ***Sequence diagram - Add an EntityMetadata.***

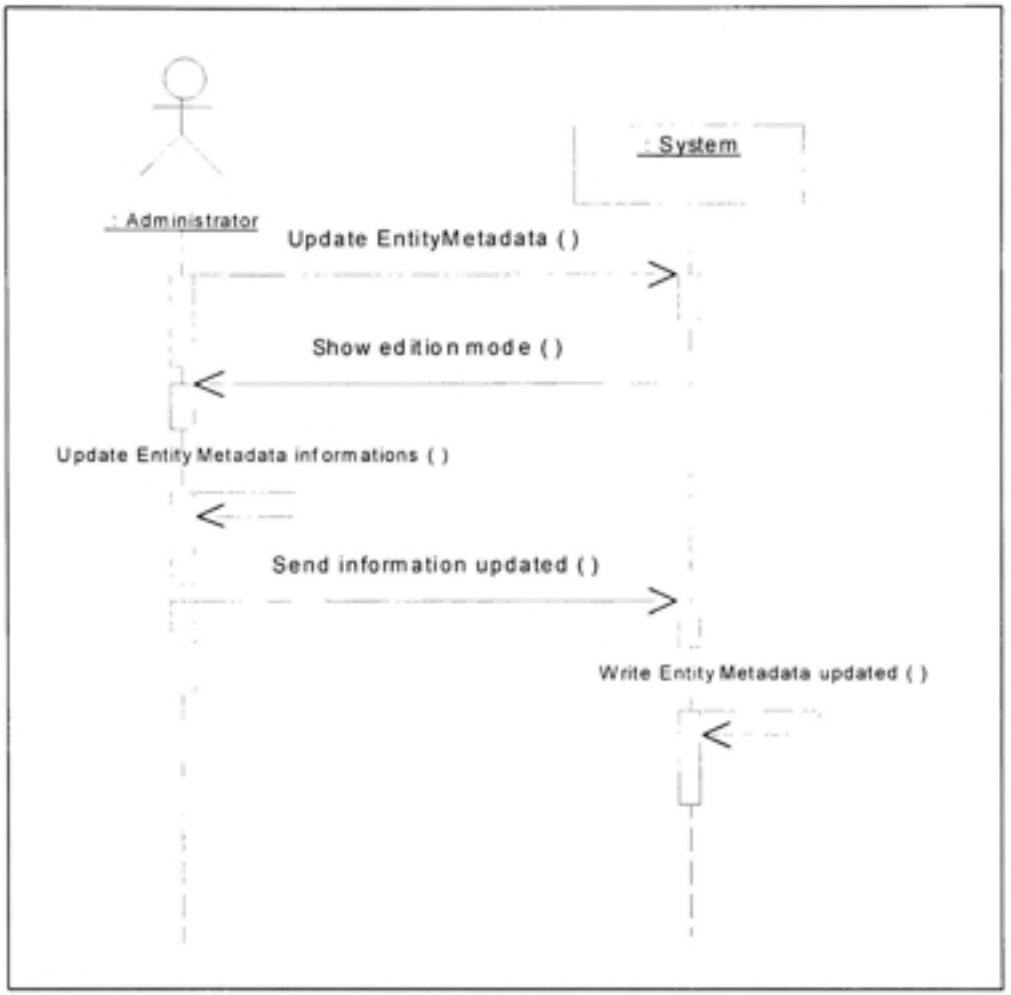

Figure X.4 ***Sequence diagram - Delete an EntityMetadata.***

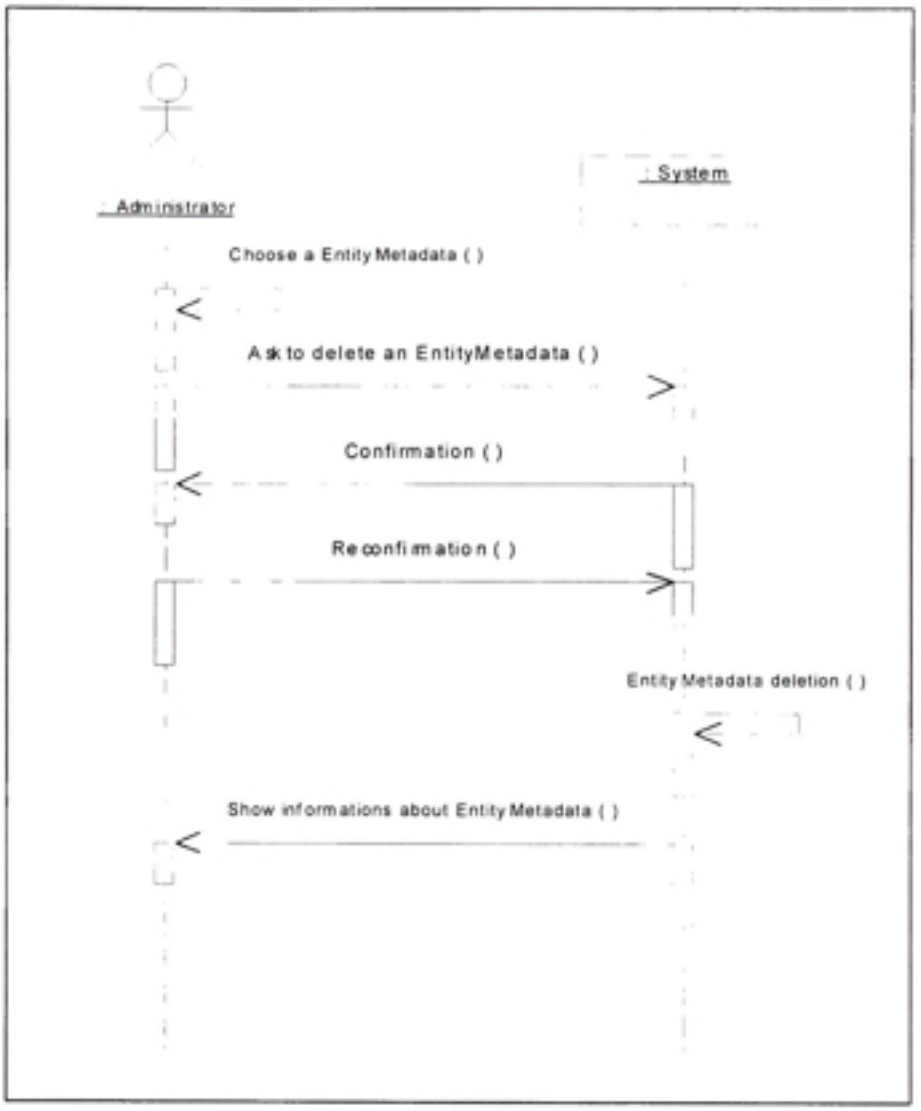

Figure X.5 ***Sequence diagram - Delete an EntityMetadata.***

1.3. Uses case definition

1.3.1. Sub process Entity Metadata

Instances of this entity store all common data associated with a given entity type, i.e. Unit, Product, or Project.

UC 1: Add Entitymetadata

Data Group: Entity Metadata.

Trigger Event: Administrator Adds an entity Metadata.

UC 2: Delete Entitymetadata

Data Group: EntityMetadata.

Trigger Event: Administrator delete an entity Metadata.

UC 3: Update Entity Metadata

Data Group: EntityMetadata.

Trigger Event: Administrator updates an entity Metadata.

1.3.2. Sub process Entity

Instances of this entity Entity store all data associated with a specific Unit, Product, Project or any other entity type defined by entity metadata.

UC 4: Add entity

Data Group: Entity.

Trigger Event: Administrator adds an entity.

UC 5: Delete entity

Data Group: Entity.

Trigger Event: Administrator deletes an entity.

UC 6: Update entity

Data Group: Entity.

Trigger Event: Administrator updates an entity.

1.3.3. Sub process Category

Instances of this entity Entity describes the different categories, i.e. Quality, Progress, Cost into which metrics are categorized.

UC 7: Add a category

Data Group: category.

Trigger Event: Administrator adds a category.

UC 8: Delete a category

Data Group: Category.

Trigger Event: Administrator deletes a category.

UC 9: Update a category

Data Group: Category.

Trigger Event: Administrator updates a category.

1.3.4. Sub process Series Metadata

This entity describes the metric being captured by the series.

UC 10: Add a Series Metadata

Data Group: Series Metadata.

Trigger Event: Administrator adds Series Metadata.

UC 11: Delete a Series Metadata

Data Group: Series Metadata.

Trigger Event: Administrator deletes a Series Metadata.

UC 12: Update a Series Metadata

Data Group: Category.

Trigger Event: Administrator updates a Series Metadata.

IX.1.3.5: Sub process Measurement

Each instance of this entity captures the value of a measurement as well as the data in which it was taken.

UC13: Add a series of measurement

Data Group: Metaseries,series,measurement,entity,measures.

Trigger Event: Administrator Adds a series.

UC 14: Delete a series of measurement

Data Group: Metaseries,series,measurement,entity,measures.

Trigger Event: Administrator deletes a series.

UC 15: Update a series of measurement

Data Group: Metaseries,series,measurement,entity,measures.

Trigger Event: Administrator updates a series.

2. Construction of OLAP cubes with MS-Analysis Services

This section presents the different aspects of the data warehouse construction: analytic & drill-down capabilities. Two measures are presented : The Earn value and the Performance Index.

2.1. OLAP Data Model hierarchy definition

The figure X.6 shows the dimension definition of an OLAP cube using MS-Analysis services.

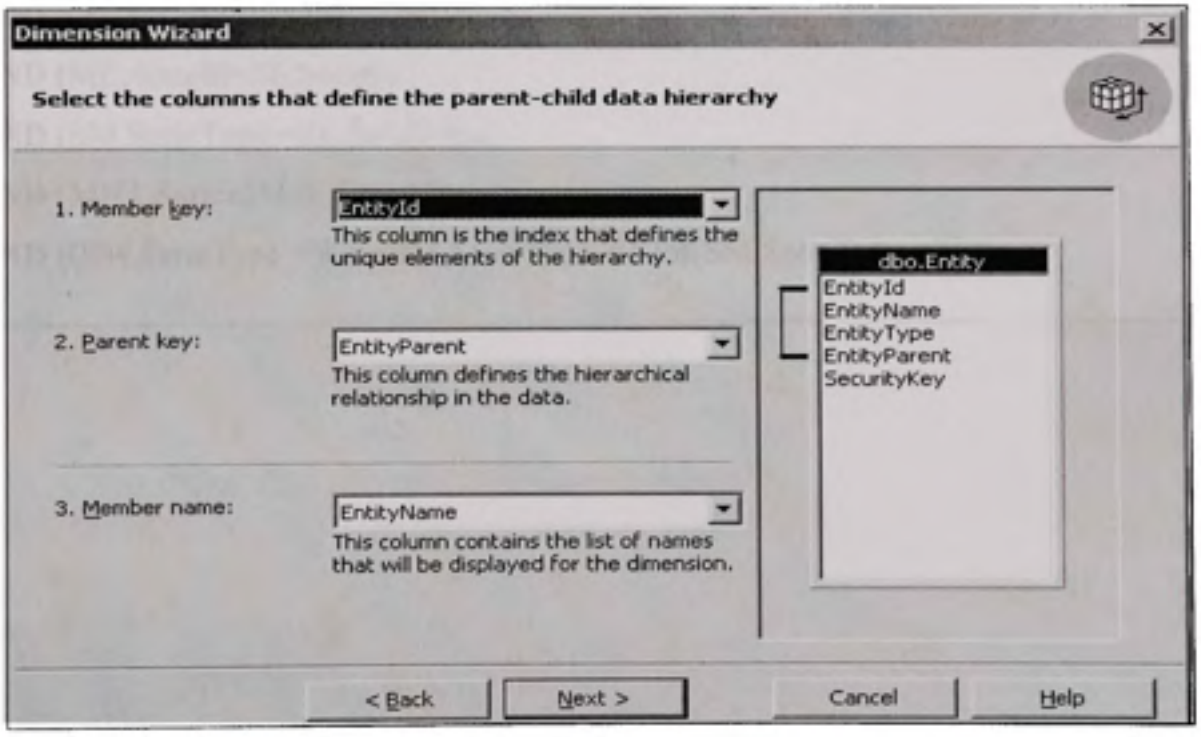

Figure X.6 ***Parent-child data hierarchy.***

2.2. OLAP Fact table definition

Fact Table Earn Value measurement

This section shows the SQL script for the fact table definition regarding the EarnED Value Measure (EVM). See Table X.1.

Table X.1

Fact Table Earn Value measurement

```
INSERT INTO Fact_Table_EarnedValue
SELECT
MNT.MeasurementID,SE.SerieType,EN.EntityId,MNT.[TimeStamp],MNT.NumericValue
FROM Entity En,Measures ME,Series SE,SeriesMetadata SM,Measurement MNT
WHERE (EN.EntityId=ME.EntityId)
AND (ME.SerieId=SE.SerieId )
AND (SM.SerieType =SE.SerieType)
AND (MNT.SerieId=ME.SerieId)
AND ((SM.SerieType ='4')or(SM.SerieType ='5')or(SM.SerieType ='6'));
```

Fact_Table_Performance Index

The SQL script for insert the data for the fact table definition. See Table X.2.

Tableau X.2

Fact_Table_Performance Index

```
INSERT INTO Fact_Table_PerformanceIndex
SELECT
MNT.MeasurementID,SE.SerieType,EN.EntityId,MNT.[TimeStamp],MNT.NumericValue
FROM Entity En,Measures ME,Series SE,SeriesMetadata SM,Measurement MNT
WHERE (EN.EntityId=ME.EntityId)
AND (ME.SerieId=SE.SerieId )
AND (SM.SerieType =SE.SerieType)
AND (MNT.SerieId=ME.SerieId)
AND ((SM.SerieType ='9')or(SM.SerieType ='10'));
```

2.2. OLAP cube definition using MS-Analysis Services

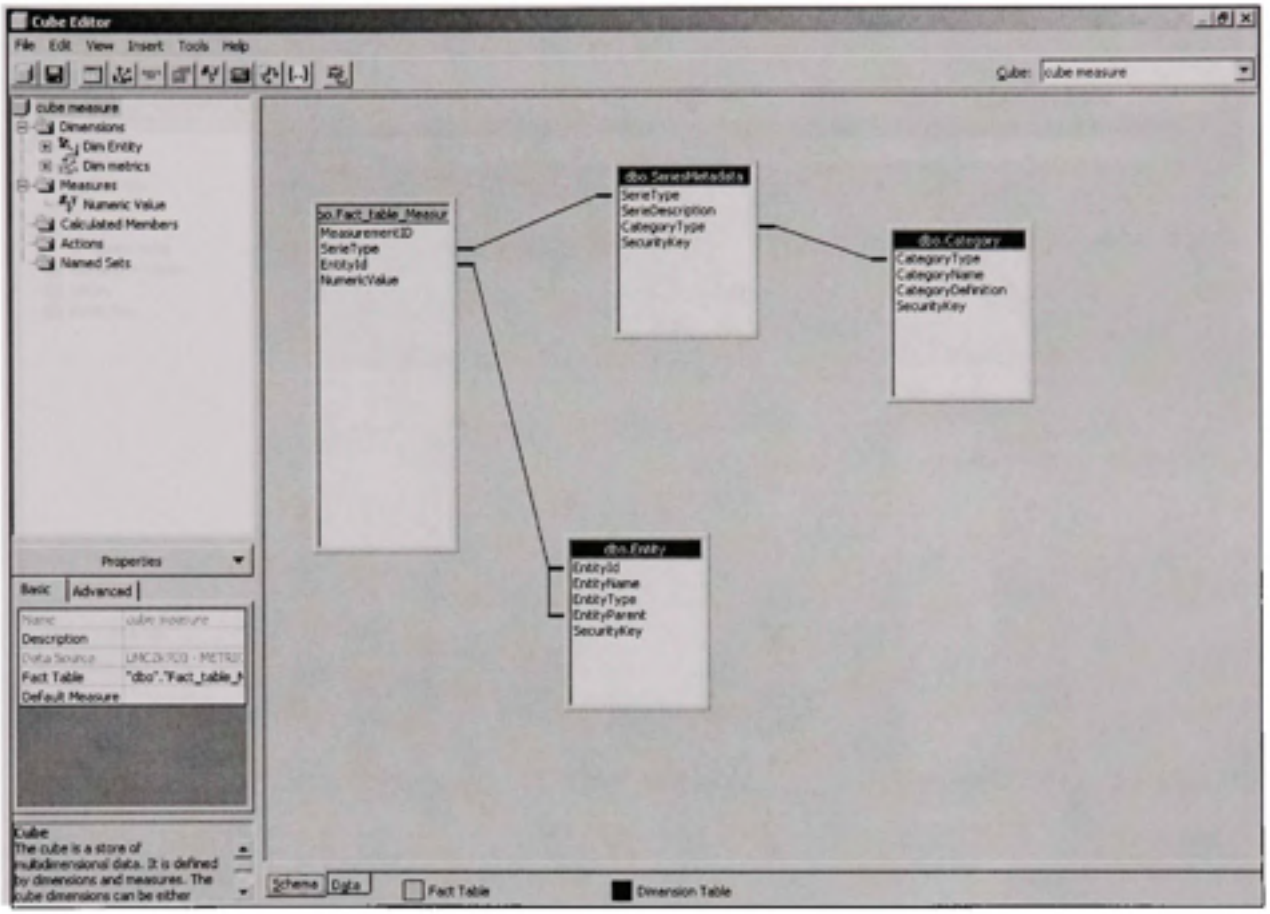

Figure X.7 ***Fact table and associations.***

OLAP cube visualization using the MS-Analysis Services

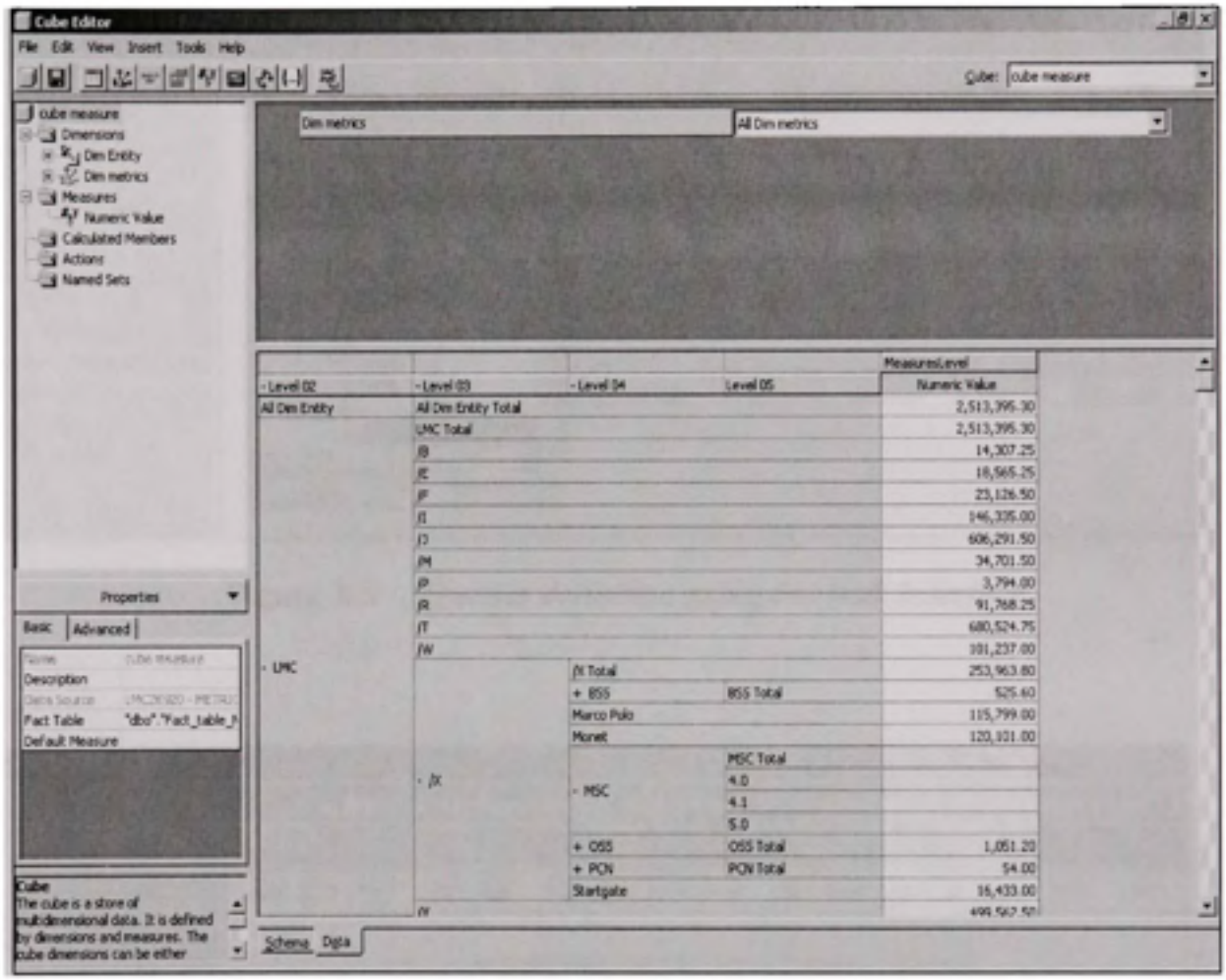

Figure X.8 ***Fact table hierarchy.***

2.4. OLAP cube visualization using the ASP Web Interface

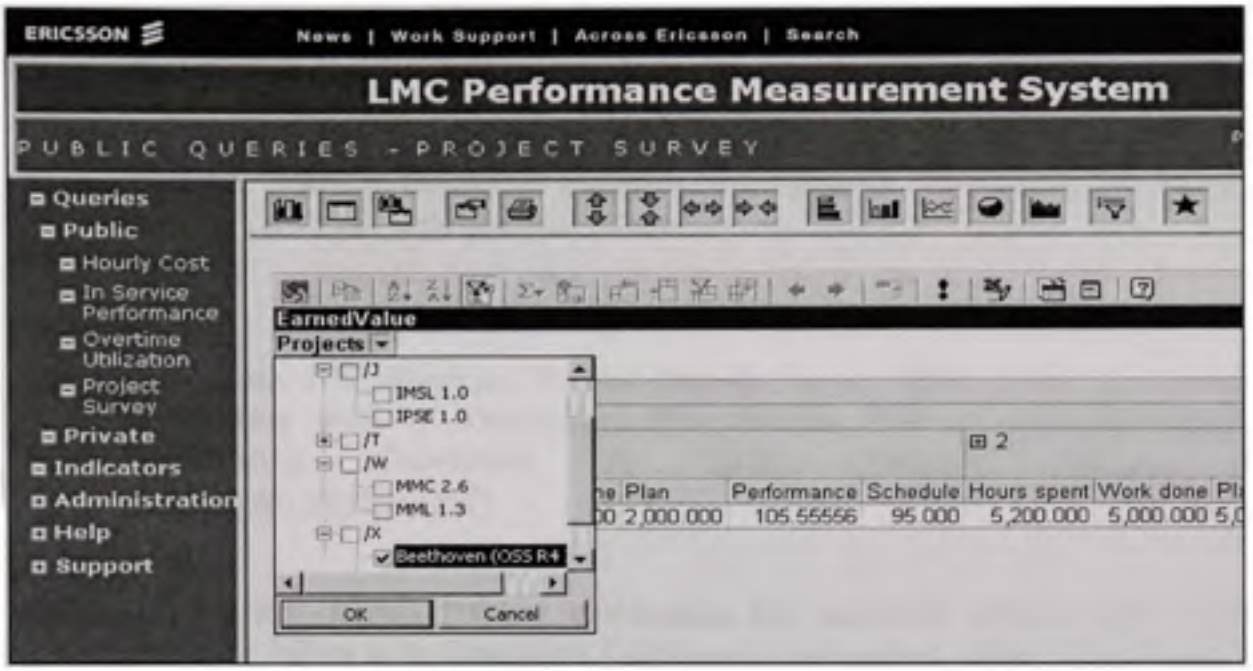

Figure X.9 *Queries definition using the Web Interface.*

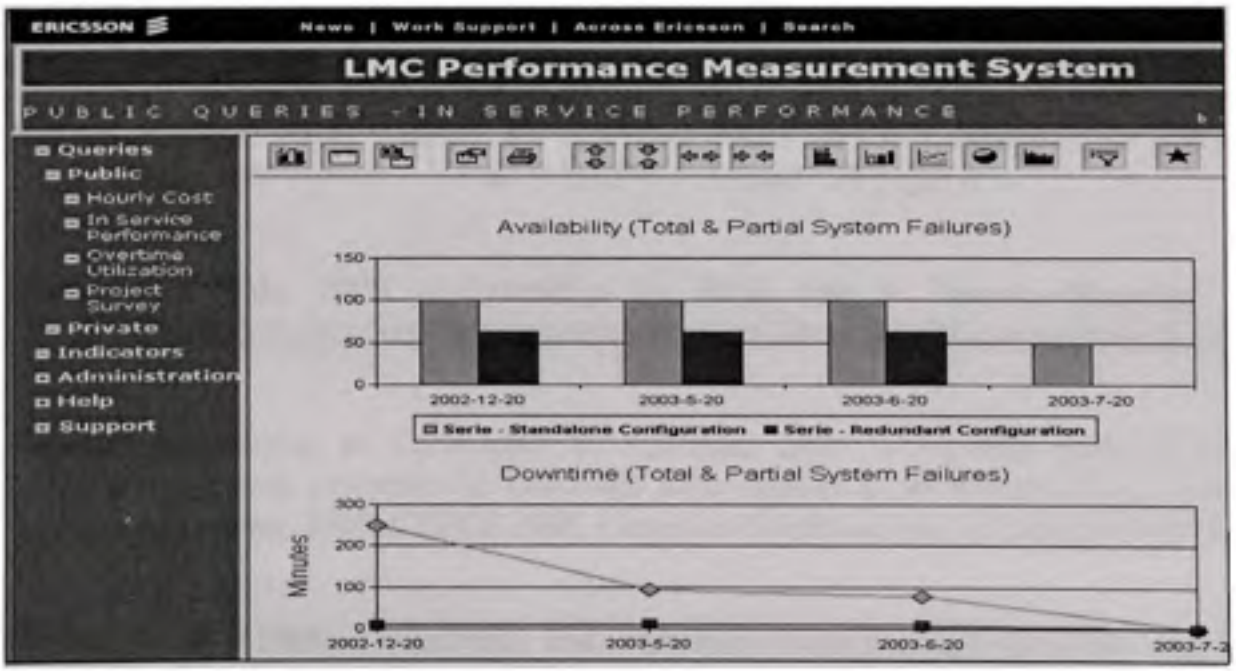

Figure X.10 *Performance measures chart.*

BIBLIOGRAPHIE

50128:2001, EN. 2001. *Railway applications. Communications, signalling and processing systems. Software for railway control and protection systems*. GEL/9.

Abran, et Palza. 2003. « Design of a Generic Performance Measurement Repository in Industry ». *13th International Workshop on Software Measurement, Montréal (Québec)*.

Abran, A., Moore, J.W., Bourque, P., and Dupuis, R., ed. 2004. *Guide to the Software Engineering Body of Knowledge, 2004 Version*. Coll. « Software Engineering Coordinating Committee, ». IEEE Computer Society Press. <http://www.swebok.org/>.

Arthur, J.D., et R.E. Nance. 2000. « Verification and validation without independence: a recipe for failure ». In *Simulation Conference Proceedings, 2000. Winter*, sous la dir. de Joines, J.A., R.R. Barton, K. Kang et P.A. Fishwick. Vol. 1, p. 859-865 vol.1. Practical.

Balduino, R. 2006. « "Introduction to OpenUP (Open Unified Process)," ». In. <www.eclipse.org/epf/general/OpenUP.pdf>.

Basque, Richard. 2004. *CMMI : Un itinéraire fléché vers le Capability Maturity Model Integration Version 1.2*. Coll. « InfoPro ». Dunod 198 pages p.

Bencomo, Alfredo. 2005. « Extending the RUP, Part 1: Process Modeling ». In. <http://www.ibm.com/developerworks/rational/library/05/323_extrup1/index.html>.

Bertrand, Christophe, et Christopher P. Fuhrman. 2008. « Towards defining software development processes in DO-178B with openup ». In *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*. p. 000851-000854.

Boehm, B. W. 1984. « Verifying and Validating Software Requirements and Design Specifications ». *Software, IEEE*, vol. 1, n° 1, p. 75-88.

Boehm, B. W. 1988. « A spiral model of software development and enhancement ». *Computer*, vol. 21, n° 5, p. 61-72.

Boehm, B.W. 1981. *Software Engineering Economics*. New York, Prentice Hall.

Bruner, Judith N. 2002. *Software Independent Verification and Validation (IV&V)*. ppt. Fairmont, West Virginia: NASA IV&V Facility.

Carolyn Salmon, et Clive Lee. 2006. *The Certification of Systems Containing Software Developed using RTCA DO-178b*. ERA Technology Ltd.

Chilenski John Joseph, et Miller Steven P. 1994. « Applicability of Modified Condition/Decision Coverage to Software Testing ». *Software Engineering Journal* vol. Vol 7, No. 5, pp 193-200.

Chris Sibbald and Kurt Sand. 2007. « Building embedded software with the Eclipse Process Framework ». In. <http://www.embedded-computing.com/departments/eclipse/2006/07/>.

CMC Electronics. 2007. « CMC Deliverables for DO-178B Certification: Software Developement Plan (SDP), Software Configuration Management Plan (SCMP), Software Quality Assurance Plan (SQAP), Software Verification Plan (SVP) ».

Codd, E.F., Codd S.B. et Salley C.T. 1993. « Providing OLAP to User-Analysts: An IT Mandate ».

Dettmer, R. 1988. « Making software safer ». *IEE Review*, vol. 34, n° 8, p. 321-324.

DoD. January 4, 1994. « 5000.59, Modeling and Simulation (M&S) Management ».

Dorsey, Cheryl 2002. *DO-178B Misconceptions*. Coll. « FAA National Software Conference ». Digital Flight.

Dowson, M. 1993. « Software process themes and issues ». In *Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the*. p. 54-62.

Dunn, W. R., et L. D. Corliss. 1990. « Software safety: a user's practical perspective ». In *Reliability and Maintainability Symposium, 1990. Proceedings., Annual*, sous la dir. de Corliss, L. D. p. 430-435.

Easterbrook, S. 1999. « Verification and validation of requirements for mission critical systems ». In *Software Engineering, 1999. Proceedings of the 1999 International Conference on*. p. 673-674.

Easterbrook, S., et J. Callahan. 1996. « Independent validation of specifications: a coordination headache ». In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 1996. Proceedings of the 5th Workshop on*, sous la dir. de Callahan, J. p. 232-237.

EN-50128. 2001. *Railway applications. Communications, signalling and processing systems. Software for railway control and protection systems*. GEL/9.

FAA. 2003. *8110.49 Software Approval Guidelines*. Coll. « AIR-120 ».

Feather, M. S., et L. Z. Markosian. 2006. « Emerging Technologies for V&V of ISHM Software for Space Exploration ». In *Aerospace Conference, 2006 IEEE*. p. 1-15.

Fuhrman, C., E. Palza et K. L. Do. 2004. « Optimizing the planning and executing of software independent verification and validation (IV&V) in mature organizations ». In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*. Vol. 2, p. 24-25 vol.2.

Fuhrman. Djlive. Palza. 2003. « Software verification and validation within the (rational) unified process ». In *Software Engineering Workshop, 2003. Proceedings. 28th Annual NASA Goddard*. p. 216-220.

Harrison, W. 2000. « A Universal Metrics Repository ». *Proceedings of the Pacific Northwest Software Quality Conference, Portland, Oregon, October 18-19, 2000.*

Haumer Peter. 2007. « Eclipse Process Framework Composer, Key concepts Part 1 and 2 ». In. <http://www.eclipse.org/epf/general/getting_started.php>.

Henderson-Sellers, et D.G. Firesmith. 2002. *The OPEN Process Framework.* Addison-Wesley Professional.

Hilderman V., et Baghi T. 2007. *Avionics certification: a complete guide to DO-178 (software), DO-254 (hardware).*

Ibrahim, Linda, et. al,. 1997. « The Federal Aviation Administration Integrated Capability Maturity Model (FAA-iCMM), ».

IEEE Std 982.2. 1988. « IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software ». In.

IEEE Std 610.12. 1990. « IEEE Standard Glossary of Software Engineering Technology ». (1990).

IEEE Std 1012. 2004. « IEEE Standard for Software Verificiation and Validation ». *IEEE Std 1012-2004 (Revision of IEEE Std 1012-1998)*, p. 0_1-110.

IEEE Std 1059. 1993. « IEEE Guide for Software Verification and Validation Plans ». (1993).

IEEE/EIA 12207. 1998. « Industry implementation of International Standard ISO/IEC 12207: 1995. (ISO/IEC 12207 standard for information technology - software life cycle processes - implementation considerations ». *IEEE/EIA 12207.2-1997.*

ISO / IEC 15504. 2004. « Information technology — Process assessment - Part 1 to 5 ».

ISO/CEI Std 90003. 2004. « Ingénierie du logiciel — Lignes directrices pour l'application de l'ISO 9001:2000 aux logiciels informatiques ». In. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC Std 9126-1 to 4. 2001. « Information Technology - Software Product Quality ». In. Geneva, Switzerland: International Organization for Standardization.

ISO/IEC Std 15939. 2007. « Information Technology - Software Engineering - Software Measurement Process ». In. Geneva, Switzerland: International Organization for Standardization.

Kornecki, Khajenoori, Thabet, Li, Chapman 1999. « Reliability Assessment through Certification Activities ». In. Bocar Raton, Florida The 10th International Symposium on Software Reability Engineering- ISSRE 99.

Kroll, Per , et Philippe Krutchten. 2003. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP* Addison-Wesley.

Kruchten, Philippe. 2000. *The rational unified process: an introduction*, 2nd. Coll. « Addison-Wesley object technology series ». Reading, MA: Addison-Wesley, xviii, 298 p.

Kunzig, R. 1997. « Europe's dream ». *Discover 18*, (May), p. 96–103.

Labrèche , P. 2008. « Conversation au tour des pratiques de V&V et des Objectifs du DO-178B chez CMC Electronique Montreal ».

Larman, C. 2005. *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development, .*

Leveson, N. G., et C. S. Turner. 1993. « An investigation of the Therac-25 accidents ». *Computer,* vol. 26, n° 7, p. 18-41.

Leveson, Nancy G. 1986. « Software Safety: Why, What, and How ». *Computing Surveys,* vol. Vol. 18, No. 2, June 1986.

Lewis, Robert O. 1992. *Independent verification and validation : a life cycle engineering process for quality software*. Coll. « New dimensions in engineering ». New York: Wiley, xxviii, 356 p.

Mayadoux, Y. 1994. « The selection of methods and tools for the procurement of safety critical software ». In *Assessment of Quality Software Development Tools, 1994, Proceedings., Third Symposium on*. p. 40-47.

NASA Software IV&V Facility. 2000. « Software Independent Verification and Validation: Program manager handbook ». In. pdf. <http://www.ivv.nasa.gov/about/tutorial/PM_Handbook_v1.pdf>.

NIST. 1993. *IDEFØ - Integrated Definition Methdod 0*. Federal Information Processing Standards Publications (FIPS PUBS),.

OMG. 2008. « SPEM 2.0 - Software & Systems Process Engineering Meta-Model Specification ».

Palza. 2005. « Facilitating Measurement Indicators In Software Improvements Projects ». *Systems & Software Engineering Review, Faculty of Systems & Computer Engineering, San Marcos University. Lima - Peru 2005.*

Palza, Abran et Fuhrman. 2003. « Establishing a Generic and Multidimensional Measurement Repository in CMMI context ». *28th Annual IEEE/NASA Software Engineering Workshop, Greenbelt, MD, USA*.

Palza, Abran et Fuhrman. 2004. « "V&V Measurements Management Issues in Safety-Critical Software" The 14th IWSM2004, International Workshop on Software Measurement, Germany, November, 8-12, 2004. ».

Palza Edgardo, Abran Alain et Fuhrman Christopher. 2004. « "V&V Measurements Management Issues in Safety-Critical Software" The 14th IWSM2004, International Workshop on Software Measurement, Germany, November, 8-12, 2004. ».

Pierce, P. 1996. « Software verification and validation ». In *Northcon/96*. p. 265-268.

Pressman, R.S. 1997. *Software Engineering: A Practitioner's Approach*. fourth edition, McGraw-Hill.

Rakitin, Steven R. 2001. *Software verification and validation for practitioners and managers*, 2nd. Boston, MA: Artech House.

Raman, S. 1999. « It is software process: key to the next millennium software quality ». In *Digital Avionics Systems Conference, 1999. Proceedings. 18th.* Vol. 1/17 pp. vol.1, p. 2.B.2-1-2.B.2-8 vol.1.

Rational Staff. 2004. « Rational Process Workbench (RPW) v2003 and v2003.06.12 ». In. <http://www.ibm.com/developerworks/rational/library/4144.html#N100D4>.

Rierson, L. K. 1998. « Using the software capability maturity model for certification projects ». In *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE.* Vol. 1, p. C24/1-C24/8 vol.1.

Rierson, L. K. 1999. « Object-oriented technology (OOT) in civil aviation projects: certification concerns ». In *Digital Avionics Systems Conference, 1999. Proceedings. 18th.* Vol. 1/17 pp. vol.1, p. 2.C.4-1-2.C.4-8 vol.1.

Robert M. O'Keefe, Osman Balci, Eric P. Smith,. 1987. « Validating Expert System Performance ». *IEEE Expert,* vol. Vol. 2(4), p. pp 81-90.

Robillard, P. N., P. Labrèche et O. Gendreau. 2007. « Peer Review As a V&V Standard Compliance Technique: An Aviation Industry Case Study ». In *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on.* p. 679-681.

Rosenberg, Linda, et Sally Godfrey. 2001. « Implementing CMMI at NASA's Goddard Space Flight Center ». In *CMMI Technical Conference.* pdf of ppt. <http://www.dtic.mil/ndia/2001cmmi/rosenberg.pdf>.

Rosenberg, Linda H. 2001. « What is Software Quality Assurance ». *Crosstalk: The Journal of Defense Software Engineering*, n° 2002.

Rosenberg, Linda H. 2002. « My path to Quality Assurance at NASA: Mission Success ». In. ppt. <www.awc-baltimore.org/programs/past_meetings/rosenberg.ppt>.

Royce, Winston W. 1970. « Managing the Development of Large Software Systems: Concepts and Techniques ». *Technical Papers of Western Electronic Show and Convention (WesCon) August 25-28, 1970, Los Angeles, USA.*

RTCA DO-178B. 1992. « Software Considerations in Airborne Systems and Equipment Certification ». (Dec. 1, 1992.).

RTCA/DO-178B. 1992. « RTCA/DO-178B: Software considerations in airborne systems and equipment certification ». (Dec. 1, 1992.).

RTCA/DO-248B. 2001. « RTCA/DO-248B: Final report for clarification of DO-178B "Software considerations in airborne systems and equipment certification" ».

Schulmeyer, G. Gordon, et Garth R. MacKenzie. 1999. *Verification and validation of modern software-intensive systems*. Upper Saddle River, NJ: Prentice Hall, xvii, 493 p.

SEI. 2006. *CMMI® for Development Version 1.2*. Pittsburgh: Carnegie Mellon University, Software Engineering Institute. <http://www.sei.cmu.edu/cmmi/models/sw-continuous.doc>.

Sellami, Asma. 2005. « Processus de vérification des mesures de logiciels selon la perspective de métrologie ». Montréal, École de technologie supérieure, PhD Thesis.

Singh, H., H. S. Bawa, A. S. Dhaliwal, T. A. Meitzler T. Meitzler et G. A. Gerhart G. Gerhart. 1995. « Management of effective verification and validation ». In *Engineering Management Conference, 1995. 'Global Engineering Management: Emerging Trends in the Asia Pacific'., Proceedings of 1995 IEEE Annual International*, sous la dir. de Bawa, H. S. p. 204-207.

Singh, R. 1999. « A systematic approach to software safety ». In *Software Engineering Conference, 1999. (APSEC '99) Proceedings. Sixth Asia Pacific*. p. 420-423.

Thomas, N. C., et E. F. Dowling. 1982. « Verification and Validation for Systems Important to Safety ». *Nuclear Science, IEEE Transactions on,* vol. 29, n° 1, p. 952-958.

Thomsen E (608 Pages). 1997. *OLAP Solutions: Building Multidimensional Information Systems*. Wiley Computer Publishing, New York, 1997.

Tribble, A. C. 2002. « Software safety ». *Software, IEEE*, vol. 19, n° 4, p. 84-85.

Vanek, Charles, et Linda Rosenberg. 2000. « IV&V at NASA: Presentation to the Software Engineering Worlshop ». In. pdf. <http://sel.gsfc.nasa.gov/website/sew/2000/topics/CharlieVanek_Slides.PDF>.

Verocel. 2005. « CD ROM: Example certification documentation and review artifacts according to DO-178B ».

W. R. Adrion, M. A. Branstad, J. C. Cherniavsky. 1982. « Validation, Verification and Testing of Computer Software ». *Computer Surveys*, vol. Vol 14(2), p. pp. 159-192,.

Walkerden, F. 1995. « A Design for a Software Metrics Repository ». *Centre for Advanced Empirical Software Research, School of Information Systems, University of New South Wales: Sydney, Australia.*

Wallace, Dolores R., et Roger U. Fujii. 1989a. *Software verification and validation : its role in computer assurance and its relationship with software project management standards*. Coll. « NIST special publication ; 500-165 ». Gaithersburg, MD
Washington, D.C.: U.S. Dept. of Commerce National Institute of Standards and Technology ; For sale by the Supt. of Docs. U.S. G.P.O., vii, 29 p.

Wallace, Dolores R., et Roger U. Fujii. 1989b. « Software Verification and Validation: An Overview ». *IEEE Software*, vol. 6, n° 3 (May 1989), p. 10.

Walters, Dan F. 2000. « Writing an effective IV&V plan ». *CrossTalk*, vol. November.

Williamson, G. F. 1997. « Software safety and reliability ». *Potentials, IEEE*, vol. 16, n° 4, p. 32-36.

Y.Wang and A. Bryant. 2002. « Process-Based Software Engneering: Building the Infrastructures ». *Annals of Software Engineering*, vol. Vol. 14, p. pp. 9-37.

Zelinski, L. K. 1995. « Constructing independent verification and validation life cycles using process kernels ». In *Computer Assurance, 1995. COMPASS '95. 'Systems Integrity, Software Safety and Process Security'. Proceedings of the Tenth Annual Conference on*. p. 233-241.

Zelkowitz, Marvin V., et Ionna Rus. 2001. « Understanding IV&V in the Safety Critical and Complex Evolutionary Environment: The NASA Space Shuttle Program ». In *23rd International Conference on Software Engineering (ICSE'01)* (May 12 - 19, 2001). p. 349. Toronto, Canada: IEEE.

Zubrow David. 2007. « Can you Trust your Data?, measurement and analysis infrastructure diagnosis ». *Software Engineering Institute - SEPG*