



Formalisation des modèles de la méthode MACAO et réalisation d'un outil de génie logiciel pour la création d'interfaces homme-machine.

Ferry Nicolas

► To cite this version:

Ferry Nicolas. Formalisation des modèles de la méthode MACAO et réalisation d'un outil de génie logiciel pour la création d'interfaces homme-machine.. Génie logiciel [cs.SE]. Université Paul Sabatier - Toulouse III, 2008. Français. <tel-00647482>

HAL Id: tel-00647482

<https://tel.archives-ouvertes.fr/tel-00647482>

Submitted on 2 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Université Paul Sabatier*
Discipline ou spécialité : *Informatique*

Présentée et soutenue par *Nicolas FERRY*
Le 26 Juin 2008

Titre

*Formalisation des modèles de la méthode MACAO et
réalisation d'un outil de génie logiciel pour la création
d'interfaces homme-machine.*

JURY

Président : *Bernard COULETTE – Professeur, Université Toulouse-2*
Rapporteur : *Jean CAELEN – Directeur de recherche au CNRS, Grenoble*
Rapporteur : *Patrick GIRARD – Professeur, Université de Poitiers*
Rémi BASTIDE – Professeur, CUFR J.F. Champollion
Jean-Bernard CRAMPES – Professeur émérite, Université Toulouse-2
Philippe MAGNIN – Capgemini France, Grenoble

Ecole doctorale : *MITT*
Unité de recherche : *équipe MACAO à l'IRIT*
Directeur(s) de Thèse : *Jean-Bernard CRAMPES*

Université de Toulouse
Laboratoire IRIT et Capgemini Sud

Ecole Doctorale MITT
Ministère de la recherche ANRT

Thèse CIFRE, Année : 2008

Thèse de DOCTORAT
Discipline : Informatique

présentée et soutenue publiquement par
Nicolas Ferry

**Formalisation des modèles de la méthode MACAO
et réalisation d'un outil de génie logiciel pour la
création d'interfaces homme-machine.**

Soutenue le Jeudi 26 Juin 2008 devant le jury composé de :

Président	Bernard COULETTE	Professeur, Université Toulouse-2
Rapporteur	Jean CAELEN	Directeur de recherche au CNRS, Grenoble
Rapporteur	Patrick GIRARD	Professeur, Université de Poitiers
	Rémi BASTIDE	Professeur, CUFR J.F. Champollion
	Jean-Bernard CRAMPES	Professeur émérite, Université Toulouse-2
	Philippe MAGNIN	Capgemini France, Grenoble

Directeur de Thèse : Professeur émérite, Jean-Bernard CRAMPES
Laboratoire IRIT – équipe MACAO – IUT de Blagnac,
1 place Georges BRASSENS, BP 60073 31703 Blagnac Cedex, France.

Directeur de Laboratoire : Professeur, Bernard COULETTE
Laboratoire IRIT – équipe MACAO – Maison de la recherche de Toulouse le Mirail,
5 allée Antonio Machado, 31058 Toulouse Cedex, France.

Responsables Entreprise unité « ADC » : François TRICOT, Pascale HAMET
Capgemini Sud, 15 Avenue du Docteur Maurice Grynfolgel BP1155
31036 Toulouse Cedex, France.

**Formalisation des modèles de la méthode MACAO et
réalisation d'un outil de génie logiciel pour la création
d'interfaces homme-machine.**

**MACAO HCI Models formalisation and Software
platform for the conception of human computer
interfaces.**

Nicolas Ferry

Si tu peux voir détruit l'ouvrage de ta vie
Et sans dire un seul mot te mettre à rebâtir,
Ou perdre d'un seul coup le gain de cent parties
Sans un geste et sans un soupir ;
Si tu peux être amant sans être fou d'amour,
Si tu peux être fort sans cesser d'être tendre
Et, te sentant haï, sans haïr à ton tour,
Pourtant lutter et te défendre ;

Si tu peux rencontrer Triomphe après Défaite
Et recevoir ces deux menteurs d'un même front,
Si tu peux conserver ton courage et ta tête
Quand tous les autres les perdront,
Alors les Rois, les Dieux, la Chance et la Victoire
Seront à tout jamais tes esclaves soumis
Et, ce qui vaut bien mieux que les Rois et la Gloire,
Tu seras un homme, mon fils.

Extrait de /Si.../ de Rudyard Kipling

**Je dédicace cette thèse à Johanne modèle de force et d'amour à jamais dans
mon coeur.**

Résumé

Cette thèse s'inscrit dans le domaine de l'ingénierie des Interfaces Homme-Machine.

Elle a pour thème la conception et la réalisation des modèles d'IHM de la méthode MACAO. Nous étudions l'axe de création des interfaces utilisateur en différenciant trois niveaux de la conception à la réalisation. L'approche adoptée s'appuie sur une analyse théorique et sur des cas d'expériences pratiques des modèles. Cette démarche entre dans le cadre très en vogue des modèles et des transformations de modèles. Cette étude nous conduit à distinguer les étapes de conception d'une IHM en milieu industriel et à mettre en place un processus pour la réalisation des modèles et des maquettes dans la phase de recueil des besoins.

En étudiant sur des projets réels la conception des IHM utilisant une représentation abstraite, nous soulignons toute l'importance de traiter de l'interface utilisateur avec l'utilisateur final et l'importance de la concevoir comme un élément de l'architecture générale dès les phases de conception. Nous proposons un procédé de conception de l'architecture de l'IHM qui répond aux besoins de la conception d'une IHM. Ce processus est complété par des modèles à des niveaux de raffinement différents. Nous proposons des métamodèles de ces différentes vues de la création des IHM. Et nous proposons un outil de génie logiciel qui permet d'éditer et d'utiliser le modèle du SNI. De part ses critères et ses techniques de représentation, MACAO représente une des plus douces et progressives façons d'implémenter une interface homme-machine.

Mots-clefs : MACAO, Système interactif, système participatif, Interface homme-machine, plateforme de conception d'IHM, modèles d'architecture logicielle, squelette d'application, génération, rétro-conception.

Abstract

This thesis is in keeping with the field of the software engineering design of human-computer interfaces. It concentrates on the design and the creation of MACAO method's HCI models. The axis of HCI creation is examined by differentiating three design levels from the conceptual one to implementation. Our approach is based on a theoretical analysis and on practical statistical experiments of models used at Capgemini. This approach takes place in the well-known model transformation theory currently in use. This research has led us to distinguish the main design steps of an HCI in an industrial environment and to set a process to work out models and mock-ups in the requirement phase.

When studying HCI design in real projects using an abstract representation, it is important to consider the user interface with the final user and to design it as a major element of the whole architecture since the design phase. A process is presented for the HCI creation that takes our criteria for the HCI design into account. This process is based on models with different levels of refinement. Meta-models are used for these different views of HCI creation. And a software engineering platform has been developed to edit and create SNI models. With these techniques and models, MACAO represents one of the most flexible and progressive ways to implement an HCI.

Key-Words : MACAO, Interactive system, participative system, HCI, HCI design platform, architecture software models, application skeleton, generation, reverse-engineering.

Remerciements

Je souhaite remercier tout particulièrement Jean-Bernard Crampes mon directeur de recherche qui m'a confié la réalisation de cette thèse. Il m'a orienté dans la plupart de mes choix et m'a soutenu lors des remises en cause. Je salue sa persévérance, sa combativité et sa justesse pour tantôt me soutenir et tantôt me dynamiser pour l'aboutissement de ce projet.

Je remercie les professeurs de l'IUT de Blagnac de m'avoir accueilli dans leur établissement que j'affectionne particulièrement et qui sont restés présents et attentifs à mes demandes. Merci à Daniel Vielle, Jaco, Mr Tuffery, Jimi, Laurent, Iullan, André pour ma thèse et à Odile, Michèle, Marie-Fafa, aux secrétaires Danielle, Nicole, Michelle et aux rires d'Aline et bien sûr à tout le personnel de l'établissement et des autres départements.

Je remercie particulièrement François Tricot responsable de l'ADC de m'avoir fortement appuyé lors de la présentation de mon projet à Capgemini. De m'avoir attribuer des responsabilités dans différents projets qui allaient couvrir un travail d'analyste fonctionnel, un travail de testeur, et un travail d'architecte dans le groupe. Je salue aussi sa passion débordante pour les nouvelles technologies que je partage également. Je voudrais lui souhaiter la réussite dans son projet à Capgemini UK à Londres.

Je voudrais remercier Vincent Fabre directeur de l'unité ADI qui a accepté mon contrat CIFRE à Capgemini. Je salue ses qualités humaines et ses qualité de manager. Je le remercie aussi pour les accords de subventions faites pour le bon déroulement des séminaires et autres frais engagés pour le CIFRE.

Je voudrais remercier les personnes qui m'ont aidé et soutenu sur le projet COFATHEC. Notamment Patrick Lebertre, et Michel Spasky qui parfois tard m'ont aidé à terminer la rédaction des spécifications. Je remercie particulièrement Pascale Hamet pour son ouverture, d'esprit, sa sympathie constante et surtout sa persévérance pour faire aboutir le projet.

Je voudrais remercier l'ANRT qui permet aux jeunes chercheurs de partager une expérience unique pour l'élaboration et la concrétisation d'un projet de recherche.

Je remercie mes compagnons de thèse et amis, Laurent Goncalvez et Johann Nadalutti, pour m'avoir soutenu, écouté, et supporté dans les moments difficiles. Merci aussi à Mohamed Diouf grand philosophe parmi les grands, pour ses répliques et nos échanges d'idées. Je le remercie également d'être allé installer un de mes posters durant une conférence à laquelle je ne pouvais assister.

A tous ceux qui savent et ceux qui ne savent pas que je les aime.

A tous ceux qui ont fait le tour du grand cercle et à ceux qui vont le faire ou le refaire.

A tous ceux qui ont cru en moi et à ceux qui n'ont pas voulu y croire.

Je dédicace cette thèse aux personnes que j'aime, à mon père et à ma mère, à Johanne, et à mon cousin Jocelyn qui n'a pas eu sa bourse pour faire sa thèse d'astrophysique.

Sommaire

Introduction

Table des matières

Partie A – Contexte de l'étude.....	8
Chapitre I : Contexte de la thèse.....	9
A.I.1 - Introduction.....	9
A.I.2 - Objectifs de la thèse.....	9
A.I.3 - Démarche adoptée pour la thèse.....	11
A.I.4 - Le planning.....	12
Partie B - Espace problème.....	14
Chapitre I : La conception de logiciels informatiques.....	15
B.I.1 - Historique.....	15
B.I.2 - Modélisation	17
B.I.3 - Le prototypage.....	18
B.I.4 - La démarche.....	18
B.I.5 - Présentation de la conception de logiciels informatiques.....	20
I.5.1 - Les cycles de développement logiciel.....	20
I.5.2 - Phases d'expression et de présentation des besoins.....	20
I.5.3 - Phase d'analyse.....	20
I.5.4 - Phase de conception.....	21
I.5.5 - Phase de développement.....	21
I.5.6 - Phase de tests et de validation.....	21
I.5.7 - Phase de maintenance.....	22
B.I.6 - Les approches du génie logiciel.....	22
I.6.1 - L'approche en cascade.....	22
I.6.2 - L'approche en V.....	23
I.6.3 - L'approche en spirale.....	23
B.I.7 - Les méthodes d'analyse et de conception.....	24
I.7.1 - Etat de l'art des méthodes de conception.....	25
I.7.2 - Les méthodes ascendantes.....	26
I.7.3 - Les méthodes descendantes (ou systémiques).....	26
I.7.4 - Les méthodes orientées-objets.....	28

I.7.5 - Les méthodes orientées interface homme-machine (Agiles).....	28
I.7.6 - Quelle est la méthode idéale ?.....	30
B.I.8 - Etude des gains et limites entre cycles.....	32
I.8.1 - Tableau récapitulatif des interviews :.....	34
I.8.2 - Avantages et inconvénients des méthodes :.....	35
B.I.9 - Bilan.....	37
Chapitre II : La démarche MACAO.....	39
B.II.1 - Introduction.....	39
B.II.2 - La démarche participative et interactive MACAO.....	40
II.2.1 - Etape 1 : Analyse globale.....	42
II.2.2 - Etape 2 : Conception globale.....	42
II.2.3 - Etape 3 : Développement.....	43
II.2.4 - Etape 4 : Finalisation.....	46
B.II.3 - Récapitulatifs :.....	47
B.II.4 - Perspectives d'évolutions :.....	49
B.II.5 - Les modèles de conception.....	50
B.II.6 - Modélisation du processus.....	50
B.II.7 - Bilan.....	51
Chapitre III : La modélisation des IHM.....	52
B.III.1 - Représentation des IHM.....	52
B.III.2 - Le Schéma de Navigation des IHM (SNI).....	57
III.2.1 - Définition.....	57
III.2.2 - Objectifs.....	57
III.2.3 - Planches d'un SNI.....	57
III.2.4 - Les Unités de Dialogue (UD).....	58
III.2.5 - Les UD élémentaires (UDE).....	58
III.2.6 - Les UD composées (UDC).....	60
III.2.7 - Les éléments de routage.....	61
III.2.8 - Les commentaires.....	63
III.2.9 - Compléments de modélisation.....	63
III.2.10 - Les compléments liés aux exigences des utilisateurs :.....	63
III.2.11 - Exemple de SNI.....	65
III.2.12 - Liens avec les modèles logiques.....	66
B.III.3 - Les Schémas d'Enchaînement (SE).....	66
III.3.1 - Typologie des objets graphiques GUI.....	67
III.3.2 - Exemple de SEF.....	69
B.III.4 - Les maquettes d'écrans.....	70
B.III.5 - Bilan.....	73

Partie C - Espace solution.....	77
Chapitre I : Etat de l'art.....	78
C.I.1 - Etude de la cible et du contexte technologique.....	78
C.I.2 - Les besoins de Capgemini.....	80
C.I.3 - Définition de la thématique globale :.....	81
C.I.4 - L'insertion de la démarche MACAO.....	82
Chapitre II : Les ateliers participatifs.....	85
C.II.1 - Capture des exigences d'IHM.....	85
C.II.2 - Ateliers participatifs à Capgemini.....	86
C.II.3 - Découpage en niveaux de l'axe d'analyse des IHM.....	88
C.II.4 - Séparation du quoi-fonctionnel et du comment-Technique.....	89
C.II.5 - Architecture globale de l'axe IHM.....	90
C.II.6 - Bilan.....	93
Chapitre III : L'éditeur graphique de modèle SNI.....	95
C.III.1 - L'éditeur de SNI : VisualSNI.....	95
III.1.1 - Objectifs :.....	95
III.1.2 - Périmètre fonctionnel.....	95
III.1.3 - Description détaillée de chaque fonction prévue dans l'éditeur.....	98
C.III.2 - Maquette de l'éditeur.....	102
C.III.3 - Manipulation de SNI.....	103
III.3.1 - Exemple de SNI : le projet Sicli.....	104
III.3.2 - Construction du SNI.....	104
III.3.3 - Structure XMI des fichiers SNI.....	109
C.III.4 - Structure du métamodèle du SNI.....	109
III.4.1 - Métamodèle du SNI – Paquetage Gestion.....	112
III.4.2 - Métamodèle du SNI – Paquetage Répartition.....	116
III.4.3 - Métamodèle du SNI – Paquetage des UD.....	118
III.4.4 - Métamodèle du SNI – Paquetage Nodes.....	120
C.III.5 - Retours d'expériences en contexte réel.....	122
C.III.6 - Bilan.....	122
Chapitre IV : La conception de l'éditeur graphique de SNI.....	123
C.IV.1 - La plate-forme Eclipse comme socle.....	123
C.IV.2 - Création d'un plugin pour l'éditeur.....	124
C.IV.3 - Structure adoptée.....	126
C.IV.4 - Description du Model avec EMF.....	128
IV.4.1 - Le métamétamodèle ECORE.....	129
C.IV.5 - Le contrôleur avec GEF.....	131
C.IV.6 - Vue (Draw2D).....	134
C.IV.7 - Bilan.....	136

Chapitre V : Le générateur de maquette JSF.....	137
C.V.1 - Les transformations de modèles.....	137
C.V.2 - Création d'un générateur avec oAW sur Eclipse.....	139
V.2.1 - Description du projet.....	139
V.2.2 - Objectifs.....	139
V.2.3 - Scénario d'utilisation.....	140
V.2.4 - Périmètre de la prestation.....	141
V.2.5 - La description fonctionnelle de la solution.....	141
V.2.6 - La description technique de la solution.....	142
V.2.7 - Technologies pour la génération.....	143
C.V.3 - Structure interne du générateur.....	144
V.3.1 - Structure d'un nouveau générateur :	144
V.3.2 - Lancement d'une génération :	147
V.3.3 - Création du template des pages JSF :	149
C.V.4 - La fusion multi-modèles.....	152
V.4.1 - La fusion de modèles.....	152
V.4.2 - XPand2.....	153
C.V.5 - L'utilisation des patrons comme base de connaissances.....	154
C.V.6 - Bilan.....	154
 Partie D - Résultats.....	 167
Chapitre I : Conclusion.....	168
Chapitre II : Bilan et Perspectives.....	171
Chapitre III : Contribution.....	174
 Partie E - ANNEXES.....	 179
Chapitre I : Sondages du forum Developpez :	180
Chapitre II : Critiques et suggestions.....	181
II.1.1 - Outils informatiques actuels.....	181
II.1.2 - Critiques.....	181
Chapitre III : Spécification du module d'IHM.....	182
Chapitre IV : Etude de la cible.....	183
E.IV.1 - Etude des domaines existants.....	183
E.IV.2 - La plate-forme Microsoft .Net.....	186
IV.2.1 - Présentation du Framework.....	186
IV.2.2 - La vision de Microsoft.....	187
IV.2.3 - Evolution de Windows.....	188
IV.2.4 - Evolution de Visual Studio 2005.....	190
IV.2.5 - Vision du futur par Microsoft.....	191

THESE-MACAO	Juin 2008	4 / 266
	Nicolas FERRY	

E.IV.3 - La plate-forme JAVA.....	191
IV.3.1 - Présentation.....	191
IV.3.2 - Description Technique de JAVA.....	191
E.IV.4 - La plate-forme WebSphere d'IBM.....	192
IV.4.1 - Présentation d'Eclipse.....	192
IV.4.2 - Evolution d'Eclipse.....	193
IV.4.3 - Plugins.....	193
E.IV.5 - Les autres plates-formes disponibles.....	193
IV.5.1 - Borland.....	193
IV.5.2 - BEA Weblogic.....	193
IV.5.3 - Autres fournisseurs.....	194
E.IV.6 - Modélisation.....	194
E.IV.7 - AGL.....	195
E.IV.8 - Tableau récapitulatif des outils de modélisation du marché.....	196
Chapitre V : Analyse de l'AGL MACAO.....	199
E.V.1 - Description.....	199
E.V.2 - Les acteurs :.....	199
V.2.1 - Maîtrise d'ouvrage : MOA.....	199
V.2.2 - Maîtrise d'oeuvre : MOE.....	199
E.V.3 - Le processus et les rôles du système.....	199
E.V.4 - Description des postes de travail :.....	202
V.4.1 - Maîtrise d'ouvrage :.....	202
V.4.2 - Maîtrise d'oeuvre :.....	203
E.V.5 - Description détaillée des rôles :.....	204
V.5.1 - La Commission Informatique (CI).....	205
V.5.2 - Les Bêta-testeurs / Valideurs (BTEST).....	207
V.5.3 - Le chef de projet (CP).....	209
V.5.4 - Le fonctionnel / analyste (FONC) :.....	215
V.5.5 - L'architecte (ARCHI).....	218
V.5.6 - Le développeur (DEV) :.....	222
V.5.7 - Les testeurs (TEST).....	224
E.V.6 - Documents à produire avec MACAO.....	226
E.V.7 - Diagramme des circuits et des tâches.....	227
E.V.8 - Les Modèles utilisés par une équipe MACAO :.....	235
Chapitre VI : Etude de faisabilité.....	236
E.VI.1 - WMACAO projet sur le Web.....	236
E.VI.2 - EMACAO projet sous Eclipse.....	236
E.VI.3 - Conclusion sur le choix des deux technologies.....	236
Chapitre VII : Description des Cas d'Utilisations.....	237

E.VII.1 - UC_CLIENT : Gestion de l'organisation du client.....	237
E.VII.2 - UC_PROCESS-MACAO : Gestion du processus MACAO.....	237
VII.2.1 - UC_PROCESS_CP.....	237
VII.2.2 - UC_PROCESS_ANALYSE.....	238
VII.2.3 - UC_PROCESS_ARCHITECTE.....	239
VII.2.4 - UC_PROCESS_DEVELOPPEUR.....	239
VII.2.5 - UC_PROCESS_TESTEUR.....	240
VII.2.6 - UC_PROCESS_ERGONOME.....	240
Chapitre VIII : Cas d'utilisation impactés durant l'élaboration d'une IHM.....	241
E.VIII.1 - Architecture fonctionnelle.....	241
Chapitre IX : Réalisation de AGL en paquetages.....	242
E.IX.1 - Paquetage Projet.....	242
E.IX.2 - Paquetage Client / Organisation.....	242
E.IX.3 - Paquetage Architecture / Process.....	242
IX.3.1 - Analyse.....	242
IX.3.2 - Conception.....	242
IX.3.3 - Réalisation.....	243
IX.3.4 - Finalisation.....	243
E.IX.4 - Paquetage Documents - Traçabilité des documents.....	243

Partie A – Contexte de l'étude

THESE-MACAO	Juin 2008	8 / 266
	Nicolas FERRY	

Chapitre I : Contexte de la thèse

A.I.1 - Introduction

Les travaux exposés dans cette thèse contribuent à favoriser l'écoute, la compréhension et le suivi de la demande d'un utilisateur de logiciels informatiques. De nature pluridisciplinaires, ils s'inscrivent dans le domaine des interfaces homme-machine et dans celui de l'ingénierie des modèles. Le but est de fournir aux créateurs de logiciels informatiques, un atelier de génie logiciel permettant de mettre en oeuvre la méthode MACAO (Méthode d'Analyse et de Conception d'Applications Orientées-objets). Celle-ci présente une démarche sous la forme d'un guide qui préconise les compromis et qui privilégie le dialogue avec les utilisateurs finaux durant la réalisation de logiciels.

Cette thèse est présentée dans le cadre d'une convention CIFRE proposée par l'association nationale de la recherche technique (ANRT). La thèse en contrat CIFRE a pour but d'associer autour d'un projet de recherche trois partenaires : une entreprise, un jeune diplômé, et un laboratoire pour conduire à une soutenance d'un doctorat professionnel.

L'intérêt du génie logiciel et des méthodes de conception rejoint le besoin d'industrialisation et de standardisation des sociétés de services en informatique (SSII). Le partenariat avec la société Capgemini Sud a permis durant trois ans d'établir des liens et de jeter un pont entre le monde de la recherche et celui de l'industrie.

A.I.2 - Objectifs de la thèse

Le sujet de cette thèse consiste en « la formalisation des modèles de la méthode de conception orientée-objets MACAO et la réalisation d'un outil de génie logiciel pour la création de modèles des interfaces homme-machine ».

La méthode MACAO [1] et [180] est présentée dans l'ouvrage « Méthode d'Analyse et de Conception d'applications orientées-objets » aux éditions Ellipse (ISBN 2-7298-1424-8). Elle est le fruit d'une synthèse d'expériences passée sur le terrain avec l'utilisation de différentes méthodes existantes auxquelles elle ajoute plusieurs innovations notamment l'approche centrée utilisateur pour la conception de l'interface homme-machine. L'IHM représente la frontière entre les décisions humaines et le système d'information. Compte tenu de cette position stratégique, elle nécessite une attention particulière lors du recueil des besoins et lors de sa conception qui doit être adaptée à l'utilisateur final [148].

THESE-MACAO	Juin 2008	9 / 266
	Nicolas FERRY	

Dans le cadre du génie logiciel, parmi tous les moyens actuellement disponibles permettant de comprendre et décrire une problématique donnée, se trouvent les « modèles ». Plusieurs définitions des modèles existent, dont celle-ci : « un modèle est une simplification de la réalité suivant un point de vue ». MACAO qui est une méthode orientée-objets, préconise de recourir à un certain nombre de modèles lorsque leur utilisation s'avère nécessaire. La méthode adopte les diagrammes de la programmation orientée-objets et supporte en particulier le standard UML [3] auquel elle ajoute des modèles spécifiques aux IHM.

Jusqu'à ce jour, il n'existait pas d'outils pour dessiner, sauvegarder, imprimer et manipuler ces nouveaux diagrammes d'IHM dans un contexte industriel. Ce qui contraignait les architectes MACAO à saisir les schémas d'IHM de manière manuelle.

Le premier objectif de cette thèse a donc consisté à formaliser les diagrammes d'IHM pour permettre leur maniement et leurs échanges sur plates-formes informatiques. Cette étude nous a amenés à classifier et élaborer le formalisme à partir de la description des schémas présentés dans le livre de MACAO. Cette formalisation a donné lieu à la réalisation d'un métamodèle.

A partir de ce métamodèle, le second objectif a été de réaliser un éditeur de graphique de SNI appelé VisualSNI permettant de manipuler ces modèles dans un cadre de production industrielle.

Le troisième objectif vise à réaliser une transformation de modèle permettant de générer une maquette d'application dans une technologie donnée comblant un fossé entre la théorie et la pratique [154].

Cette thèse contribue à l'émergence de modèles pour la modélisation des IHM, et donne des conseils pour découper l'analyse de l'IHM en plusieurs niveaux suivant l'avancement du processus de développement.

Dans le processus de conception, nous verrons l'intérêt de l'utilisation de plusieurs niveaux de modèles et étudierons le passage d'un type de modèle à un autre pour raffiner la conception des interfaces.

Cette thèse montre également que les transformations de modèles peuvent aussi être mises en oeuvre de manière à prendre en compte la conception de l'IHM dans la génération de projet.

THESE-MACAO	Juin 2008	10 / 266
	Nicolas FERRY	

A.I.3 - Démarche adoptée pour la thèse

Pour le déroulement de la thèse, il a été décidé d'utiliser la méthode MACAO elle-même pour la réalisation de VisualSNI. L'approche choisie a permis de considérer Capgemini comme un utilisateur auquel nous fournissons un outil interne de génération de maquettes d'IHM. Comme le conseille la démarche Macao, nous avons décrit le rôle et les fiches descriptives des postes des intervenants dans les projets informatiques de la SSII. L'expérience des chefs de projet nous a été d'une précieuse aide, car chaque projet véhicule des expériences et des approches différentes.

Lors des alternances en entreprise, les modèles conceptuels ont pu être testés avec des clients de Capgemini. Cette immersion dans le travail concret de la SSII a permis de tenir des rôles très variés comme celui d'analyste fonctionnel, d'architecte, de développeur, et de testeur. Certains retours de conférences ont été transmis à Capgemini pour lui faire profiter des avancées de la recherche. Inversement les alternances au laboratoire ont permis d'affiner les connaissances et la conception pour coller au plus près de l'utilisation finale désirée. C'est en quelque sorte un compromis entre le domaine de la recherche et le domaine industriel.

Le procédé utilisé pour l'étude a consisté d'une part à mener une analyse des besoins à Capgemini et d'autre part à suivre les impératifs du sujet de la thèse. Les choix stratégiques ont souvent été décidés à partir de l'expérience acquise au plus près du terrain. Celle-ci nous a fourni une vision des possibles pour concevoir notre plate-forme. A partir de là, le cahier des charges et les spécifications de l'AGL global ont été rédigés.

Nous avons ciblé les technologies et les stratégies pour établir la cible de la plate-forme et définir son périmètre. Pour cela, nous avons construit un premier prototype permettant de donner un aperçu pour les équipes de Capgemini. Pendant la recherche des technologies cibles ce prototype a permis d'implémenter un DSL de manière rapide. Ainsi a-t-il été réalisé comme une extension du diagramme d'activités UML par stéréotypage graphique dans Rational Rose [42].

Ce premier prototype sous Rational Rose a été livré à Capgemini. Il offrait la capacité de modéliser les diagrammes d'IHM avec l'outil de référence de l'entreprise. Celui-ci a pu être utilisé sur plusieurs projets, et nous a permis d'acquérir les retours de l'entreprise sur l'utilisation de ces diagrammes.

Fort de cette expérience, l'axe d'IHM de la plate-forme Macao a pu être enrichi et développé; notamment, le découpage de l'analyse d'une IHM en plusieurs niveaux successifs permet de s'adapter avec souplesse à une description aux différentes étapes du processus de développement logiciel.

THESE-MACAO	Juin 2008	11 / 266
	Nicolas FERRY	

A.I.4 - Le planning

Le planning global a été spécifié dans la convention CIFRE. Dans ce qui suit, « T0 » représente la date de démarrage de la convention. De façon générale, un rapport est produit tous les six mois à partir de « T0 ». Par ailleurs, tous les ans, un rapport d'avancement a été remis à l'ANRT pour préciser l'état des travaux.

T0 + 6 mois : Fourniture d'un rapport intermédiaire R6 contenant :

- une bibliographie et un état de l'art en matière d'outils de développement.
- le cahier des charges utilisateur (CCU) et le dossier de spécification de l'AGL (DSP).

T0 + 12 mois : Rapport intermédiaire R12 contenant au moins :

- le dossier de conception global (DCG) et le plan de développement (PDV) en 2 ou 3 prototypes.
- Le contenu de la communication soumis ciblé sur les pistes de recherche.

T0 + 18 mois : Rapport intermédiaire R18 contenant au moins :

- le dossier de définition du premier prototype (DDP1)
- les communications soumises et/ou le développement théorique réalisé
- le point d'avancement de la réalisation du prototype 1

T0 + 24 mois : Rapport intermédiaire R24 contenant au moins :

- le dossier de réalisation du prototype 1 (DRP1)
- les sources et l'exécutable du prototype 1
- les communications soumises et/ou les développements théoriques réalisés

T0 + 30 mois : Rapport intermédiaire R30 contenant au moins :

- un projet de mémoire de thèse
- le dossier de définition et le dossier de réalisation du deuxième prototype (DDP2 et DRP2)
- les exécutables et les sources du deuxième prototype
- les communications soumises

T0 + 36 mois : Rapport final comprenant :

- le mémoire de thèse finalisé
- le dossier de réalisation du troisième prototype (si développement en trois prototypes)
- le dossier de réalisation de l'AGL complet (DR) tel que décrit dans le CCU.
- Les sources et les exécutables de l'AGL.

THESE-MACAO	Juin 2008	12 / 266
	Nicolas FERRY	

Partie B - Espace problème

THESE-MACAO	Juin 2008	14 / 266
	Nicolas FERRY	

Chapitre I : La conception de logiciels informatiques

B.I.1 - Historique

Le génie logiciel est le domaine dédié à l'étude, la conception et le développement de logiciels informatiques. Il est généralement défini comme un ensemble composé d'une méthode, d'outils de développement et de maintenance, et de critères d'évaluation de la qualité, permettant à un maître d'œuvre de produire un logiciel répondant aux besoins exprimés par un maître d'ouvrage.

On appelle *maître d'ouvrage* : le donneur d'ordre. Typiquement c'est le client qui commande le logiciel à développer. Il peut être l'un des futurs utilisateurs du logiciel mais ce n'est pas toujours le cas. Le *maître d'œuvre* quant à lui est le responsable de la production du logiciel. Il peut ou non participer activement à sa réalisation.

Le génie logiciel est un domaine de recherche qui a été défini du 7 au 11 octobre 1968, à Garmisch-Partenkirchen, sous le parrainage de l'OTAN. Il a pour objectif de répondre à un problème qui partait de deux constatations :

« d'une part le logiciel n'était pas fiable, d'autre part, il était incroyablement difficile de réaliser dans des délais prévus des logiciels satisfaisant leur cahier des charges. »

De nos jours, c'est plus ou moins encore le cas si bien que la production de logiciels requière des environnements de développement, comprenant toute une variété d'outils et d'approches ainsi que des méthodes et des techniques de gestion de processus. Les aspects humains au sein de l'équipe de développement et les relations que celle-ci entretient avec les commanditaires et les utilisateurs du produit jouent également un rôle important.

L'objectif du génie logiciel est d'optimiser le coût de développement du logiciel. L'importance d'une approche méthodologique a été montrée lors de la crise de l'industrie du logiciel à la fin des années 70 :

- augmentation des coûts,
- difficultés d'évolution,
- non fiabilité,
- non respect des spécifications,
- non respect des délais.

THESE-MACAO	Juin 2008	15 / 266
	Nicolas FERRY	

Les exemples suivants montrent l'ampleur de l'impact des défaillances dues au manque de méthodologie de développement :

- la sonde Mariner en route vers Vénus s'est perdue dans l'espace à cause d'une erreur de programme FORTRAN ;
- en 1972, lors d'une expérience météorologique en France, 72 ballons contenant des instruments de mesure furent détruits à cause d'un défaut dans le logiciel ;
- en 1981, le premier lancement de la navette spatiale a été retardé de deux jours suite à un problème logiciel. La navette a d'ailleurs été lancée sans que l'on ait localisé exactement le problème (mais les symptômes étaient bien délimités) ;
- le développement du compilateur PL1 de Control Data n'a jamais abouti ;
- EDF a récemment renoncé à la mise en service de nouveaux systèmes de contrôle commande de ses centrales 1400 mégawatts ;
- la SNCF a rencontré des difficultés importantes lors de la mise en service du système Socrate.
- L'explosion d'Ariane 5, le 4 juin 1996, qui a coûté un demi milliard de dollars était due à une faute logicielle d'un module dont le fonctionnement n'était pas indispensable durant le vol.

Devant une telle problématique, le génie logiciel a su évoluer et s'adapter pour faire émerger de nouvelles réponses. Globalement, une première approche semble consister en une réduction de la complexité globale. Cet adage amène à décomposer le logiciel en sous modules ou composants traitant d'une sous problématique particulière. Parallèlement, pour mieux appréhender un système, il existe des outils de modélisation qui offrent une représentation simplifiée mais néanmoins pertinente.

Une deuxième approche consiste à structurer le processus de conception pour satisfaire la qualité souhaitée pour le produit. RUP catégorise la qualité en plusieurs types et établit des critères de satisfaction pour un logiciel. Les SSII sont dans une phase qui consiste à rechercher l'industrialisation de ce processus afin de réduire les coûts de production.

La pertinence de la démarche et les documents à produire sont primordiaux. Les documents apportent le support commun de communication entre les différents intervenants chargés de la réalisation.

Pour réagir face aux changements rapides des technologies, de nombreuses méthodes adoptent des développements par prototypages. Ainsi retrouve-t-on cette manière de faire dans les cycles de vie : comme par exemple avec TTUP qui valide la plate-forme technique par un prototype avant le lancement, ou comme UP avec une itération par cycle ou bien encore les démarches par prototypes comme le RAD et MACAO.

THESE-MACAO	Juin 2008	16 / 266
	Nicolas FERRY	

Cet ensemble n'est pas exhaustif mais il permet d'introduire les notions qui nous semblent intéressantes dans les méthodes de développement actuelles. Tentons de définir plus précisément ces concepts.

B.I.2 - Modélisation

La modélisation d'un problème est une façon de faire habituelle pour tous les projets de fabrication industrielle. Il serait inconcevable de construire un nouvel avion ou une nouvelle automobile sans les avoir préalablement modélisés de différentes manières : soufflerie pour visualiser les écoulements aérodynamiques, logiciels pour calculer les structures, maquettes pour affiner le design, schémas de montage pour l'assemblage des pièces, etc.

« Un modèle est une simplification de la réalité. »

C'est une abstraction de quelque chose de réel qui permet de comprendre avant de construire. Il est plus aisé de se référer à un modèle qu'à l'entité d'origine car le modèle simplifie la réalité en offrant des points de vue et des niveaux d'abstractions plus ou moins détaillés selon les besoins. Un bon modèle prend en compte les éléments qui ont un effet important pour le problème étudié et ignore ceux qui sont hors du sujet ou présentent un aspect mineur.

Il n'y a pas de « modèle correct unique » pour une situation donnée, mais un modèle est plus ou moins adéquat selon qu'il réussit à saisir les aspects cruciaux et qu'il néglige les autres en fonction du but recherché. Le terme d'*abstraction* dans ce contexte signifie l'examen sélectif de certains aspects du problème ; c'est l'outil qui permet de délimiter notre connaissance de l'univers aux entités et aux interactions qui nous concernent dans une situation donnée.

« Un modèle ne représente qu'un aspect d'un problème. »

Chaque type de modélisation est adapté à l'étude d'un phénomène particulier (aérodynamisme, résistance des matériaux, esthétique, fabrication...). Il en va de même pour l'industrie du logiciel où des modèles différents seront utilisés pour représenter des aspects différents d'un logiciel : sa structure interne, son comportement dans le temps, ses interactions avec les utilisateurs, ses possibilités de connexion avec d'autres logiciels, etc.

« Un modèle permet de mieux comprendre un problème complexe. »

Plus le problème est complexe et plus la nécessité de le modéliser se fait sentir. Cependant, en matière de logiciels, la modélisation doit s'appliquer à tous les types de problèmes, du plus simple au plus compliqué. La nécessité de modéliser un problème simple se justifie par le fait que les logiciels ont une fâcheuse tendance à évoluer très rapidement sous la pression des utilisateurs ou de leur environnement. Un logiciel qui paraissait simple au départ peut devenir complexe au bout de quelques années. Si aucune modélisation n'en a été faite dès la première

THESE-MACAO	Juin 2008	17 / 266
	Nicolas FERRY	

version, les évolutions successives seront réalisées en *bricolant* le code d'origine et notre programme aura rapidement la consistance d'un plat de spaghettis.

« Un modèle permet de communiquer les connaissances. »

La fabrication d'un objet complexe comme un immeuble, un avion ou une automobile fait intervenir plusieurs personnes ayant des compétences différentes et complémentaires qui communiquent entre elles en utilisant les modèles mis à leur disposition. En prenant pour exemple la construction d'un immeuble, le bureau d'études produira plusieurs types de plans qui représenteront chacun une composante de l'immeuble liée à un corps de métier particulier : plan de masse pour l'obtention du permis de construire, fondations pour l'entreprise de terrassement, plan des murs porteurs, plan de charpente, schémas électriques... Il en ira de même pour l'industrie du logiciel qui met en œuvre des spécialistes des systèmes d'exploitation, de la programmation, des bases de données, des réseaux... Par ailleurs les personnes chargées de la maintenance du logiciel ne seront pas nécessairement les mêmes que celles qui l'ont développé. Il est donc indispensable qu'elles soient informées de la structure du logiciel pour pouvoir le corriger ou le faire évoluer de façon rationnelle. Cette information est communiquée au travers des modèles établis.

B.I.3 - Le prototypage

Le prototypage incrémental va permettre d'affiner la vision du système. Un prototype constitue une simplification de celui-ci mais adopte des contraintes bien réelles. Généralement, un premier prototype montre la faisabilité du système. Dans les démarches qui utilisent des prototypes multiples qui tendent vers la solution, chaque prototype est enrichi au niveau fonctionnel à chacune des itérations. Les méthodes dites « agiles » ou RAD utilisent souvent un prototypage incrémental pour créer l'application finale.

B.I.4 - La démarche

Le développement d'une nouvelle application informatique doit se faire suivant une certaine logique. En effet, il paraît normal de prendre connaissance des besoins des utilisateurs avant de coder les programmes. Par ailleurs, les ingénieurs qui doivent concevoir la solution informatique ont besoin d'être guidés dans leur travail pour effectuer dans le bon ordre toutes les tâches qui leur permettront d'aboutir au logiciel désiré. A titre d'exemple nous prendrons le cas du montage d'un meuble en kit que l'on peut acheter dans certains magasins spécialisés. Avec les pièces et accessoires est livré un plan de montage qui comporte généralement trois aspects : une liste des pièces constitutives du meuble, des schémas montrant la manière avec laquelle les pièces doivent être assemblées (les modèles) et une succession d'opérations élémentaires numérotées que le client doit exécuter dans le bon ordre (la démarche).

THESE-MACAO	Juin 2008	18 / 266
	Nicolas FERRY	

Voyons succinctement l'évolution historique :

Les techniques de programmation ont mûri et ont progressé avec les concepts phares du génie logiciel [2]. Nous pouvons citer plusieurs concepts qui ont révolutionné la façon de programmer. D'une programmation purement séquentielle dans les années 70 et principalement axée sur les traitements, nous sommes passés avec les systèmes multitâches à la programmation dite événementielle. Le paradigme de la programmation orientée-objets a permis de fusionner les données et les traitements au sein d'une même unité : la classe. Pour fournir aux développeurs des outils adaptés, de nombreux modèles et méthodes de conception et de développement tels que OOA/RD de Shlaer et Mellor, OOD de G. Booch, OMT de J. Rumbaugh, OOSE de I. Jacobson, UML et RUP de la société Rational Software, sont apparus. Les méthodes OOD, OMT et OOSE sont très orientées vers les techniques de programmation, alors qu'UML s'intéresse essentiellement à la modélisation c'est-à-dire à la représentation des différents concepts qui interviendront dans l'écriture du logiciel. RUP propose une démarche (un processus) itérative de production du logiciel s'appuyant sur la notation UML et pilotée par les cas d'utilisation.

D'une façon générale, nous pouvons dire que la conception orientée-objets a pour but de mettre en évidence les classes d'objets et leurs inter relations afin de faciliter une programmation orientée-objets qui consiste en une "modularisation" des programmes en classes d'objets.

La grande majorité des méthodes de conception utilisées à ce jour met surtout l'accent sur la structuration des classes et moins sur d'autres aspects du logiciel tout aussi importants. Par exemple la structuration de l'IHM et l'aspect Base de Données sont souvent à peine abordés dans les méthodes de conception orientée-objets. Or ces deux aspects revêtent une importance considérable dans les logiciels actuels. Il existe peu de logiciels qui ne fassent pas appel à des données enregistrées dans une base de données et pour lesquelles il faille définir une structure relationnelle. La grande majorité des logiciels proposés aux utilisateurs fonctionne en mode interactif et a donc besoin d'une IHM conviviale.

THESE-MACAO	Juin 2008	19 / 266
	Nicolas FERRY	

B.I.5 - Présentation de la conception de logiciels informatiques

Les outils de développement et de maintenance, souvent désignés par "Ateliers de Génie Logiciel" (AGL) assistent les développeurs dans la gestion d'un projet informatique.

I.5.1 - Les cycles de développement logiciel

Un projet de développement de logiciel est commandé par le *maître d'ouvrage* (ou client) qui peut éventuellement être l'un des futurs utilisateurs du logiciel.

Nous désignons par *cycle de vie* d'un projet informatique l'ensemble des phases qui se déroulent depuis le moment où le client *commande* une application auprès d'un quelconque fournisseur jusqu'à la livraison et l'exploitation de la solution par les utilisateurs.

En informatique, un cycle est constitué généralement des phases suivantes :

- La phase d'expression des besoins et de capture des exigences
- La phase d'analyse
- la phase de conception
- la phase de réalisation
- la phase de test et de validation
- la phase de maintenance

I.5.2 - Phases d'expression et de présentation des besoins

Le client qui sera le maître d'ouvrage du projet exprime ses besoins de façon plus ou moins complète auprès de l'informaticien (cette phase est souvent appelée *étude des besoins*). Ce dernier en fait une représentation sous forme de spécifications en utilisant divers outils de représentation (le dossier de spécifications en donne une présentation structurée). On peut remarquer que le client n'est pas forcément le futur utilisateur de l'application.

I.5.3 - Phase d'analyse

La phase d'analyse consiste en la construction d'un modèle du monde réel (le modèle d'analyse) qui met en évidence les propriétés importantes du problème. Le modèle d'analyse est une abstraction précise et concise du but de l'application et non de la façon dont elle sera bâtie. Les objets du modèle d'analyse doivent être les concepts du domaine d'application et non des objets informatiques tels que des tables, des fichiers, des fenêtres....

Le modèle d'analyse doit permettre :

- de structurer les idées (comprendre le problème)
- de simuler le fonctionnement du réel (maîtriser la complexité)
- d'introduire la phase de conception (trace des décisions de spécifications)

THESE-MACAO	Juin 2008	20 / 266
	Nicolas FERRY	

- de communiquer avec le client pour "valider" les spécifications (accord sur le Quoi?)
 - de communiquer entre informaticiens (équipes d'analyse et de développement)
- Il s'agit en fait d'un modèle de "comportement" du système à réaliser.

I.5.4 - Phase de conception

Elle se rapporte aux prises de décisions de haut niveau concernant l'architecture d'ensemble du système à mettre en œuvre. Le système est découpé en sous-systèmes ou *domaines techniques* fondés sur le modèle d'analyse et sur l'architecture proposée pour le logiciel. La phase de conception du système fournit un support à l'implémentation physique du système et doit traduire la solution dans une architecture réalisable en termes de domaines techniques.

Quatre catégories de domaines techniques apparaissent :

- 1 - **Domaines fonctionnels** : fonctions réalisées par l'application.
- 2 - **Domaines de services** : IHM, gestion des impressions, droits d'accès à l'application...
- 3 - **Domaines d'architecture** : mécanismes et composants architecturaux (comme la structure des données, les timers, la gestion des priorités...)
- 4 - **Domaines d'implémentation** : système d'exploitation, langage de programmation, gestionnaire de réseau et de Base de Données, bibliothèque de classes utilisées, plateforme d'exploitation.

Les catégories 1 et 2 sont directement issues de la phase d'analyse.

Les catégories 3 et 4 sont introduites dans la phase de conception du système.

I.5.5 - Phase de développement

Elle concerne l'écriture des programmes (traitements et IHM) dans les langages de programmation déterminés lors de la phase de conception, la saisie initiale des données contenues dans les fichiers ou la base de données, ainsi que la mise en place des outils et protocoles de communications sur le réseau.

I.5.6 - Phase de tests et de validation

Cette phase permet d'une part de valider le fonctionnement des traitements en vérifiant l'exactitude de tous les résultats produits en fonction des entrées, d'autre part de valider avec l'utilisateur la logique de l'application au travers de son IHM. En fonction des anomalies constatées dans le programme (bogues) et des remarques faites par l'utilisateur, différentes corrections peuvent être apportées pouvant aller jusqu'à la reprise d'une partie des phases précédentes (besoins de l'utilisateur et modèles de conception).

La mise au point des programmes réalisés par le programmeur lui-même avec ou sans l'utilisateur est souvent appelée *alpha-tests* alors que la phase de validation réalisée par l'utilisateur seul ou les futurs utilisateurs, est appelée *bêta-tests*. La phase de bêta-tests revêt une importance de plus en plus grande dans la diffusion des logiciels. Les grands éditeurs de

THESE-MACAO	Juin 2008	21 / 266
	Nicolas FERRY	

logiciels (Borland, Lotus, Microsoft...) n'hésitent plus à constituer un réseau de développeurs ou d'utilisateurs expérimentés auxquels ils adressent régulièrement les versions *bêta* de leurs nouveaux produits. Ils en attendent la remontée de critiques, remarques et autres *bogues*. Le résultat final de cette phase génère les programmes dépourvus des bogues les plus fréquemment rencontrés ou les plus dangereux.

I.5.7 - Phase de maintenance

Cette phase est en fait assez semblable à la phase de tests et validation précédente ; la principale différence réside dans sa durée. Alors que l'on pose une date butoir pour la mise en exploitation des programmes, la phase de maintenance se poursuit pendant toute la durée de vie de l'application. Les modifications apportées dans les programmes peuvent être de trois types :

- les modifications correctives consistant à corriger les nombreux bogues résiduels découverts par les utilisateurs lors de dysfonctionnements du système
- les modifications évolutives se produisant lors de modifications de l'environnement fonctionnel tel qu'un changement des besoins des utilisateurs, un changement de structure de l'organisation ou des méthodes de gestion.
- Le Refactoring est un terme anglais sous lequel sont regroupées les évolutions sans modifications des fonctionnalités.

La phase de maintenance s'arrête le jour où le logiciel n'est plus utilisé. Dans de nombreux cas, c'est la situation inverse qui est rencontrée : un logiciel n'est plus utilisé car sa maintenance n'est plus assurée (disparition de la SSII¹ ou de la personne qui était chargée de ce travail, manque de temps de la part du service informatique, impossibilité d'évolution lors d'un changement important du système informatique²).

B.I.6 - Les approches du génie logiciel

Il existe plusieurs manières (approches) pour mettre en œuvre les différentes phases présentées au paragraphe précédent. Parmi toutes les approches possibles, nous en retiendrons trois : en cascade, en "V", en spirale.

I.6.1 - L'approche en cascade

Le processus se présente sous la forme de plusieurs étapes séquentielles. Chaque étape porte sur la totalité du système. L'utilisateur ne découvre l'application qu'en tout dernier lieu ; il n'intervient pas dans le processus de conception. Dans ce type d'approche, le risque

¹ Société de Services et d'Ingénierie en Informatique

² L'apparition de la micro informatique et du mode client-serveur a souvent été la cause de tels abandons. De même, l'apparition des interfaces graphiques (GUI) ont souvent mis au rebut un certain nombre d'applications fonctionnant avec une interface texte (TUI) que les utilisateurs rejetaient car jugées peu conviviales.

d'augmentation des coûts de correction des erreurs croît avec le temps. La découverte d'une erreur d'analyse ou de conception lors des tests utilisateurs, ce qui est relativement fréquent, peut entraîner l'abandon pur et simple du projet compte tenu du coût prohibitif de la rectification.

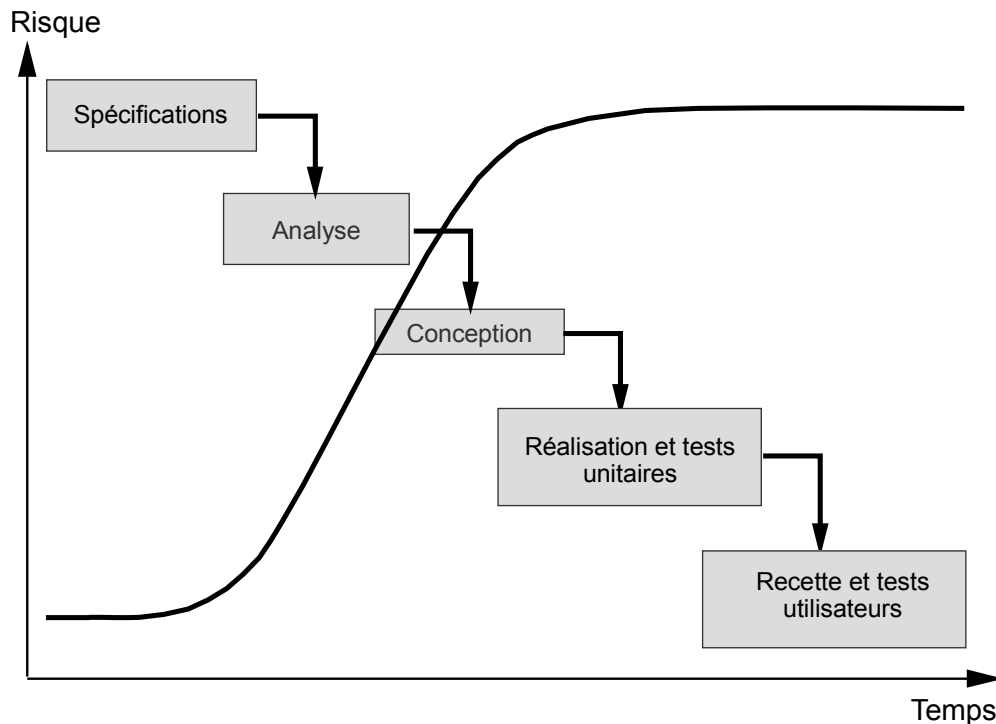


Figure 1 : Les étapes d'une approche en cascade. Remarquons l'augmentation du risque au fur et à mesure de l'avancement du projet.

I.6.2 - L'approche en V

Le système est découpé en différents sous-systèmes conçus, développés et testés de façon indépendante. Les tests réalisés sur les sous-systèmes sont faits par les programmeurs (alpha-tests). Comme pour l'approche en cascade, l'utilisateur n'intervient pas dans le processus de conception et ne valide l'application qu'à la fin lorsque le logiciel complet lui est livré. La prise de risque est donc identique à celle de l'approche en cascade.

I.6.3 - L'approche en spirale

L'application est conçue et développée par prototypages successifs en incluant dans chaque nouveau prototype des fonctions supplémentaires. L'utilisateur valide chaque prototype au fur

THESE-MACAO	Juin 2008	23 / 266
	Nicolas FERRY	

et à mesure de son codage. Le nouveau prototype est construit en intégrant les précédents. Lorsque toutes les fonctions demandées ont été intégrées et que l'utilisateur est entièrement satisfait du dernier prototype celui-ci devient l'application mise en exploitation. La prise de risque est alors limitée à la mise au point d'un seul prototype.

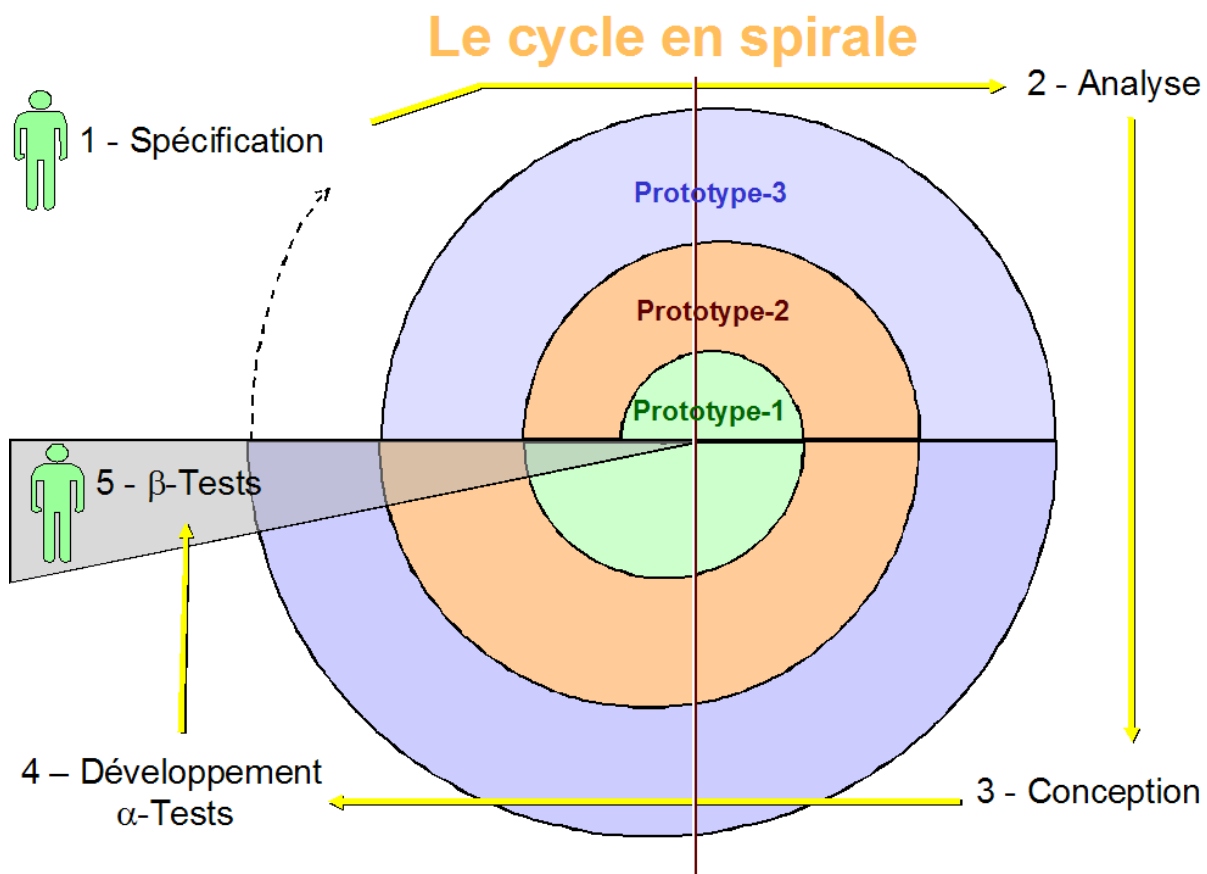


Figure 2 : Les étapes d'une approche en spirale.

B.I.7 - Les méthodes d'analyse et de conception

Il est habituel de distinguer cinq catégories de méthodes d'analyse et de conception des applications : les méthodes descendantes, les méthodes ascendantes, les méthodes orientées-objets, les méthodes agiles et les méthodes dites formelles (Z, B,...) . A chacune de ces

THESE-MACAO	Juin 2008	24 / 266
	Nicolas FERRY	

catégories on peut faire correspondre une ou plusieurs des approches précédentes. Examinons, sans entrer dans le détail, chacune de ces catégories de méthodes.

I.7.1 - Etat de l'art des méthodes de conception

Nous pouvons référencer un ensemble de méthodes qui ont percé ces dernières années. L'ensemble n'est sûrement pas complet mais dégrossit les plus importantes. Notons parmi celles-ci :

Les méthodes dites traditionnelles :

- Merise
- NIAM

Les méthodes dites objets :

- OMT (Object Modeling Technique)
- Booch
- OOSE
- SADT
- SART
- SA/SD
- FAST
- APTE
- UP
- MACAO

Les méthodes dites Agiles :

- Adaptive software development (ASD)
- Crystal clear
- Dynamic systems development method (DSDM)
- Extreme programming (XP)
- Feature driven development
- Rapid Application Development (RAD)
- Scrum

Méthodes avec notation formelle

- Méthode Z
- Méthode B
- Méthode VDM Lotos Lds

THESE-MACAO	Juin 2008	25 / 266
	Nicolas FERRY	

I.7.2 - Les méthodes ascendantes

Ces méthodes sont les plus anciennes. La première d'entre elles fut la méthode *de Blanpré* (du nom de son fondateur) utilisée à la fin des années 60 et au début des années 70.

La démarche proposée par ce type de méthode consiste à commencer par les résultats externes demandés par les utilisateurs (à l'époque, ces résultats étaient essentiellement imprimés), puis à **remonter** de manière logique pour trouver les traitements permettant de les obtenir et enfin à **remonter** à nouveau pour découvrir les données d'entrée nécessaires et suffisantes pour l'exécution de ces traitements (données saisies et données enregistrées).

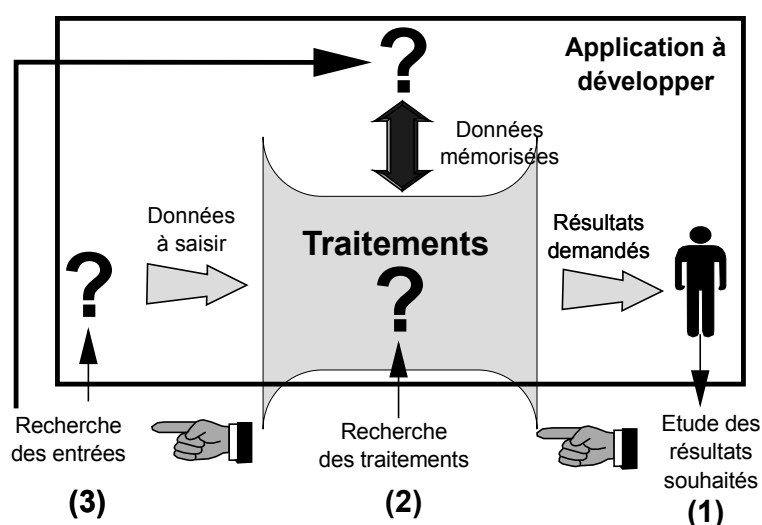


Figure 3 : Méthodes ascendantes : elles portent ce nom car le processus d'analyse remonte le déroulement logique de l'exécution des programmes. Les numéros donnent l'ordre dans lequel se déroulent les phases de l'analyse.

La seule approche possible pour ce type de méthode est l'approche en cascade, car toute la structure de l'application dépend des besoins à un instant donné de l'utilisateur. Si ces besoins changent, il faut reprendre tout le processus depuis le début. C'est essentiellement pour cette raison que ce type de méthode a été abandonné à la fin des années 70 pour faire place aux méthodes descendantes.

I.7.3 - Les méthodes descendantes (ou systémiques)

La méthode de ce type la plus connue en France est la méthode MERISE (les méthodes SADT ou SART sont également très répandues).

Le point de départ est la structure de l'entreprise considérée comme un système (système organisationnel). A partir de la structure existante, on construit un modèle de données (le MCD : Modèle Conceptuel des Données de la méthode MERISE par exemple) à partir duquel

THESE-MACAO	Juin 2008	26 / 266
	Nicolas FERRY	

on définit la structure des données qui seront enregistrées dans une base de données et qui devront être saisies. A partir de la base de données ainsi constituée et des besoins des utilisateurs, on construit les traitements.

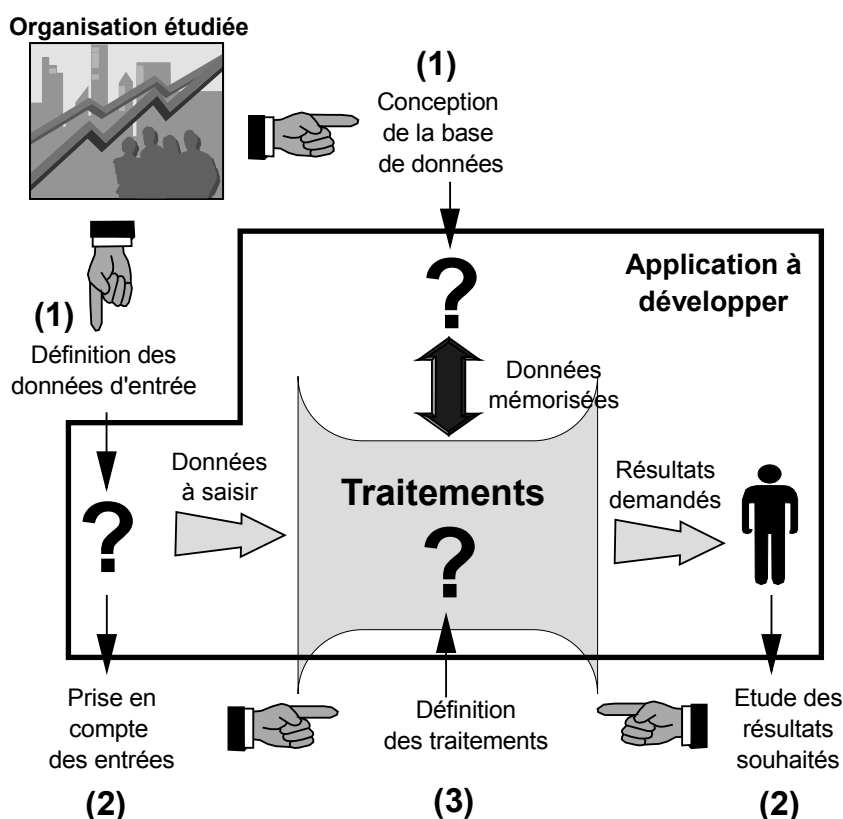


Figure 4 : Méthodes descendantes ou systémiques : on part de la structure de l'organisation pour constituer la base de données ; on demande ensuite quels sont les besoins des utilisateurs pour construire les traitements. Les numéros donnent l'ordre dans lequel se déroulent les phases de l'analyse (les numéros identiques représentent des phases réalisées en même temps).

Les méthodes descendantes ont, en fait deux inconvénients majeurs :

- Pour bien connaître la structure de l'entreprise, il est nécessaire de procéder à des interviews d'utilisateurs qui décrivent souvent leurs problèmes en terme de résultats informatiques à produire ; on se retrouve alors dans le cas d'une méthode ascendante car, c'est grâce à ces résultats que l'analyste remontera aux entrées pour structurer la base de données.

- Les données et les traitements sont définis de façon complètement séparée (indépendance données-traitements), ce qui rend très difficile la réutilisation de modules de programmes et nuit à la productivité des développeurs³.

Pour mettre en œuvre les méthodes descendantes, on peut utiliser aussi bien l'approche en cascade que l'approche en V. On peut remarquer que l'approche en V est la plus appropriée car elle se prête bien à une décomposition en sous-systèmes.

I.7.4 - Les méthodes orientées-objets

Beaucoup de méthodes sont disponibles sur le marché. Les plus connues sont OOA/RD de Shlaer et Mellor, OOD de G. Booch, OMT de J. Rumbaugh et RUP (Rational Unified Process) de Rational Software Corporation.

Les méthodes orientées-objets n'ont vu le jour qu'à partir du moment où les langages de programmation du même nom se sont répandus. En effet, très vite les programmeurs *objets* se sont trouvés démunis pour construire leurs nouvelles applications, malgré la séduction des concepts apportés par l'*objet*. Le principal intérêt de ces méthodes est de permettre une structuration des programmes en *classes d'objets* regroupant données et traitements, de façon à éliminer l'inconvénient de séparation des données et des traitements induite par les méthodes ascendantes et descendantes. Par ailleurs, la structuration des programmes en classes permet une modularité facilitant la réutilisation et la maintenance.

La mise en œuvre des méthodes orientées-objets peut être réalisée en utilisant l'une des trois approches (cascade, V ou spirale). Cependant, l'approche en spirale est la plus appropriée car chaque prototype réalisé est construit à partir d'un ensemble de classes d'objets qui peuvent être réutilisées sans modification pour construire les prototypes de niveau supérieur.

I.7.5 - Les méthodes orientées interface homme-machine (Agiles)

Dernières-nées des méthodes d'analyse, elles ont vu le jour en partant de la constatation que la partie IHM d'une application devient de plus en plus importante en taille et en qualité dans une application. En fait, la majorité des applications modernes sont construites *autour* de leur interface homme-machine et il est intéressant de faire intervenir l'utilisateur très tôt dans le processus de conception de façon à éviter des modifications importantes du programme dans le cas où l'utilisateur n'est pas satisfait de l'interface qui lui est proposée.

Partant de cette constatation, un certain nombre de SSII ont pensé qu'un développement autour de l'IHM leur permettrait d'améliorer leur productivité en morcelant et simplifiant les phases d'analyse et de conception de façon à ne prendre en compte qu'un sous ensemble limité de fonctions qui sont intégrées immédiatement aux fonctions déjà programmées. Pensant avoir enfin trouvé la solution idéale pour diminuer les coûts des projets informatiques beaucoup de SSII ont adopté ce type de méthode sous le nom de **méthode RAD** (Rapid

³ Les programmeurs sont quasi-perpétuellement en train de reprogrammer des modules qui ont pourtant déjà été programmés par eux-mêmes ou par d'autres programmeurs.

THESE-MACAO	Juin 2008	28 / 266
	Nicolas FERRY	

Application Development). Ces méthodes sont basées essentiellement sur la réalisation de prototypes de plus en plus complets (approche en spirale).

Citons aussi la méthode ExtremeProgramming (XP) [29] qui est quant à elle centrée sur les tests et vise l'intégration en continu.

Malheureusement, les méthodes RAD ont amené un certain nombre d'inconvénients qui ont souvent débouché sur l'effet contraire de celui escompté (augmentation importante des durées de développement et de maintenance des applications).

Examinons cinq de ces inconvénients :

Limitation des possibilités de développement.

Pour gagner encore plus de temps, les développeurs ont tendance à utiliser exclusivement des outils de développement visuels tels que ACCESS, Visual Basic ou autres WinDev qui ont pour objectif principal de prendre directement en charge un certain nombre de tâches de programmation sous forme de fonctions, classes d'objets ou modules préprogrammés. Tous les cas ne sont bien sûr pas prévus dans ces outils et le programmeur devra parfois soit développer certaines parties de l'application dans un langage de programmation plus classique (C par exemple avec utilisation de l'API Windows⁴), soit éliminer le cas gênant avec ou sans l'accord de l'utilisateur.

Obligation d'adopter une méthode orientée-objets pour structurer chaque prototype

Nous avons vu que l'approche en spirale n'était intéressante que si les modules développés pour le prototype de niveau "n" étaient réutilisables sans modification dans le prototype de niveau "n+1" sans quoi, on passe plus de temps à modifier les modules précédents pour les adapter qu'à programmer les nouveaux modules (le temps passé pour la réalisation d'un nouveau prototype croît exponentiellement en fonction du nombre de modules qu'il contient). La seule façon de rendre réutilisable un module de programme est de l'avoir programmé "*objets*" et d'avoir donc utilisé une méthode orientée-objets pour le structurer. Il se trouve que la maîtrise d'une méthode de conception et d'un langage de programmation orientés-objets demande une formation spécifique que n'ont pas eue la plupart des informaticiens et dont l'ampleur est souvent incompatible avec la rentabilité immédiate recherchée par beaucoup de SSII.

Nécessité d'adopter une méthode systémique préalable à la construction du premier prototype

⁴ API : Application Programming Interface. Ce sont des fonctions pré-programmées qui sont appelées dans le source d'un programme exactement comme des instructions de base pour permettre la mise en œuvre de tâches exécutées dans la couche logicielle sur laquelle s'appuie l'application (Windows par exemple).

THESE-MACAO	Juin 2008	29 / 266
	Nicolas FERRY	

Afin que l'application ait une certaine cohérence, il est absolument nécessaire d'analyser préalablement la totalité du système. Le but de la phase d'analyse est de recueillir les besoins du client. Tandis que la phase de conception a pour but de définir l'architecture générale de l'application avec la structure de la base de données, les objets métiers et la navigation générale dans l'IHM. Là encore avec une méthode RAD ces phases sont souvent bâclées voire complètement ignorées afin d'impliquer les utilisateurs au plus tôt.

Obligation d'une participation intense des utilisateurs

Pour une bonne application d'une méthode RAD, il est nécessaire d'impliquer les utilisateurs très fréquemment et de façon importante afin qu'ils donnent rapidement leur avis sur l'état du prototype, et éviter ainsi des retours en arrière trop importants. Or, il se trouve que les utilisateurs ont, par définition, autre chose à faire que de participer trop longuement au développement d'une application ; c'est d'ailleurs pour cela qu'ils font appel à des spécialistes. Les informaticiens ont d'énormes difficultés pour prendre rendez-vous avec eux et, quand un utilisateur doit réaliser par lui-même différents tests sur un prototype, les délais prévus sont souvent dépassés.

Inexistence de modèles de conception et de méthodes de réalisation des IHM

Ce dernier inconvénient est assez intéressant puisque aucune méthode d'analyse et de conception actuelle ne propose de modèle cohérent et actualisé pour représenter et concevoir une interface de qualité qui permettrait pourtant d'éviter beaucoup de perte de temps en discussions avec l'utilisateur et en tâtonnements (rappelons qu'une méthode de type RAD s'appuie sur l'IHM pour développer la totalité de l'application). Actuellement, la réalisation d'une IHM relève plutôt du système "D". Chacun réalise ses propres idées avec plus ou moins de bonheur et toujours sans aucune cohérence. On voit alors bien souvent d'excellentes applications rejetées par l'utilisateur à cause de son interface ratée (incompréhension de certains concepts, difficulté ou lourdeur de manipulation, temps de réponse importants, absence de contrôles...).

I.7.6 - Quelle est la méthode idéale ?

Nous venons de voir, au cours de ce chapitre, que tous les types de méthodes existantes comportent un certain nombre de défauts qui pourraient laisser croire qu'aucune d'entre elles n'est intéressante à utiliser. C'est bien sûr faux, car chacune possède ses propres atouts qu'il faut savoir adapter au type d'application à développer.

Si l'application ne possède aucune interactivité, il est évident qu'une méthode orientée-IHM n'a aucun intérêt. De même si l'application ne doit tourner que sur des micro-ordinateurs et se contente d'accéder à une base de données existante, il est inutile d'envisager l'utilisation d'une méthode systémique.

THESE-MACAO	Juin 2008	30 / 266
	Nicolas FERRY	

Dans le cas d'une application complexe pour laquelle beaucoup d'éléments doivent être définis : structure de la base de données, nature des communications, fonctions à réaliser, résultats à éditer, IHM..., le choix de la méthode est beaucoup plus délicat.

Nous dirons dans ce cas, que la méthode la plus appropriée est une combinaison des quatre types présentés :

- méthode ascendante pour recenser les besoins des utilisateurs et définir les données minimum à structurer,
- méthode systémique pour construire un système cohérent, possédant une bonne pérennité,
- méthode orientée-objets pour réaliser des modules réutilisables et réduire la complexité des programmes,
- méthode orientée-IHM avec une approche en spirale pour faire participer les utilisateurs au plus tôt dans le processus d'analyse-conception-réalisation et éviter ainsi des retours en arrière trop importants (nous parlerons de méthode participative).

Il est évident que la combinaison de ces méthodes devra se faire suivant un savant dosage en fonction de l'importance et du type d'application à développer ainsi que du point du cycle de vie où l'on se situe (cf. Figure 1 au §3).

La méthode MACAO propose ce type de combinaison.

THESE-MACAO	Juin 2008	31 / 266
	Nicolas FERRY	

B.I.8 - Etude des gains et limites entre cycles

Quelles sont les méthodes et les cycles de vie utilisés concrètement en entreprises ? Quelles sont les directives en matière de génie logiciel et de qualité ?

Pour les sociétés de service en informatique, les tendances du marché évoluent et définissent avec elles de nouvelles normes de certification qualité. Le nouveau standard de certification CMMi (Capability Maturity Model Integration) détermine dorénavant une norme qualité pour l'industrialisation d'un procédé de fabrication. CMMi permet notamment de classer le niveau d'industrialisation de la chaîne de production. Dans le domaine de l'informatique, il a permis de référencer aussi les outils et les meilleures pratiques employés pour produire des applications. Plusieurs méthodes sont appliquées en interne et en particulier dans le cadre de la convention CIFRE qui sous tend cette thèse, il a été décidé d'employer la méthode MACAO sur un certain nombre de projets.

Sans entrer dans les détails, CMMi définit 5 niveaux de maturité d'un processus de création :

Au **niveau 1**, le processus est empirique, **artisanal**. Les pratiques ne sont ni systématiques ni formalisées, les résultats dépendent essentiellement des acteurs, le processus est instable, les risques client sont forts, et les résultats ne sont pas maîtrisables par avance.

Au **niveau 2**, le processus est documenté, il est **reproductible**. Un système de mesures est défini, les risques client sont faibles, l'efficacité est à améliorer.

Au **niveau 3**, le processus est bien **défini**. Il est **réglé**, appliqué de façon correcte, un système de gestion et de mesure est mis en oeuvre, une capitalisation est mise en place.

Au **niveau 4**, le processus est **dirigé**, des prises de **mesures quantitatives** sont intégrées, elles sont exploitées pour prévoir et améliorer les performances du processus.

Au **niveau 5**, le processus est **optimisé**, l'**amélioration continue** du processus est intégrée dans le fonctionnement quotidien du projet (pilotage, étalonnage, amélioration).

Capgemini a préparé sa certification CMMi en 2005. Partant des besoins de la norme CMMi celle-ci nécessitait au préalable de répertorier et classer les différentes méthodes mises en place sur les projets existants. Le travail a coïncidé avec la phase de recueil des besoins de MACAO, ce qui a permis d'établir le contact avec les chefs de projets afin de référencer leur expérience sur les méthodes appliquées.

THESE-MACAO	Juin 2008	32 / 266
	Nicolas FERRY	

Cette étude a visé plus particulièrement le cycle de vie de chacune des méthodes. Des interviews ont été menées auprès des chefs de projets de Capgemini dans le but de recueillir des critères de recoupement entre les différents types de projets. Cette étude est centrée autour de trois axes majeurs : le cadre, les méthodes et leur cycle de vie. Après avoir recherché les technologies, il a fallu classer les méthodes et leur domaine applicatif suivant un typage efficace. Ce travail a nécessité de prendre rendez-vous avec les interlocuteurs moteurs, de les écouter, de prendre en compte leur expérience et de recueillir leurs critères pertinents.

L'étude est focalisée sur les projets de conception objets, elle a été recueillie dans plusieurs unités de l'entreprise. Le tableau du paragraphe I.8.1 récapitule les projets et les chefs de projets interviewés.

Les différents types de projets :

- **Spatial** : projets pour le domaine du spatial. Par exemple le projet CARINS qui modélise le comportement du moteur Vulcain de la fusée Ariane 5.
- **Gestion** : projets de suivi de gestion. Par exemple : CMH est un projet qui gère l'arborescence de la structure des composants d'un avion d'Airbus.
- **Partage SI** : projet pour une application centrale de l'entreprise qui offre une plate-forme de services pour partager son système d'information en interne.
- **Maintenance** : projet qui vise à maintenir un logiciel construit par Capgemini ou un autre prestataire.
- **Migration** : projet qui vise à changer les données et/ou les services d'une application dans une nouvelle technologie.
- **Pilote SNI** : projet ayant utilisé et participé à la mise en place du SNI de MACAO.

Parmi les interviews on note les clients suivants : CNES, AIRBUS, le STNA, OCTALIS, Crédit Agricole, la SNECMA.

Les chefs de projets interviewés sont cités ci-dessous pour remerciements :

Adel SASSI, Jean-Claude LALLEMANT, Gilles BIDOT, Laurence DUQUESNE, Pascal BOISGARD, Jean-Christian DUBROCA, Karen MEXE, Thierry BEIGBEDER, Christophe GOUGEARD, Christophe ESTAQUE et Bénédicte CADEOT.

La répartition des unités pour les projets indiqués sont :

- BAYONNE : pour décrire l'unité située sous le soleil de Bayonne.
- SPACE : l'unité toulousaine pour les projets du spatial.
- ATM : l'unité de gestion pour l'aéronautique.
- ADC : l'unité pour les projets de « nouvelles technologies ».

THESE-MACAO	Juin 2008	33 / 266
	Nicolas FERRY	

I.8.1 - Tableau récapitulatif des interviews :

Type d'application / Client	Nom du projet	Interviewé	Skill	Cycle de vie	Méthodes
Spatial :					
CNES	CARINS	Adel SASSI	BAYONNE	MACAO	MACAO
CNES	Projet QI	JC LALLEMANT	SPACE	Cycle en V + Phasage	Cycle en V
Gestion :					
CNES	CATS	Gilles BIDOT	BAYONNE	Cycle en V	Cycle en V
CNES	REGATE	Laurence DUQUESNE	ATM	?	?
AIRBUS	CMH	Pascal BOISGARD	ADC	Cycle en Y	TTUP
AIRBUS	Atelier Electrique	Jean-Christian DUBROCA	ADC	Cycle en Y	Empirique
STNA	SINOPSIS	Karen MEXE	ATM	Cycle en spirale	Libre
OCTALIS	OCTALIS 2	Thierry BEIGBEDER	ADC	Cycle en Y	TTUP
Partage de SI :					
CNES	SITOOOLS	Christophe GOUGEARD	ADC	Cycle en Y	UP
Maintenance :					
STNA	ODS	Christophe ESTAQUE	ATM	Cycle en V (Medias)	Cycle en V
STNA	SAPP	Christophe ESTAQUE	ATM	Cycle en V (Medias)	Cycle en V
AIRBUS	BFE Online	Bénédicte CADEOT	ADC	?	?
Migration :					
C.Agricole	ALPHEE	Thierry BEIGBEDER	ADC	Cycle en Y	TTUP
SNECMA	Maquette RLR	Thierry BEIGBEDER	ADC	Cycle en Y	TTUP
Pilotes SNI :					
SICLI	Sicli	Gilles BENDERITTER	ADC	Cycle en V	SNI MACAO
COFATHEC	Cofathec	Pascal HAMET	ADC	UP - MACAO	SNI MACAO
INRA	SI CNRGV	Thierry BEIGBEDER	ADC	UP - MACAO	SNI MACAO
Militaire	SARSAT	Jean-Marie DAMAS	SPACE	Cycle en V	SNI MACAO
EDF	CIIGAZ	Pascal HAMET	ADC	?	PROPAL SNI

Tableau 1 : Tableau des interviews menées auprès de chefs de projets.

THESE-MACAO	Juin 2008	34 / 266
	Nicolas FERRY	

De manière générale l'élaboration de solutions informatiques a principalement recours au partenariat entre plusieurs intervenants et notamment aux chefs de projets pour mener la mission. Lors des interviews, l'écoute du récit des projets des tenants et des aboutissants, des liens avec les clients, des répartitions des équipes, de la communication entre les personnes, des technologies employées ont été un moment privilégié et très appréciable. Tout simplement parce que la réalité d'un projet est unique et l'approche prise par ces personnes sont des choix guidés par l'expérience.

Ce travail a eu plusieurs conséquences : d'une part l'ouverture et l'échange avec des personnes que je n'aurais pas forcément connues autrement, d'autre part la compréhension d'une partie de leur travail et des tâches qu'ils doivent mener chaque jour.

D'un point de vue technique, l'interview se déroulait en tête à tête avec les personnes qui présentaient leur projet et l'expérience qu'ils en avaient vécue. La conduite de l'interview s'est présentée sous forme d'un questionnaire établi par l'équipe en charge de CMMi. Les critères technologiques ont été recoupés avec les méthodologies, les cycles de vie, le type de projet,... afin de déterminer des corrélations et des bonnes pratiques par méthodologie.

A partir de là, les caractéristiques des méthodologies ont été obtenues suivant les critères de convenance avec le type de client. Les tableaux suivants présentent les résultats de cette étude du point de vue des cycles de vie des projets avec leurs avantages et leurs inconvénients. Ce travail a par ailleurs contribué à donner des grilles de caractéristiques sur les outils, les diagrammes UML, les méthodologies les mieux adaptées en fonction du type de projet.

I.8.2 - Avantages et inconvénients des méthodes :

8.2.1. Méthode : Cycle en V

Adaptabilité		
La méthode du cycle en V est la méthode traditionnelle qui a pour approche de séparer les données et les traitements. Ce type de cycle est rigide. C'est aussi le plus efficace lorsque l'on maîtrise la technologie pour le projet. Ce cycle est choisi lorsque le client utilise des technologies éprouvées et souvent pour des projets dits "critiques".		
S'adapte plutôt :	Ne s'adapte pas	
Client qui sait très bien ce qu'il veut.	Client incertain de ses propres besoins ou qui croit les connaître	
Client qui utilise une technologie connue et maîtrisée.	Projet de 'Nouvelles technologies' ou qui peuvent varier pendant le projet	
Client plutôt "Rigide"	Client plutôt "Innovant"	
Client voulant contrôler un projet dit "critique"		
Client qui fait ses analyses en amont et sous-traite la réalisation		
THESE-MACAO	Juin 2008	
	Nicolas FERRY	
		35 / 266

8.2.2. Méthode : Cycle en Y (Two Track Unified Process)

Adaptabilité	
<p>Les méthodes UP (RUP, TTUP, ..) s'adaptent bien pour les projets novateurs utilisant de nouvelles technologies basées sur les concepts objets. Cette méthode définit des dates jalons pour fixer les délais. Elle est très réactive car possède des itérations à chacune de ses étapes tant que le client n'a pas validé le passage au jalon suivant. La séparation en une branche d'analyse et branche technique permet de valider la faisabilité des fonctions pour une plate-forme technique donnée. Généralement on peut générer un prototype pour valider la technique. Tout le cycle est répété de manière itérative pour chaque livraison au client.</p>	
S'adapte plutôt :	Ne s'adapte pas
Client qui connaît assez bien ses besoins	Aux projets de petite envergure.
Client très réactif ou interactif.	Aux projets de durée très faible.
Client qui utilise des nouvelles technologies.	Aux projets dits "critique"
Client sur un projet de longue durée.	
Client désirant une architecture objet complexe.	

8.2.3. Méthode : MACAO – Spirale

Adaptabilité	
<p>Les méthodes en spirale (MACAO,...) s'adaptent bien dans les projets de recherche, ou lorsque le client ne cerne pas bien ses besoins ou que les technologies sont très novatrices. Les itérations des cycles en spirale favorisent l'interaction avec le client et le suivi de l'évolution de ses besoins. Les prototypes permettent d'étaler les risques. Ce cycle favorise très tôt la participation des utilisateurs. Les délais de livraison sont respectés grâce à la règle de non régression des prototypes. La méthode donne une documentation complète pour la maintenance.</p> <p>MACAO est une méthode novatrice dans plusieurs domaines : avec la modélisation de l'IHM, le découpage du projet en plusieurs prototypes et par ses règles de non-regression entre les prototypes. Par ailleurs, MACAO inclut un certain nombre d'avantages de RUP.</p>	
S'adapte plutôt :	Ne s'adapte pas
Client qui ne cerne pas assez bien ses besoins	Aux petits projets ou de durée très faible
Client très réactif ou interactif.	Aux projets dits "critique"
Client désirant une architecture objet complexe.	Aux projets qui n'ont pas de date précise de fin de réalisation
Client sur un projet de longue durée.	
Client sur projet à "risques".	

Globalement, se dégage le fait qu'il n'y a pas vraiment de solution «miracle». En contrepartie chacune des méthodes a ses avantages et ses inconvénients. Le cycle en V correspond plutôt à des projets ayant des structures assez figées mais efficaces pour des secteurs critiques comme celui du spatial. Le cycle en Y s'adapte mieux aux projets de nouvelles technologies qui les intègrent dans le cycle au fur et à mesure (branche en Y). Enfin, un cycle de vie en spirale comme celui de MACAO convient bien à tout type de projets dont le périmètre n'est pas complètement connu et sera amené à se modifier.

B.I.9 - Bilan

Ce chapitre a présenté le métier du génie logiciel ou l'art de réaliser des programmes informatiques. C'est le créneau des Sociétés de Services en Ingénierie Informatique. Nous avons vu la problématique de la conception de logiciels avec leur complexité exponentielle et effleuré une partie de la difficulté du métier.

Nous avons vu qu'elles étaient les approches mises en oeuvre pour réduire les difficultés liées au génie logiciel. Cette réduction tient d'une part aux outils mieux adaptés pour supporter des contraintes qui évoluent constamment, d'autre part à la modélisation qui permet de conceptualiser et de mieux comprendre le problème, et enfin à la définition d'un processus de réalisation au moyen d'une méthode.

D'un point de vue de l'amélioration du processus, un bref historique a rappelé les types de démarches et les types de méthodes existantes. Leurs cycles de vie définissent les principales étapes pour la réalisation d'un logiciel. Ensuite, une étude a montré les avantages et les inconvénients des différents cycles de vie utilisés sur des projets objets standard.

Cette étude a permis notamment de conclure qu'aucune méthode n'était « parfaite », au mieux une méthode résout certains problèmes particuliers. Cependant ce qui est important c'est de la choisir au mieux pour s'adapter à un problème donné. Il faut d'ailleurs noter que c'est le rôle principal du chef de projet de faire face et de s'adapter aux écueils rencontrés.

Les avantages de la méthode MACAO sont apparus comme applicables à certains projets industriels de Capgemini. Ainsi au cours des trois années de thèse, MACAO a été utilisée sur six projets de nature très variés principalement pour sa modélisation de la composante IHM. Ceci a permis de l'expérimenter sur des projets réels et de prendre en compte les retours des architectes.

THESE-MACAO	Juin 2008	37 / 266
	Nicolas FERRY	

Nous avons vu que MACAO est une méthode itérative. Son concepteur Jean-Bernard Crampes ne cache pas que cette méthode est le fruit d'un savant compromis de compositions des différentes méthodes existantes. Elle est apparue un peu avant RUP et toutes deux présentent de nombreuses similitudes.

THESE-MACAO	Juin 2008	38 / 266
	Nicolas FERRY	

Chapitre II : La démarche MACAO

B.II.1 - Introduction

MACAO est le fruit de plus de trente années d'expérience acquise dans des travaux de recherche, d'enseignement ou de conseil en entreprises. Ces différentes activités, réalisées dans le domaine du génie logiciel et des méthodes de conception des systèmes d'information, l'ont amenée à devoir solutionner de nombreux problèmes techniques, organisationnels et humains. En partant d'une part des techniques les plus récentes en matière de conception et de programmation, d'autre part d'une réflexion portant sur les difficultés rencontrées au cours du déroulement d'un projet informatique, MACAO jette les bases d'une méthode complète, moderne et efficiente qui doit permettre de mener un projet de bout en bout en réduisant le risque d'échec de façon significative.

Les principaux objectifs de la méthode MACAO sont :

- aider les informaticiens dans la recherche de toutes les informations nécessaires à la conception et au développement d'un logiciel,
- concevoir la structure globale et détaillée du logiciel en termes de classes d'objets,
- définir l'IHM la mieux adaptée aux besoins des utilisateurs,
- développer des programmes orientés-objets fiables, évolutifs et réalisés dans les délais prévus,
- aider à la maintenance du logiciel livré.

Pour atteindre ces objectifs, MACAO propose une démarche participative de bout en bout, depuis la définition des besoins jusqu'à la livraison du logiciel et sa maintenance.

Les principales caractéristiques de MACAO sont les suivantes :

- étude structurée de l'existant ainsi que des besoins organisationnels et environnementaux,
- développement itératif et participatif par prototypage incrémental permettant de réduire considérablement les risques,
- utilisation des modèles UML,
- modélisation conceptuelle de l'IHM en utilisant des patrons (design patterns [5][74]),
- représentation de l'IHM pour Windows et l'Internet.
- proposition de critères d'évaluation permettant d'estimer la charge de travail pour le développement,
- production d'une documentation type et d'une structure d'AGL permettant de gérer le projet et de maintenir efficacement le logiciel après sa livraison.

THESE-MACAO	Juin 2008	39 / 266
	Nicolas FERRY	

MACAO prend en compte le facteur risque tout au long du processus de fabrication du logiciel et propose des critères d'évaluation des charges de travail pour le développement du logiciel.

La démarche proposée par MACAO est proche du processus RUP (Rational Unified Process) produit par Rational Software Corp [53]. Elle s'en éloigne cependant sur trois points principaux :

- les tests utilisateurs (β -tests) sont réalisés pour chacun des prototypes produits,
- la conception des parties interactives du logiciel fait appel à une modélisation spécifique de l'IHM (SNI et MLI),
- la démarche est allégée car MACAO ne propose pas de méthode de gestion de projet poussée.

B.II.2 - La démarche participative et interactive MACAO

La démarche est dite *participative* car elle fait intervenir les futurs utilisateurs de façon régulière et intensive afin de parvenir à une solution logicielle qui les satisfasse.

MACAO adopte une démarche regroupant quatre paliers dans la réalisation d'un logiciel. Cette démarche, comparable à celle proposée par RUP, est composée de quatre étapes : l'analyse globale du système, la conception globale, le développement, et la finalisation. L'analyse globale et la conception globale traite grosso modo de 50 à 80% des besoins. L'étape de développement est une phase itérative qui produit les prototypes par incréments successifs. Cette étape est elle-même décomposée en plusieurs sous étapes pour élaborer un prototype. Nous verrons que les utilisateurs sont mis à contribution pour la définition et les bêta-tests de chaque prototype. Enfin, l'étape de finalisation correspond au dernier prototype et elle intègre la préparation de la livraison.

THESE-MACAO	Juin 2008	40 / 266
	Nicolas FERRY	

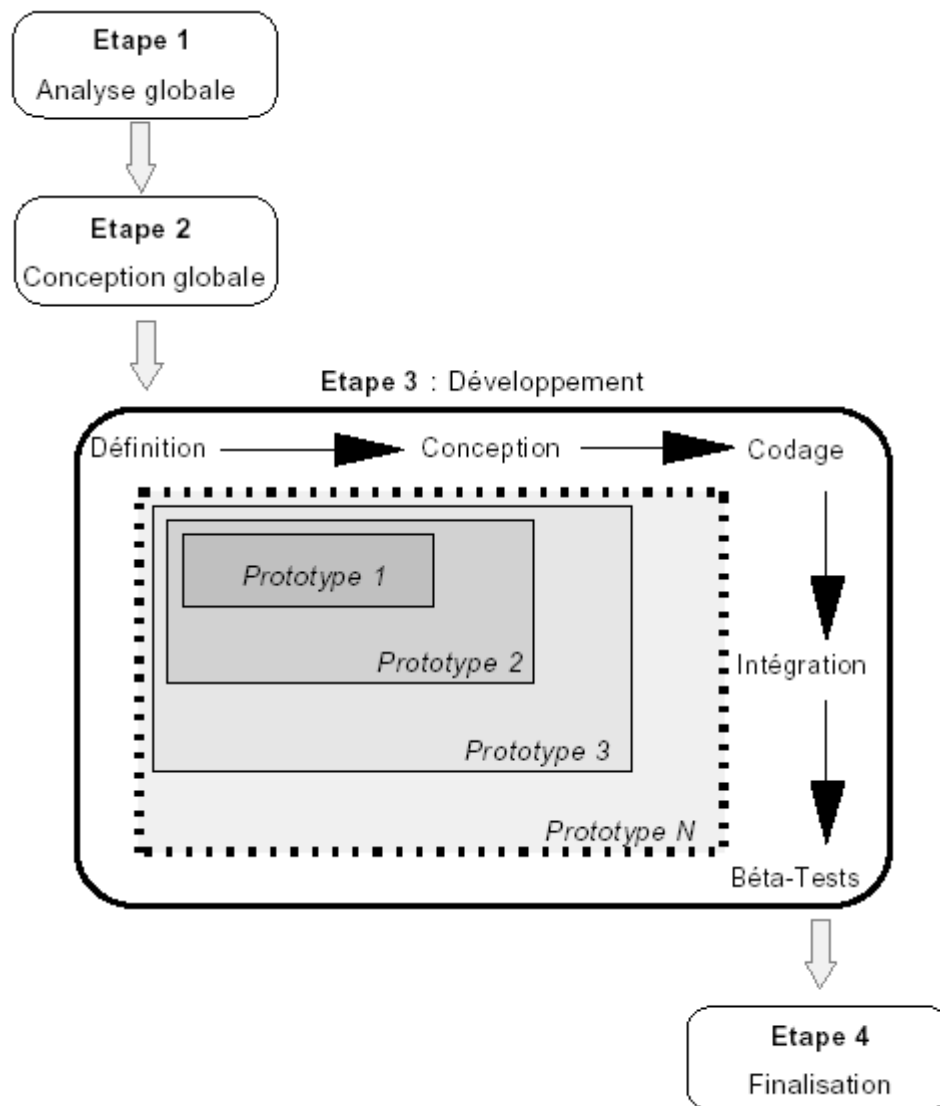


Figure 5 : Les étapes du processus MACAO

THESE-MACAO	Juin 2008	41 / 266
	Nicolas FERRY	

II.2.1 - Etape 1 : Analyse globale

L'objectif de cette étape est de prendre connaissance de l'existant et de cerner les besoins des utilisateurs concernant la totalité du projet envisagé. Remarquons que totalité ne signifie pas exhaustivité. En effet, compte tenu de l'aspect itératif du processus de développement il n'est pas nécessaire de recenser tous les besoins à ce niveau. Seuls les besoins principaux seront mis en évidence. Tous les autres apparaîtront lors de la phase d'analyse des prototypes.

Les besoins sont présentés en terme de *cas d'utilisation* (*Use Cases*). L'existant peut être décrit sous différentes formes : diagrammes des circuits et des tâches, schémas divers, langage naturel ou toute autre forme plus adaptée et facile à interpréter par l'utilisateur.

Documentation produite :

- Cahier des Charges Utilisateur (CCU)
- Dossier des Spécifications (DSP)
- Rapport d'Opportunité (ROP)

II.2.2 - Etape 2 : Conception globale

Le but est de concevoir l'architecture du logiciel sur plusieurs plans :

Fonctions à réaliser (architecture fonctionnelle)

A partir des cas d'utilisation définis lors de l'étape 1, on définit toutes les fonctions qui devront être présentes dans le logiciel et réparties en trois grandes catégories :

les fonctions d'IHM, les fonctions de traitement et les fonctions d'interface avec les logiciels externes. Les fonctions d'IHM sont déduites des cas d'utilisation définis dans le dossier de consultation. Les fonctions de traitement et d'interface sont obtenues à partir des autres besoins exprimés par les utilisateurs.

Objets et classes

Une classe est la représentation abstraite d'un ensemble d'objets de même nature. A ce niveau, seules les classes métiers (classes qui correspondent à des ensembles d'objets connus des utilisateurs) sont répertoriées. Un diagramme des classes métiers est construit pour représenter les classes elles-mêmes ainsi que les relations qui les lient (association, héritage, composition...).

Des diagrammes d'objets peuvent compléter le diagramme des classes à fin d'explications complémentaires.

IHM

L'IHM est conçue en définissant les unités de dialogue c'est-à-dire les groupes de données qui seront présentés en une seule fois à l'utilisateur. L'enchaînement des unités de dialogues

THESE-MACAO	Juin 2008	42 / 266
	Nicolas FERRY	

(navigation) est provoqué par des choix laissés à l'utilisateur. Les unités de dialogues et la navigation sont représentées par un Schéma Navigationnel d'IHM (SNI) obtenu essentiellement à partir des fonctions et du diagramme des classes.

Cette étape propose un découpage de la production du logiciel suivant plusieurs prototypes qui seront développés de façon incrémentale. Elle comprend également une méthode d'évaluation du volume de travail en mois/homme pour les étapes suivantes.

Documentation produite :

- Dossier de Conception Globale (DCG)
- Plan de développement (PDV)

II.2.3 - Etape 3 : Développement

A partir du plan de développement rédigé à l'étape précédente on propose un découpage du développement en plusieurs prototypes suivant un cycle en spirale. Chaque prototype est décrit par les fonctions qu'il devra mettre en œuvre (périmètre fonctionnel initial) et est développé suivant cinq phases, chacune se terminant par la rédaction d'un ou plusieurs dossiers de documentation.

THESE-MACAO	Juin 2008	43 / 266
	Nicolas FERRY	

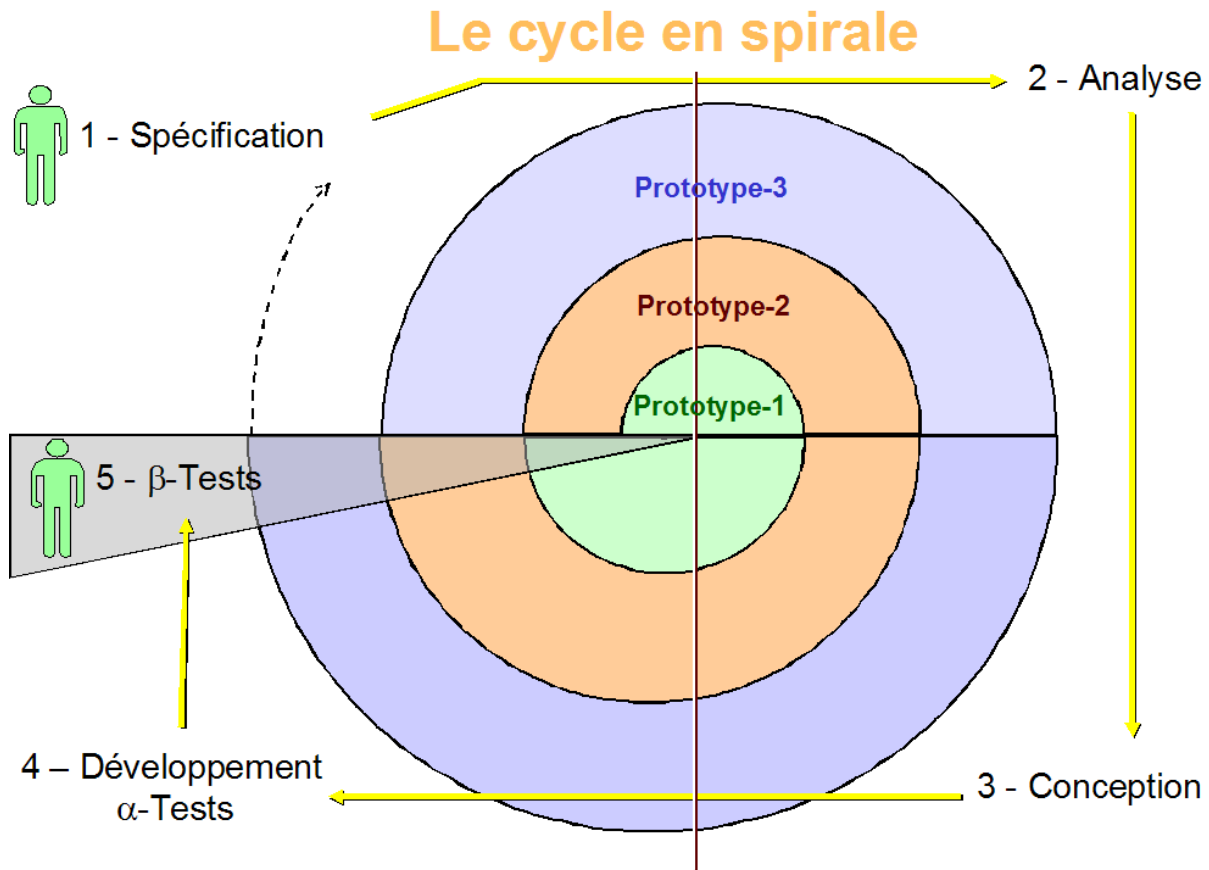


Figure 6 : Etape de développement par prototypage successif

2.3.1. Phase 1 : Définition

Le périmètre exact du prototype à réaliser (périmètre fonctionnel) doit être précisé par :

- affinage de l'expression des besoins des utilisateurs
- mise en évidence de nouvelles fonctions
- dessins ou maquettes des principaux résultats à obtenir (imprimés, écrans, autres).
- définition des droits d'accès ...

Documentation produite :

- Dossier de Définition du Prototype (DDP)
- Embryon du Dossier de Réalisation du Prototype (DRP)

THESE-MACAO	Juin 2008	44 / 266
	Nicolas FERRY	

2.3.2. Phase 2 : Conception détaillée

Le but de cette phase est :

- affiner la conception, notamment le diagramme des classes du prototype en cours en y rajoutant les classes techniques avec les méthodes et attributs supplémentaires.
- concevoir l'architecture détaillée et la dynamique détaillée en construisant les diagrammes d'états transitions, d'interaction, de séquences, etc. qui sont nécessaires.
- prévoir le découpage en composants de l'architecture et le déploiement des unités pour les applications de type client-serveur.
- réaliser les schémas d'enchaînement pour les applications interactives. Ces schémas sont obtenus à partir du schéma navigationnel d'IHM construit à l'étape 2 et réduit au prototype en cours de développement.
- étudier la structure et l'ergonomie des pages ou des boîtes de dialogue finales.
- construire les diagrammes d'enchaînement pour les chaînes batch.
- réaliser les diagrammes des classes persistantes qui représenteront un sous schéma de base de données associé au prototype.
- évaluer le volume de travail en jour/homme pour le codage et les tests du prototype en cours.

Documentation produite :

- Complément du Dossier de Réalisation du Prototype (DRP)
- Plan d'Essais du Prototype (PEP)

2.3.3. Phase 3 : Codage

Le but de cette phase est le :

- Codage du prototype à partir du diagramme des classes du prototype et en dérivant les classes du prototype précédent.
- Mise en œuvre de la base de données limitée au prototype. Tests unitaires réalisés par les développeurs.

Le codage utilise la règle de non-régression qui interdit la modification de classes définies dans un prototype précédent ceci afin d'éviter d'introduire des bogues sur des fonctions déjà validées par le prototype précédent. Bien entendu, la règle de non-régression n'est réellement applicable que si les structures de classes ont été un minimum étudiées dans ce but. (Cf. §B.II.4)

MACAO interdit toute modification sauf dans des cas particuliers où le coût de modification nécessiterait de construire des « usines à gaz » pour des modifications mineures. Dans ces cas MACAO introduit les Demandes de Modifications de Prototypes (DMP) qui sont des requêtes spéciales pour la modification de certaines classes avec l'accord du chef de projet.

THESE-MACAO	Juin 2008	45 / 266
	Nicolas FERRY	

Documentation produite :

- Complément du Dossier de Réalisation du Prototype (DRP)

2.3.4. Phase 4 : Intégration

Cette phase consiste essentiellement à réaliser les tests complets sur le prototype et à constituer la version bêta destinée aux tests utilisateurs : exécutable, composants externes, fichiers de paramètres, guide d'installation, manuel utilisateur...

Documentation produite :

- Rapport d'Essais d'Intégration du Prototype (REIP)
- Finalisation du Dossier de Réalisation du Prototype (DRP)
- Manuel Utilisateur du Prototype (MUP)
- Guide d'Installation du Prototype (GIP)

2.3.5. Phase 5 : Tests utilisateurs et recette

Tests du prototype réalisés par les utilisateurs et rédaction de fiches d'anomalies. Lorsque l'utilisateur est satisfait du prototype on procède à sa recette.

Documentation produite :

- Rapport d'Essais Utilisateur du Prototype (REUP)
- Retouches des documents précédents : DRP, MUP, GIP
- Certificat de Recette du Prototype (CRP)

II.2.4 - Etape 4 : Finalisation

Le dernier prototype pour lequel le client a signé la recette correspond au logiciel terminé. On procède alors à la constitution de la documentation finale : dossier de réalisation qui sera utilisé pour la maintenance du logiciel, du manuel utilisateur et du guide d'installation.

Documentation produite :

- Dossier de Réalisation (DR)
- Guide d'Installation (GI)
- Manuel Utilisateur (MU)

THESE-MACAO	Juin 2008	46 / 266
	Nicolas FERRY	

B.II.3 - Récapitulatifs :

Documentation produite

Analyse globale	Conception globale	Développement					Finalisation
		Définition	Conception	Codage	Intégration	β Tests	
CCU Cahier des Charges Utilisateurs DSP Dossier des Spécifications	DCG Dossier de Conception Globale	DDP Dossier de Définition Prototype DRP Dossier de Réalisation Prototype	PEP Plan d'Essais Prototype	REDP Rapport d'Essais de Dével. Prototype DMP Demande de Modif. Prototype	REIP Rapport d'Essais d'Intégration Prototype MUP Manuel Utilisateur Prototype GIP Guide Installation Prototype	FAP Anomalies FMP Modif. MUP GIP DRP	MU Manuel Utilisateur GI Guide Installation DR Dossier de Réalisation
	PDV Plan de Développement		DRP	DRP	DRP	DRP	
	PQL Plan Qualité Logiciel						

Tableau 2 : Documentation à produire à chaque étape du processus

Applications des diagrammes par étapes

Modèles	Analyse globale	Conception globale	Développement				Finalisation
			Définition	Conception	Codage	Intégration	
Use Cases	C	U	M	U			
DCT	C	M	U				
Activité	C	M	U				
Classes		C		M	U	U	M
Objets		C		M			
Composants				C	U	M	U
Déploiement				C		C	U
Séquence	C		C, M	C, M	U		
Collaboration	C		C, M	C, M	U		
Etats		C		C, M	U		
SET				C (B)			U
SNI		C (I)		U			U
MLI				C (I)	U		U

Tableau 3 : Chronologie d'utilisation des diagrammes UML + MACAO dans la démarche générale

Légende :

C : le diagramme est créé dans cette étape.

M : le diagramme est modifié au cours de cette étape.

U : le diagramme va être utilisé à cette étape.

I : S'applique aux traitements interactifs

B : S'applique aux traitements Batch (Traitements non interactifs)

En gras sont indiqués les modèles les plus couramment utilisés.

THESE-MACAO	Juin 2008	48 / 266
	Nicolas FERRY	

B.II.4 - Perspectives d'évolutions :

MACAO peut être améliorée de plusieurs façons pour éviter les ambiguïtés, et pour faciliter l'application de la règle de non-régression.

Avec le retour d'expérience du terrain, les DMP (Demande de modifications prototypes) sont parfois confondues avec les FMP (Fiches de modifications de prototypes). La première est émise par l'équipe de développement pour le chef de projet, alors que la seconde est une demande d'évolution du logiciel émise par le client. Pour éviter cette confusion, les DMP seront appelées DPNR pour Dérogation au principe de Non-Régression.

Par ailleurs, la règle de non-régression des prototypes de MACAO stipule que les sources d'un prototype N-1 (où N est le prototype en cours de réalisation) ne doivent plus être modifiées par les développeurs. Pour une application Objets ce procédé est mis en oeuvre en utilisant l'héritage. Une classe métier qui doit être modifiée dans un prototype N doit hériter de sa mère dans le prototype N-1. Cette règle doit être scrupuleusement suivie par les développeurs afin d'éviter l'introduction de bogues par « effet de bord » sur le code existant.

Cependant, il arrive que dans certains cas, un héritage simple ne suffise pas à gérer tous les cas d'évolution de la conception. Si une architecture est correctement prévue, les classes des prototypes précédents doivent être peu ou pas modifiées. Dès lors, il est important que la conception fournisse une architecture relativement équilibrée. L'utilisation de patrons dans les phases amonts peuvent sensiblement améliorer la conception [133].

Bertrand Meyer énonce des principes pour la conception orientée objet :

- Gestion des évolutions et des dépendances entre classes.
- Organisation des applications en modules
- Gestion de la stabilité de l'application

Il faut donc que la conception soit robuste dans son architecture pour justement être évolutive sans pour autant être modifiable. Citons quelques principes comme l'ouverture / fermeture des modules (OCP) et l'inversion des dépendances (DIP).

L'ouverture/fermeture des modules (OCP) consiste à laisser un module libre et ouvert aux extensions mais fermé aux modifications. Au coeur de ce principe se trouve la définition d'une classe interface qui fournit les services. L'inversion de dépendance (DIP) consiste à ne plus hériter directement d'une classe métier mais plutôt d'une classe interface déportant ces méthodes.

THESE-MACAO	Juin 2008	49 / 266
	Nicolas FERRY	

L'utilisation de ces patrons de conception (design pattern) dès les phases de conception, permettrait de limiter les restructurations impactant l'architecture de base et donc les DPNR sur ces classes métiers.

Enfin, dans sa prochaine version la méthode MACAO supportera les diagrammes définis par UML2 [6][7][8].

B.II.5 - Les modèles de conception

MACAO reprend largement dans la phase de conception la notation UML [51]. MACAO laisse le choix aux concepteurs d'utiliser avec pertinence un diagramme plutôt qu'un autre.

MACAO dispose de quatre types de modèles et de quinze diagrammes. Nous trouvons :

- **des modèles organisationnels** comportant le diagramme des cas d'utilisation, le diagramme des circuits et des tâches (DCT), Business Process Modeling Notation (BPMN).
- **des modèles structurels** comprenant le diagramme des classes, le diagramme d'objets, le diagramme structure composite, le diagramme des composants et le diagramme de déploiement.
- **des modèles dynamiques** avec les diagrammes de collaboration, le diagramme de séquences, le diagramme d'états transitions, le diagramme d'activités et le diagramme d'enchaînement.
- **des modèles d'IHM** comprenant le schéma navigationnel d'IHM (SNI), les modèles de définition technique des IHM (DTI), et les écrans de maquettes.

Dans ce qui suit, nous n'allons pas entrer dans le détail de chaque modèle ce qui n'est pas le but de cette thèse. La plupart de ces diagrammes sont issus d'UML. En revanche, nous allons mettre l'accent sur les diagrammes spécifiques qui ont trait à la conception des interfaces homme machine.

B.II.6 - Modélisation du processus

Actuellement le livre de MACAO fait état de propositions pour la métamodélisation d'un AGL. Nous avons cherché à détailler plus amplement le métaprocessus de MACAO, en partant d'une part de ce qui a été décrit dans le livre et d'autre part par complément de lecture et vision de la pratique de cas concrets. La tâche a nécessité de récupérer une masse de connaissances importantes. Celle-ci provient en partie des interviews à Capgemini et de lectures basées sur RUP.

THESE-MACAO	Juin 2008	50 / 266
	Nicolas FERRY	

Pour décrire les cas d'utilisation de la méthode MACAO, la démarche a été de retrouver les tâches des intervenants à partir des interviews avec les équipes. MACAO définit plusieurs rôles dans une équipe comme chef de projet, architecte, développeurs, etc. Le travail d'une personne dans un rôle donné a permis de référencer ses tâches. Celles-ci ont été notées et répertoriées sous forme de cas d'utilisation. Suivant le niveau de détail désiré, les tâches peuvent être décomposées en sous tâches plus précises. Cependant pour ne pas fixer un processus trop rigide, l'étude n'est pas descendue à plus de trois ou quatre niveaux.

Les documents ou artefacts pour UP, les rôles, les tâches des informaticiens, leurs vues importantes, etc. ont été répertoriés. Ceci a permis de constituer un glossaire et de faire une classification de premier ordre. Ainsi pour chaque membre d'une équipe MACAO sont référencés : la liste de ses buts principaux et la liste de ses tâches pour les atteindre.

Le résultat de cette étude a été de produire des fiches récapitulatives des buts, des tâches et des vues sur le système pour chaque rôle d'une équipe MACAO. Ces fiches de références ne doivent pas être vues comme des motifs impératifs de contraintes qualité pour les équipes mais plutôt comme un guide de conseils et un support pour la création d'un atelier de génie logiciel incorporant les concepts de MACAO.

B.II.7 - Bilan

Comme nous l'avons vu dans ce chapitre, MACAO est une méthode participative et interactive originale qui apporte des concepts novateurs notamment dans le fait de placer l'utilisateur final au centre de la solution, par son analyse détaillée de l'aspect IHM, par sa réalisation de prototypes incrémentaux ainsi que par sa règle de non-régression des prototypes. Ses critères principaux sont actuellement évalués sur plusieurs projets issus de domaines très variés. A cause de son évolution historique en provenance du « terrain », les utilisateurs de MACAO réalisent les diagrammes à la main, alors que de nos jours les documentations UML sont couramment produites via des générateurs.

Tant que la méthode MACAO ne possédera pas de plate-forme convenable et d'outils performants pour l'utiliser, son utilisation ne sera pas pratique pour des projets industriels. Pour faciliter le travail du concepteur à une époque où l'informatique peut lui épargner beaucoup de tâches répétitives et fastidieuses, il est nécessaire de fournir des outils capables de répondre à ses attentes.

L'un des travaux de cette thèse consiste donc à apporter le support informatique aux modèles d'interfaces homme-machine de Macao.

THESE-MACAO	Juin 2008	51 / 266
	Nicolas FERRY	

Chapitre III : La modélisation des IHM

B.III.1 - Représentation des IHM

A l'heure actuelle, il existe un certain nombre d'approches pour modéliser les IHM. Les démarches RAD prennent le problème en modélisant par le bas directement les maquettes avec les utilisateurs. Par conséquent, elles n'emploient pas de modèles puisqu'elles s'efforcent de saisir la représentation finale souhaitée par l'utilisateur. Elles n'ont donc pas de représentation conceptuelle pour étudier la navigation globale.

Parmi les solutions existantes, nous remarquons que beaucoup reprennent certaines capacités des diagrammes UML pour faire de la conception d'IHM. Certains diagrammes stéréotypés basés sur ceux d'UML sont construits pour répondre à ce besoin. D'autres solutions définissent des représentations spécifiques. C'est l'approche qui a été faite par Paulo da Silva avec UMLi [62][88]. Ou bien encore diaModl qui offre une représentation IHM-objet.

Actuellement, divers modèles sont utilisés pour la modélisation des IHM. On trouve les diagrammes d'activités [129], d'états transitions, d'interaction, les réseaux de Pétri [131]. Tous ces modèles sont souvent très généraux et proposent des représentations souvent complexes et difficilement utilisables pour un dialogue avec un utilisateur. Trop généraux dans le sens où de manière générale, ils peuvent être appliqués à plusieurs domaines différents autre que l'étude des IHM. Ainsi, le diagramme d'activité modélise l'enchaînement d'activités. Par abus de langage une activité peut être transformée en écran ou page. MACAO ne considère pas que les commandes et saisies de l'utilisateur sont des « activités » comme les autres mais préfère parler de « dialogue » avec l'utilisateur. Il est vrai que dans UML les diagrammes d'états transitions et d'activités sont les candidats les mieux placés pour décrire une navigation générale et qu'ils peuvent convenir pour une description succincte. Certains auteurs utilisent ces diagrammes comme Pascal Roques dans son livre [86]. Mais de l'avis de l'auteur lui-même : ces diagrammes n'ont pas été prévus pour cet usage et il faudrait les stéréotyper pour qu'ils puissent correspondre au domaine tout en leur rajoutant de la sémantique. Il y a donc bien un besoin exprimé pour la modélisation des IHM.

Certains travaux ont été menés avec les réseaux de Pétri, pour la représentation de la navigation globale : les réseaux de Pétri sont plutôt avantageux pour réduire la nature combinatoire à un graphe fini. Par ailleurs, des outils peuvent générer des machines à états pour l'exécution de l'IHM. Cependant, nous remarquerons que pour construire une IHM, les écrans peuvent être trouvés plus simplement par un dialogue avec l'utilisateur lui-même qui n'a en outre pas forcément les compétences d'appréhender toutes les subtilités des réseaux de

THESE-MACAO	Juin 2008	52 / 266
	Nicolas FERRY	

Pétri. Nous ne remettons pas en cause qu'au final et d'un point de vue fonctionnel, un réseau de Pétri peut très bien générer l'automate d'une IHM d'une application. Encore une fois, c'est dans le processus de recueil des besoins que cela nous semble pour la majorité des projets, difficile à mettre en oeuvre. Nous voyons donc que les réseaux de Pétri, par leur caractère souple, peuvent être employés dans des domaines très variés et qui n'ont rien à voir avec la modélisation d'une IHM. Il nous semble plus judicieux d'utiliser une sémantique dédiée et par conséquent plus efficace pour décrire les problèmes d'IHM.

Une tentative d'adaptation d'UML à la modélisation des IHM a été faite par Paulo da Silva avec UMLi [12]. UMLi définit une sémantique de représentation des IHM, en s'intéressant à l'aspect statique d'une interface, la partie dynamique étant laissée à la charge de diagrammes d'activités et d'états transitions. La démarche adoptée par UMLi est intéressante, cependant MACAO va plus loin dans le sens où sa description de l'IHM traite aussi l'aspect dynamique avec l'enchaînement des unités de dialogues, les droits d'accès, les conditions de navigation, les tris et filtres sur les affichages de collections et cela au niveau conceptuel avec un seul diagramme, là où UMLi en utilise trois.

DiaMODl est un autre langage de modélisation d'IHM. Il permet de documenter et spécifier des boîtes de dialogues graphiques. Ce langage vise à décrire une IHM de façon transversale, en permettant de décrire des contrôles d'IHM et des interactions entre des objets métiers. Il est ainsi à l'intersection du domaine de l'IHM et des classes métiers. Ceci implique qu'il n'est pas entièrement consacré à l'IHM. MACAO fait, quant à lui, grande part à la modélisation et au mécanisme de l'ingénierie dirigée par les modèles avec la transformation de modèles. Notamment le passage d'un modèle à un autre nous permet de séparer des mécanismes représentés par plusieurs modèles. Pour nous le SNI se positionne bien sur la modélisation du logiciel du point de vue de l'axe IHM. Le modèle métier étant déjà bien défini par les diagrammes UML.

Nous pouvons citer aussi le modèle de tâches CTT de Paterno [171][172] qui est hérité de LOTOS [178]. Son pouvoir sémantique avec ses opérateurs de concurrences des tâches permet d'exprimer une sémantique temporelle formelle. Des travaux ont été réalisés pour exprimer à partir d'un modèle CTT, un programme en langage formel B [130][173][174]. Dans la méthode MACAO, des modèles organisationnels sont prévus tels que le DCT (Diagramme des Circuits et des Tâches) qui est un modèle proche de BPMN. Il présente l'organisation du travail des utilisateurs, acteurs d'un processus, en terme de tâches, documents entrants et sortants, circuits scénarios...

Avec MACAO, l'approche méthodologique préconisée est de développer au préalable un modèle organisationnel de l'entreprise pour permettre notamment d'extraire les tâches

THESE-MACAO	Juin 2008	53 / 266
	Nicolas FERRY	

interactives (IHM) mais aussi les fonctionnalités (traitements) et les fichiers et documents informatisés (données métiers). L'idée développée en partenariat avec **Françoise Adreit** [175] qui est spécialiste des modèles tâches est qu'une tâche (complexe) du DCT peut être transformée en tâches élémentaires parmi lesquelles on peut mettre en évidence les tâches interactives qui vont déboucher sur les unités de dialogues (UD) du SNI. L'étude approfondie de l'informatisation d'un DCT ne faisant pas partie du périmètre de cette thèse, nous n'en décrivons ici que les principales différences :

- Le DCT utilise un chronographe fixe pour décrire l'ordonnancement des tâches; CTT utilise des opérateurs temporels formels hérités de LOTOS.
- Le DCT permet de décrire des scénarios utilisateurs par des circuits de tâches. Pour CTT le lien avec les scénarios est établi avec l'outil CTTE.
- Le DCT permet de spécifier des documents échangés entre tâches (et des fichiers informatisés pour les tâches informatisées). Il s'apparente à une description de workflow, et permet de spécifier globalement l'information échangée entre tâches.

L'outil TERESA permet de générer des transformations de modèle CTT vers des plates-formes physiques. Notre approche est semblable et il doit être possible d'étudier, grâce à l'ingénierie des modèles, des passerelles pour générer des SNI à partir de CTT, et horizontalement des liens DCT – CTT.

Il existe aussi des plates-formes avec leur API toutes prêtes, telle que Struts [79] qui utilise le patron de conception MVC, avec une plate-forme bas niveau pour laquelle un framework fonctionnel est disponible. Là aussi un diagramme d'activités est utilisé pour l'enchaînement.

Nous le notons, le besoin de modéliser les IHM s'avère de plus en plus nécessaire. D'une part parce que les interfaces des logiciels se complexifient et d'autre part parce que la quantité d'écrans est proportionnelle à la taille des projets. Nous pouvons affirmer que c'est bien un besoin et que UML n'y répond que partiellement.

C'est dans ce contexte que MACAO introduit les premiers modèles d'IHM dédiés pour la conception d'applications informatiques. Ces modèles conviennent à la modélisation des IHM d'une application. Le périmètre de ces modèles ne couvre pas les objets métiers ni la persistance des données qui sont gérées par des diagrammes UML ou Merise.

En quoi consiste un modèle d'IHM ?

Je me souviens, avant cette thèse en entreprise, avoir eu le besoin de voir l'ensemble des fenêtres et des liens de navigations de mon application. Ces fenêtres avaient déjà été réalisées à partir d'un IDE évolué. A l'époque instinctivement, j'avais réalisé des captures de chaque écran puis posé l'ensemble sur une page et relié les boutons des écrans sur la fenêtre

THESE-MACAO	Juin 2008	54 / 266
	Nicolas FERRY	

destination pour montrer l'enchaînement. Inconsciemment, je venais de réaliser mon premier schéma d'enchaînement des fenêtres dans une forme plus simplifiée que le diagramme SEF actuel de la méthode MACAO.

Cette approche est ascendante et nécessite de créer au préalable des maquettes d'écrans et de les proposer au client. Ceci convient lorsque les données sont bien connues et les fonctions bien séparées. Ensuite, pour obtenir l'enchaînement global, il suffit de « remonter » au niveau abstrait supérieur et représenter toutes les fenêtres qui s'enchaînent. En principe, il y a peu d'intérêt immédiat à « remonter » à un niveau plus abstrait. Le schéma convient à expliquer la navigation globale qui peut même être intégrée à une documentation pour l'utilisateur. Remonter d'un niveau pour atteindre le conceptuel est intéressant pour faire de la rétroconception [48] ou pour re-générer vers une application technologiquement différente.

Prenons maintenant l'hypothèse où nous devons réaliser une application depuis le début en partenariat avec le client. Dans cette étape très floue, certains clients n'ont pas réfléchi eux-mêmes à l'IHM car cela ne relève pas de leur métier tout simplement. Nous sommes alors amenés à construire avec eux l'interface pour comprendre leurs besoins. Par exemple, un client parlera facilement d'un « écran » qui « doit » faire la « gestion de ses dossiers » sans plus de détails. Ainsi il relie ses fonctions métiers à, grosso modo, ce qu'il veut obtenir. Ce travail se fait actuellement chez certains clients sur tableau blanc ou avec des post-it. La dynamique de groupe est alors très importante pour la créativité de l'ensemble.

Cette phase qui relève typiquement de l'analyse des besoins et des exigences, est plutôt descendante et se prête bien à l'utilisation de modèles abstraits. Plusieurs retours sur les projets de l'utilisation du SNI ont montré que les concepteurs appréciaient d'utiliser le Schéma Navigationnel des IHM en phase d'analyse et de conception. L'intérêt ici est double, d'une part le SNI permet de trouver les fonctionnalités majeures et de construire rapidement l'enchaînement global, d'autre part, il sépare le découpage fonctionnel de l'application et prévoit plus tard le raffinement de certaines zones floues lors de l'analyse détaillée d'un prototype.

Le déroulement conseillé avec l'approche descendante est de construire le Schéma d'Enchaînement (niveau logique). Ce passage se fait en choisissant une technologie informatique d'implémentation ou une plate-forme de génération (pages GUI, pages Web, mais aussi un diagramme sur une technologie particulière telle que Struts, etc.). Comme nous le verrons plus tard, il s'agit concrètement d'un typage des éléments abstraits. Il faut remarquer que le passage à une technologie donnée peut être associé à un choix de plate-forme physique (PC, PDA, etc.) [158].

THESE-MACAO	Juin 2008	55 / 266
	Nicolas FERRY	

Nous allons étudier ces modèles, du niveau le plus abstrait au niveau le plus concret, en commençant par le Schéma Navigationnel des IHM ou SNI. Historiquement, ces diagrammes étaient fait à la main c'est pour cela qu'ils ont été dénommés schémas.

B.III.2 - Le Schéma de Navigation des IHM (SNI)

III.2.1 - Définition

Le SNI permet de concevoir et de représenter l'enchaînement du dialogue entre le logiciel et l'utilisateur en tenant compte du comportement supposé de ce dernier. L'utilisateur pourra *naviguer* entre les informations qui lui sont proposées de façon apparemment libre, mais en étant cependant contraint à une logique imposée par le concepteur de l'application.

Le SNI est un modèle conceptuel à haut niveau d'abstraction permettant de représenter les exigences fonctionnelles. De ce fait, il se veut complètement indépendant de la plate-forme physique et en particulier du type d'IHM envisagé pour la réalisation (Windows, Web, Mobile, Multimodal [142] ou autres). De même, il ne représente ni les moyens d'interaction (menu déroulant, bouton poussoir, glisser déposer...), ni les aspects matériels mis en œuvre (clavier, type d'écran, souris, activateurs [160],...) qui sont du ressort du niveau réalisation. Par ailleurs, il ne prend pas en compte l'exécution des traitements (calcul, accès aux données...).

Nous pouvons remarquer aussi que le SNI représente les fonctions proposées à l'utilisateur par l'application ainsi que les résultats qui lui sont soumis sans qu'interviennent ni les méthodes d'obtention des résultats ni l'ergonomie de leur présentation.

III.2.2 - Objectifs

Les principaux objectifs du SNI sont notamment de modéliser l'IHM en terme de navigation entre les différents éléments de base, les droits d'accès et la couverture fonctionnelle liée à l'IHM.

Il propose un langage adapté à la communication entre les réalisateurs et les utilisateurs pour donner le feedback de l'architecture générale du point de vue de l'IHM. Pour cela, nous allons le voir, le SNI adopte une sémantique épurée dans ses éléments de base.

Enfin, il va permettre de définir l'IHM d'un point de vue abstrait et va permettre par raffinements une génération quasi-automatique du code conformément à l'approche MDE [35] en utilisant les transformations de modèles.

III.2.3 - Planches d'un SNI

Un SNI peut être composé de une ou plusieurs planches. Chaque planche représente une surface éditée assimilée à la notion de page dans un livre.

THESE-MACAO	Juin 2008	57 / 266
	Nicolas FERRY	

III.2.4 - Les Unités de Dialogue (UD)

L'unité de dialogue (UD) est l'élément de base de construction du SNI. Une unité de dialogue est une représentation abstraite de l'interface. L'interface est à la frontière entre l'utilisateur et le logiciel. Les interactions avec l'utilisateur se font, soit sous forme de commandes vers le logiciel, soit par retour d'informations depuis le logiciel. Elles contribuent à l'échange interactif avec l'utilisateur c'est pour cette raison que le nom « d'unités de dialogues » est employé. Nous distinguons deux types d'UD possibles : les UD dites « élémentaires » et les UD « composées ».

III.2.5 - Les UD élémentaires (UDE)

Une unité de dialogue est dite élémentaire si elle correspond à une opération d'IHM indécomposable. Le tableau ci-dessous décrit une classification des UDE en six catégories telles qu'elles apparaissent dans MACAO. L'expérience acquise dans de nombreux projets concrets a montré que ces six catégories étaient nécessaires et suffisantes pour modéliser conceptuellement tout type d'IHM quelle que soit sa nature et sa complexité. Paulo da Silva dans UMLi est d'ailleurs arrivé à la même conclusion en proposant une catégorisation semblable.

On trouve six types d'UD élémentaires :

➤ Saisie de données



Il s'agit ici d'une fonction de saisie pure d'information pour créer un nouvel objet, pour entrer la valeur d'un paramètre ou pour saisir un événement. Nous verrons plus loin comment modifier des données déjà saisies.

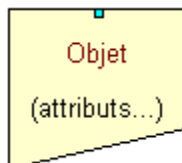
➤ Affichage d'un résultat ou d'un objet



L'affichage est réalisé sur un support informatique (généralement un écran mais peut être vocal ou sous forme de fichiers) et présente soit la totalité des données concernant un résultat simple tel qu'un document électronique, un tableau statistique, une image, un histogramme, une courbe,... soit les attributs relatifs à un objet. Par exemple pour une personne on affiche ses attributs nom et prénom.

THESE-MACAO	Juin 2008	58 / 266
	Nicolas FERRY	

➤ Impression d'un résultat



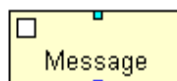
Ce type d'UD est équivalent au précédent mais le résultat est persistant pour l'utilisateur. En règle générale, il modélise une impression sur support papier de documents liés aux données manipulées.

➤ Affichage d'une collection d'objets



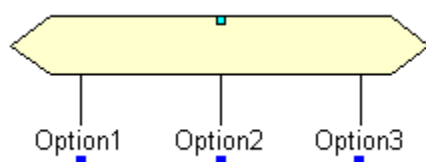
C'est un ensemble d'objets qui est concerné par ce type d'affichage. Dans la plupart des cas, ceci représente une collection d'objets informatisés avec l'affichage de certains de leurs attributs. L'affichage se présente sous forme d'un tableau, d'une liste (ou parfois une collection finie d'objets tels que l'affichage de villes sur une carte) comportant en pratique, généralement, une ligne par objet et une colonne par attribut.

➤ Affichage d'un message



Le message est affiché à l'initiative du logiciel et non à celle de l'utilisateur. Il peut s'agir d'un message d'information, d'avertissement, d'erreur, de confirmation, d'alerte...

➤ Les menus



THESE-MACAO	Juin 2008	59 / 266
	Nicolas FERRY	

Ce type d'unité de dialogue ne correspond pas à une fonction mais au choix donné à l'utilisateur pour sélectionner une fonction parmi un ensemble possible. Notons bien la différence entre les choix provenant des utilisateurs et les choix automatiques du système. Les UD de menu représentent en fait la navigation de l'utilisateur dans l'IHM.

Ces unités de dialogue de base suffisent à modéliser tous les types d'interface informatique. Cette représentation est complète. En effet, de façon générale, elle aurait même pu être limitée à deux grandes catégories : soit une information en direction du système, soit une information issue du système. L'ajout d'éléments supplémentaires augmente la richesse de la sémantique.

En contrepartie il contribue à la spécialisation du diagramme pour des cas de plus en plus particuliers. Les six éléments proposés sont un enrichissement minimal et satisfaisant pour l'adapter au domaine de l'IHM et humainement facilement exploitable. Un cerveau humain, d'après Georges Miller, retient facilement jusqu'à sept éléments mémorisables en une seule fois [psychological review p.81-94 – The magical number seven]. Dès lors l'apprentissage de la sémantique du SNI est à la portée de pratiquement tout le monde. Ceci le place en bonne position s'il faut exposer la navigation globale du projet au client.

III.2.6 - Les UD composées (UDC)

Le SNI offre un mécanisme de composition intéressant qui permet de créer des unités de dialogues dites composées (comme les mots sont composés de lettres). Plusieurs UD élémentaires peuvent être regroupées pour constituer une UD composée. Une unité composée permet d'une part de regrouper plusieurs informations de nature proche pour optimiser le travail de l'utilisateur, d'autre part de paralléliser plusieurs interactions conformément à la façon de faire de l'utilisateur dans l'exécution de ses tâches.

La composition de plusieurs UD peut être réalisée simplement par juxtaposition de deux unités élémentaires (fig. 7) ou bien en utilisant un regroupement avec une « boîte de groupage » (fig. 8). Ces deux représentations sont sémantiquement équivalentes, l'utilisation de l'une ou de l'autre dépend essentiellement d'une facilité de représentation mentale ou de comportement opérationnel pour le créateur.

Bien évidemment, nous soulignons le fait que ce mécanisme est un moyen pour nous d'implémenter le mécanisme de gestion de patron conceptuel à ce niveau. En effet, les unités de dialogues composées ne sont ni plus ni moins que le concept de composants réutilisables au niveau conceptuel. Nous verrons plus tard l'aide apportée par l'outil pour créer ces patrons.

THESE-MACAO	Juin 2008	60 / 266
	Nicolas FERRY	

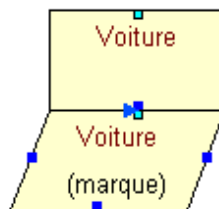


Figure 7 : Exemple de composition par juxtaposition. La figure montre une juxtaposition d'un affichage et d'une saisie pour former une UDC de modification.

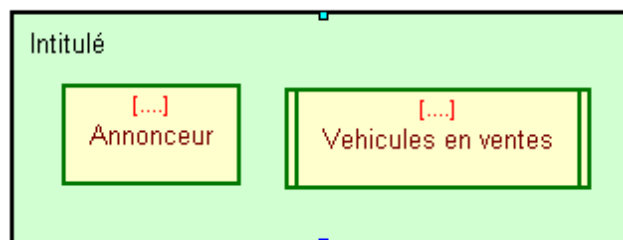


Figure 8 : Exemple de composition d'une UDC par groupage. Ici on affiche simultanément un annonceur et la liste des voitures mises en vente.

III.2.7 - Les éléments de routage

Pour relier les unités de dialogues entre elles, le SNI utilise un graphe orienté qui définit l'ensemble des enchaînements entre les unités de dialogues. Les enchaînements représentent les possibles dans l'espace de navigation. Le SNI regroupe sous le terme « routage » les éléments unitaires permettant de modéliser la navigation dans l'IHM. Nous trouvons parmi ces éléments :

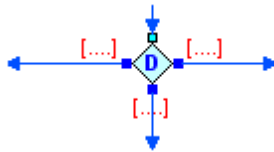
➤ Les liens



Les liens connectent les unités et les éléments de routages. Ils définissent la navigation globale dans l'IHM. Dans le SNI, les liens sont considérés comme des objets à part entière. Nous verrons le fonctionnement du système de connexions avec l'éditeur graphique.

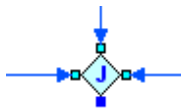
➤ La décision

THESE-MACAO	Juin 2008	61 / 266
	Nicolas FERRY	



Les décisions sont des branchements de navigation conditionnés. Elles permettent de gérer des enchaînements d'IHM dépendant de critères automatisés en opposition aux menus qui sont des choix issus de l'utilisateur.

➤ La jonction



Les jonctions sont comme les décisions des branchements de navigation. Pour créer une arrivée multiple sur une unité de dialogue, il sera nécessaire d'utiliser une jonction la précédant.

➤ Les étiquettes de début



Le SNI introduit aussi la notion « d'étiquette ». Ce terme regroupe la navigation intra et inter diagrammes. Les étiquettes de début sont des points d'arrivées potentiels.

➤ Les étiquettes d'invocation ou d'appel



Les invocations permettent de construire un SNI complexe tout en gardant une bonne lisibilité.

Ces étiquettes d'invocation ou d'appel permettent de « sauter » sur une étiquette de début qu'elle soit sur la même planche du diagramme ou non. L'intérêt est de pouvoir découper un SNI complexe en plusieurs sous SNI. La navigation est ainsi déportée géographiquement mais pas fonctionnellement. L'utilisation d'une étiquette d'invocation permet d'exploiter deux comportements majeurs : d'une part le découpage en sous diagrammes permet de scinder la

THESE-MACAO	Juin 2008	62 / 266
	Nicolas FERRY	

complexité, d'autre part l'invocation avec retour permet d'ajouter le concept de factorisation pour mettre en commun des parties d'IHM répétitives.

➤ Les retours



Les retours permettent de revenir au niveau du SNI qui a préalablement fait une invocation. Par anticipation, pour une machine à états générant le comportement dynamique de l'IHM, cela se traduira par l'implémentation d'une pile d'appel mais cela n'est pas de notre ressort puisque c'est le niveau logique puis physique qui implémentera cette capacité dans une technologie spécifique.

III.2.8 - Les commentaires

Commentaire lié à
application développée

Enfin, dernier groupe de composants du SNI et non le moins important : les commentaires permettent de spécifier plus informellement le système. Par exemple, le périmètre du SNI s'arrête à l'IHM. Souvent une action d'IHM se traduit par une fonction qui aura un impact sur un traitement, le commentaire peut être un moyen de traduire les éléments frontaliers avec les autres domaines principalement des traitements logiques ou des accès aux données.

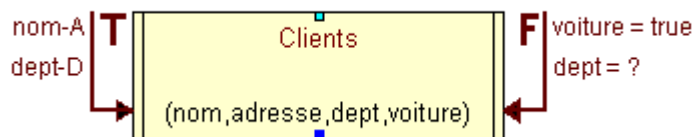
III.2.9 - Compléments de modélisation

Un certain nombre de concepts supplémentaires ont été définis dans le modèle SNI pour d'une part permettre la prise en compte d'exigences particulières des utilisateurs du logiciel concernant l'IHM, d'autre part pour intégrer une certaine flexibilité dans la structuration des diagrammes.

III.2.10 - Les compléments liés aux exigences des utilisateurs :

L'utilisation de fonctions "FILTRE" et "TRI" lors de l'affichage d'une collection d'objets recueille les souhaits des clients sur la présentation. Notons que les filtres et les tris sont bien du niveau conceptuel.

THESE-MACAO	Juin 2008	63 / 266
	Nicolas FERRY	

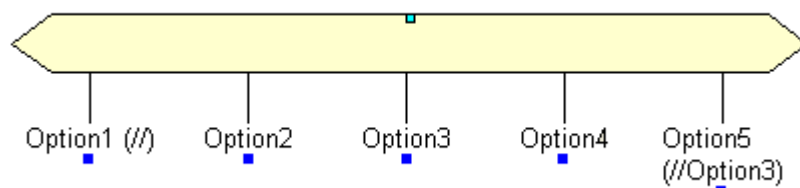


Le filtre « voiture = true » affiche uniquement les clients qui possèdent une voiture. Le filtre « dept = ? » affiche les clients situés dans le département indiqué par l'utilisateur.

Dans une application multi-utilisateurs la notion de "rôle" permet d'identifier un utilisateur dans le système. Des restrictions d'accès peuvent être mentionnées en indiquant les autorisations ou les interdictions d'accès au parcours de certaines branches de navigation. De manière plus générale, le SNI gère la notion de conditions ou "garde" permettant de parcourir certaines branches en fonction de l'état de certains objets. Nous verrons en détail par la suite les choix qui ont été faits pour implémenter les conditions.

Parallélisme de commandes :

Il est possible de déclencher des options en parallèle pour les menus.



L'option1 est déclenchée en parallèle à toutes les autres options. Et l'option5 est déclenchée en parallèle à l'option3.

III.2.11 - Exemple de SNI Concessionnaire Voitures

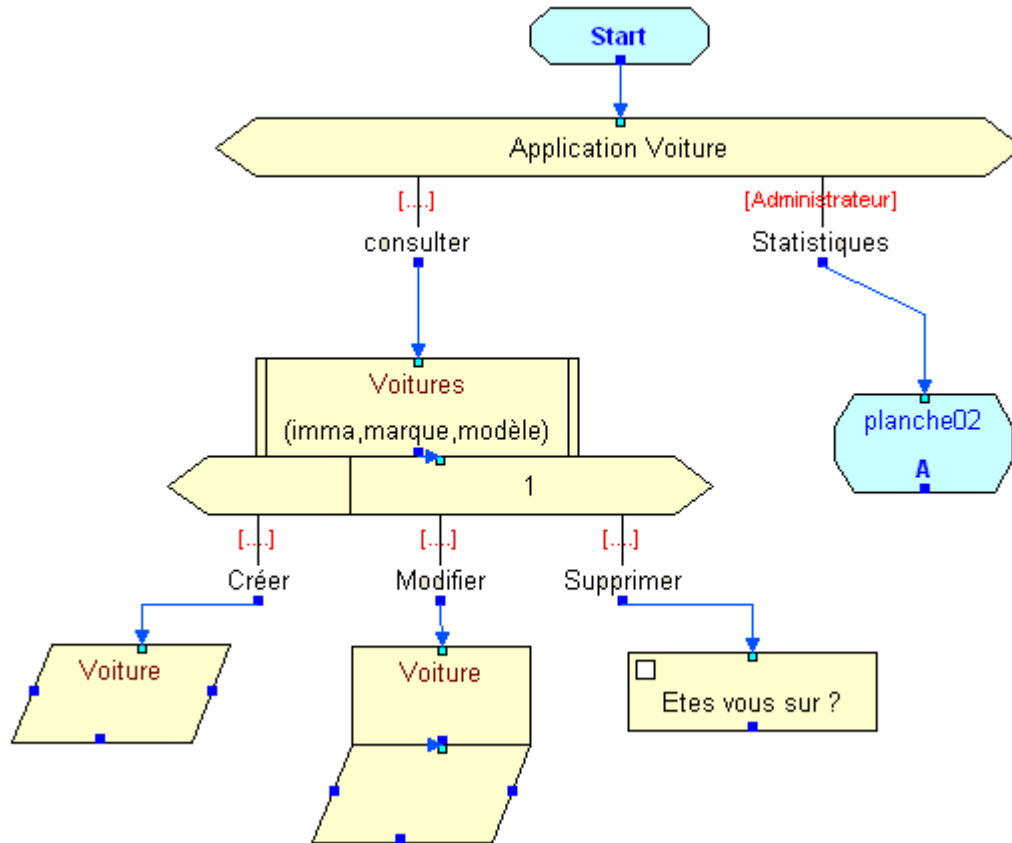


Figure 9 : Exemple d'une IHM d'un concessionnaire de véhicule. Le concessionnaire peut accéder à la liste des voitures disponibles. Il peut rentrer une nouvelle voiture, voir le détail d'un véhicule, ou en retirer une de la concession. Seul l'administrateur pourra consulter un écran de statistiques. Eventuellement l'application pourra être complétée en rajoutant une seconde planche SNI afin de définir la partie d'IHM concernant les statistiques.

Par analogie, le SNI peut se voir comme un langage composé d'un alphabet de lettres (les UDE), de mots (les UDC) et de phrases (les planches de SNI). Les lettres sont reliées par concaténation (par juxtaposition ou boîte de groupage) pour constituer des mots, les mots sont reliés par la ponctuation (les liens, jonctions et décisions) pour constituer les phrases (planches).

THESE-MACAO	Juin 2008	65 / 266
	Nicolas FERRY	

III.2.12 - Liens avec les modèles logiques

Nous allons maintenant considérer les principaux domaines d'IHM courants en examinant leurs possibilités. En effet, dès que nous définissons une plate-forme ou un domaine technologique, nous faisons implicitement des choix d'implémentation qui ont un impact sur la structure. Ainsi, c'est à ce niveau que nous allons faire des choix sur la plate-forme cible (affichage sur un écran, taille, résolution, etc.)

Suivant la manière de présenter les informations sur l'écran et en fonction des moyens utilisés pour communiquer avec l'utilisateur, nous pouvons distinguer les types d'IHM suivants :

- les IHM de type caractères ou texte : TUI pour *Text User Interface*,
- les IHM orientées fenêtre telles que dans Windows ou MAC-OS : GUI pour *Graphic User Interface*.
- les IHM orientées pages Web : PUI pour *Page User Interface*,
- les IHM orientées documents comme dans Lotus-Notes : DUI pour *Document User Interface*,
- les IHM multi-modales avec entrées et sorties vocales, écran tactile,...
- et de manière globale toutes les technologies d'IHM.

Cependant le niveau logique peut englober aussi des langages liés à un domaine technologique ou à une plate-forme. Nous pouvons citer par exemple les « clients riches » qui possèdent une répartition n-tiers par nature plus équilibrée. Ou une plate-forme répartie comme INDIGO [144]. Rien ne nous empêche de choisir une plate-forme technique et de construire un schéma de l'IHM propre à cette technologie. Par exemple une plate-forme comme Struts utilise un diagramme UML pour définir l'enchaînement des pages.

Dans ce dernier cas un mappage du SNI vers un diagramme de définition pour la plate-forme revient à pouvoir utiliser les capacités de la plate-forme pour générer directement le code.

Dans ce qui suit, nous étudierons dans le détail uniquement un des modèles logiques fourni par MACAO pour définir les écrans graphiques d'une application en mode fenêtré de type GUI.

B.III.3 - Les Schémas d'Enchaînement (SE)

Alors que le modèle de haut niveau (SNI) manipule des *unités de dialogue* (UD), les modèles de niveau logique mettent en œuvre des *composants* d'IHM (ou Unités Logiques de Dialogue) qui représentent la mise en œuvre des UD sur une plate-forme technique spécifique (Domain Specific Language) : fenêtre pour les IHM de type Windows, page pour les IHM de type WEB, message vocal pour les IHM multimodales... Remarquons que sur ces dernières des travaux de recherches établissent la vérification formelle fondée sur la preuve [139].

THESE-MACAO	Juin 2008	66 / 266
	Nicolas FERRY	

Le niveau logique se compose en réalité de deux types de modèles :

- le SE (Schéma d'Enchaînement) qui modélise la dynamique d'ouverture des composants en réponse aux événements : action utilisateur, condition remplie, détection d'une erreur, interruption système...
- le DE (Définition des Eléments) qui décrit le contenu statique c'est-à-dire l'aspect des composants en terme de *widgets* (menus, contrôles, textes, icônes, messages vocaux, entrées vocales...) et leur positionnement, de couleurs de fond, etc.

Pour notre exemple, nous nous intéresserons à une IHM de type Windows (GUI) pour le Schéma d'enchaînement. Dans ce contexte, les composants d'IHM sont les fenêtres et la modélisation de leur enchaînement est réalisée grâce au modèle appelé SEF (Schéma d'Enchaînement des Fenêtres).

III.3.1 - Typologie des objets graphiques GUI

Classiquement, les IHM GUI sont composées de deux grandes catégories d'objets : les conteneurs qui représentent les composants (menu ou fenêtre), et les contrôles qui sont les objets élémentaires d'interaction avec l'utilisateur.

Pour le SEF, nous distinguerons quatre types de fenêtres :

- FP : Fenêtre Primaire (ou Principale) de l'application,
- FS : Fenêtre Secondaire,
- BD : Boîte de Dialogue,
- BM : Boîte de Message,

ainsi que cinq types de menus :

- BA : Barre d'actions
- BO : Barre d'Outils
- MD : Menu déroulant
- MC : Menu Contextuel
- GO : Groupe d'Onglets

Les contrôles se répartissent en diverses classes telles que :

S - Statiques :

- **STC** pour Statique Texte Constant
- **STV** pour Statique Texte Variable...

B - Boutons :

- **BP** pour Bouton Poussoir

THESE-MACAO	Juin 2008	67 / 266
	Nicolas FERRY	

- **BC** pour Bouton Case à cocher
- **BR** pour Bouton Radio

E - Entrées :

- **ES** pour Entrée Simple
- **EM** pour Entrée Multi-lignes

Listes et Collections :

- **LS** pour Liste déroulante Simple
- **LD** pour Liste déroulante Développable
- **LCS** pour Liste Combinée (Combo) Simple
- **LCD** pour Liste Combinée Développable
- **TAB** pour Table
- **ARB** pour Arbre...

Options :

- **OM** pour Option de Menu
- **OO** pour Option Onglet
- **OI** pour Option Icône...

Cette liste, bien que non exhaustive car d'autres types de contrôles personnalisés peuvent être définis en fonction des exigences des utilisateurs, sera la base pour l'élaboration du métamodèle du SEF.

Le SEF permet de représenter la dynamique d'ouverture des fenêtres de l'application. Il n'utilise qu'un seul type de symbole graphique pour représenter les conteneurs : un rectangle divisé verticalement en deux parties. La partie gauche contient les caractéristiques générales du conteneur : type (FP, FS, BD, BM, MD, MC), nom, droits d'accès, paramètres d'ouverture... La partie droite est divisée en autant de lignes que de contrôles ou d'options situés dans le conteneur.

Si le nombre de contrôles est trop important on n'indiquera que ceux qui participent à un événement utilisateur impliquant l'ouverture d'un nouveau conteneur. Un dessin de fenêtre (DEF) est alors réalisé pour présenter la totalité d'entre eux ainsi que leur disposition géographique dans la fenêtre.

Les événements sont représentés par des liens ayant pour origine les contrôles ou options générateurs d'événements et pour destination les conteneurs à ouvrir. Les types d'événements sont indiqués sur les liens (l'événement par défaut est le clic gauche souris). Des paramètres peuvent également être transmis à la fenêtre destinatrice. Ils sont indiqués sur le lien, entre parenthèses.

THESE-MACAO	Juin 2008	68 / 266
	Nicolas FERRY	

BD-Consulter Vehicules DESSIN	LS-Voitures (Imma,Marque,Modele)
	BP-Créer
	BP-Modifier
	BP-Supprimer

Figure 10 : Représentation d'une fenêtre dans le SEF. Il s'agit ici de la boîte de dialogue (BD) présentant le détail d'un véhicule. Le dessin (DESSIN) explicite le contenu de la boîte de dialogue.

III.3.2 - Exemple de SEF

Reprenons, à titre d'exemple le SNI de l'administration de la classe "Véhicules" vue précédemment (cf. figure 9). L'administration s'effectue en trois boîtes de dialogue.

Les UD de création et de modification d'un véhicule ont été regroupées en une seule boîte de dialogue : *BD-CreModifVehicule*. Les boîtes de dialogue *BDDétailVehicule* et *BD-CreModifVehicule* sont incomplètes c'est pourquoi elles seront détaillées par un dessin non présenté ici. La boîte de dialogue *CreModifVehicule* comporte un paramètre optionnel (entre crochets) selon qu'elle est ouverte à la suite d'une création (aucun paramètre) ou d'une modification (l'objet "véhicule" doit être présent).

L'impression est représentée par le même symbole que dans le SNI car il ne met pas en jeu de fenêtre si ce n'est une boîte de dialogue commune non envisagée ici, de choix de l'imprimante et des options d'impression.

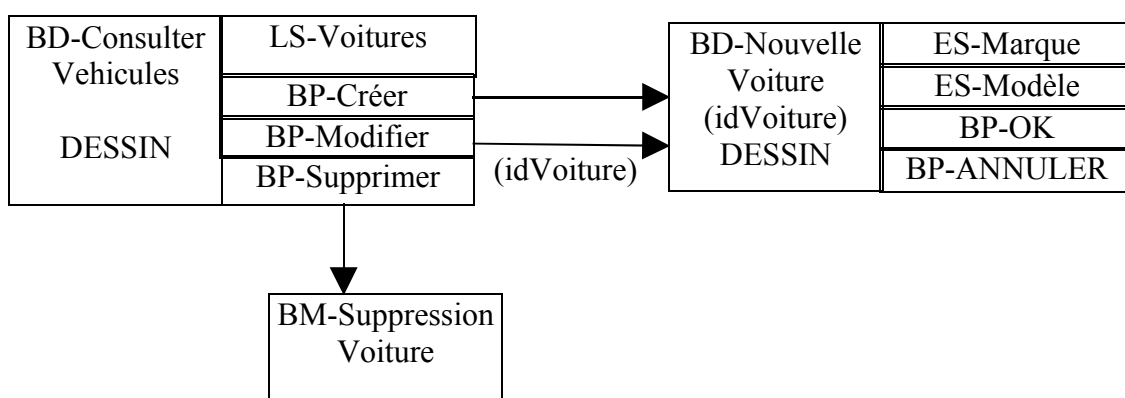


Figure 11 : SEF de l'administration de la classe "Véhicule". Remarquons la présence du paramètre "IdVoiture" qui permet de réaliser la liaison entre les fenêtres.

B.III.4 - Les maquettes d'écrans



Le dessin des boîtes de dialogue complexes est réalisé en utilisant des symboles de représentation des principaux types d'objets visuels comme indiqués ci-dessous. Ces dessins correspondent aux maquettes d'écrans présentées au client.

Les symboles des contrôles sont positionnés à l'intérieur du contour de boîte. Les contrôles statiques constants (textes et cadres) sont indiqués en clair. Les contrôles actifs sont numérotés séquentiellement de gauche à droite et de haut en bas. Le numéro de contrôle est reporté dans une légende pour lui faire correspondre son nom préfixé par son type.

Une maquette peut utiliser une représentation abstraite pour rester indépendant du type de composants, ou de boîtes à outils de composants enrichis comme [138], ou directement montrer le résultat à l'utilisateur final en situation.

Les deux figures ci-dessous montrent deux exemples de dessins de boîtes de dialogue : une boîte de présentation et une autre de saisie.

THESE-MACAO	Juin 2008	70 / 266
	Nicolas FERRY	

Création d'un étudiant	
Code : <input type="text" value="1"/>	Promotion : <input type="text" value="2"/> 
Nom : <input type="text" value="3"/>	
Prénom : <input type="text" value="4"/>	
5 <input checked="" type="radio"/> Masculin 6 <input type="radio"/> Féminin	Date de naissance <input type="text" value="7"/> <input type="text" value="8"/> <input type="text" value="9"/>
Adresse Rue 1 : <input type="text" value="10"/> Rue 2 : <input type="text" value="11"/> Code postal : <input type="text" value="12"/> Ville : <input type="text" value="13"/>	
Série de bac : <input type="text" value="14"/> 	
15 <input type="button" value="Valider"/>	16 <input type="button" value="Annuler"/>
17 <input type="button" value="Aide"/>	

1 STV-CodeEtu
2 LD-Promotion
3 ES-Nom
4 ES-Prenom
5 BR-Masculin
6 BR-Feminin
7 ES-JourNais
8 ES-MoisNais
9 ES-AnNais
10 ES-Adr1
11 ES-Adr2
12 ES-CodePostal
13 ES-Ville
14 LD-SerieBac
15 BP-Valider
16 BP-Annuler
17 BP-Aide

Figure 12 : Maquette d'écran d'une saisie d'étudiant avec sa légende

Contrats


2

Page : 1

Recherches et sélections [Aide sur les recherches](#)


3 Tous les contrats


4 Contrats impayés

Type de contrat : 5  OK 6

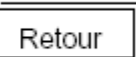
Début du nom : 7 OK 8

Numéro de contrat : 9 OK 10

Assuré	Numéro de contrat	Type	Date de paiement	Paielement
11 <input type="text"/>	12 <input type="text"/>	13 <input type="text"/>	14 <input type="text"/>	15 
⋮	⋮	⋮	⋮	⋮

16 Haut de page 

Pages : 17 . . .

18 

Légende	
1 : V-NumPage	10 : HI-OkNumContrat
2 : HC-Aide	11 : HV-NomAssuré
3 : HC-Tous	12 : V-NumContrat
4 : HC-Impayés	13 : V-TypeContrat
5 : LD-TypeContrat	14 : V-DatePaielement
6 : HI-OkType	15 : HI-Paielement
7 : ES-Nom	16 : HC-HautPage
8 : HI-OkNom	17 : HV-Pages
9 : ES-NumContrat	18 : BA-Retour
100 contrats par page	

Figure 13 : Maquette d'écran d'une page Web pour gérer des contrats

THESE-MACAO	Juin 2008	72 / 266
	Nicolas FERRY	

B.III.5 - Bilan

Dans ce chapitre, nous avons vu une manière de modéliser les interfaces homme-machine. La méthode MACAO introduit en plus des modèles définis par la standardisation UML, un ensemble de modèles spécifiques à la modélisation des IHM. Ces modèles sont adaptés pour décrire différents niveaux d'une même problématique d'IHM. Ils couvrent la plupart des applications informatiques classiques mais ils ne couvrent pas vraiment le domaine des jeux vidéo, de la réalité virtuelle ou de la réalité augmentée [134].

Le SNI a pour tâche de modéliser globalement une IHM. Nous y lisons d'un coup d'oeil la navigation générale des éléments visuels d'une application, les droits d'accès pour les utilisateurs et enfin le moyen d'atteindre les fonctionnalités de l'application.

Ce diagramme montre beaucoup d'intérêt dans les phases de capture des exigences et de conception de l'interface globale. Sa structure favorise la discussion et l'échange avec un client sur les questionnements classiques mentionnés ci-dessus [80].

Les schémas d'enchaînements quant à eux interviennent lors de la phase de conception où les architectes construisent l'architecture de l'IHM. Plus pragmatiques, ils montrent l'enchaînement global et récapitulent par écrans les éléments logiques contenus. Tous les objets sont typés en vue de leur implémentation sur une plate-forme. Les diagrammes sont déductibles automatiquement des modèles SNI. Des règles de transformations ont été créées pour remplir cette tâche.

Enfin, les dessins de maquettes correspondent à la définition visuelle de l'interface. Le graphiste donne l'aspect des écrans en précisant les attributs visuels des objets graphiques. Les maquettes commencent à montrer un retour pour le client de l'aspect que prendra l'IHM.

D'ores et déjà dans le travail réalisé au cours cette thèse, nous remarquons que ces diagrammes véhiculent les mêmes concepts, au travers de différentes étapes, pour la création d'une IHM. Le passage d'un modèle de haut niveau à celui d'un niveau inférieur évolue par raffinement. Ainsi de l'information est elle ajoutée à chaque étage au fur et à mesure. A contrario, le passage d'un modèle de niveau inférieur vers un niveau supérieur constitue une abstraction qui masque de l'information.

Nous appelons « axe IHM » la dimension qui va définir l'IHM d'une application. Lors de la conception, l'axe IHM va intervenir pour décrire le logiciel d'un point de vue de son IHM. Les modèles d'IHM vont définir l'IHM par étapes successives de façon complémentaire aux modélisations actuelles principalement fédérées par le standard UML.

THESE-MACAO	Juin 2008	73 / 266
	Nicolas FERRY	

Nous allons définir plus en détail l'axe IHM et son processus. Nous montrerons comment celui-ci peut être mis en oeuvre et coller aux pratiques d'une société de services en Informatique. Les différents modèles définissent l'IHM sur cet axe, les modules que nous cherchons à développer viendront se positionner comme moyen de réalisation.

Mais avant de présenter ces outils intervenant dans le cadre d'un atelier de génie logiciel, nous devons étudier « la cible » c'est-à-dire le but à atteindre et dans quel contexte MACAO va devoir évoluer. Cette étude stratégique vise à déterminer une plate-forme stable pour l'évolution et le développement de l'atelier de génie logiciel MACAO.

THESE-MACAO	Juin 2008	74 / 266
	Nicolas FERRY	

Partie C - Espace solution

Chapitre I : Etat de l'art

C.I.1 - Etude de la cible et du contexte technologique

Dans le cadre de l'état de l'art, nous allons énumérer les principaux intervenants et les forces en présence dans la conception de logiciels informatiques. Nous verrons qu'il émerge plusieurs grandes tendances. Nous examinerons, au travers d'articles ou interviews sur Internet, les objectifs et les orientations stratégiques des principaux intervenants du moment, puis, nous verrons quels sont les intervenants tiers. Nous listerons les différents produits disponibles sur le marché qui viennent appuyer ou compléter la modélisation UML sur ces plates-formes. Le but est de cerner la place la plus propice à l'épanouissement de la démarche MACAO et la conception d'interface homme machine.

L'évolution rapide du Web au travers du réseau Internet engendre une multiplication des besoins de services informatiques délocalisés. Ainsi, une multitude de possibilités s'ouvrent devant nous en terme de séparation et déploiement de notre connaissance et de notre savoir-faire. Ceci devrait entraîner un désengorgement des clients lourds au profit des clients dits légers.

La technique en terme de présentation d'IHM sur le Web évolue, on remarque que depuis la page HTML texte, ont été rajoutés du graphique et de l'animation puis on a cherché à générer ces pages dynamiquement. Aujourd'hui, la technique est mûre pour passer à la prochaine étape, c'est-à-dire la volonté de fournir des services au travers d'une présentation aussi riche fonctionnellement que les applications des clients lourds. Cette technologie est appelée communément : le client riche (RCA : Rich Client Application ou RIA : Rich Internet Application).

Le concept de client riche [75] reprend celui du client/serveur avec pour principe la séparation entre la présentation et les données mais avec des traitements répartis. Le client riche permet une présentation complexe réaliste avec des traitements locaux. C'est un compromis entre le client lourd et le client léger. L'interface cliente devrait progresser à terme vers une présentation dynamique fortement interactive supportant les capacités de calcul proposées par les cartes graphiques 3D.

Dans un processus classique de standardisation, plusieurs techniques tentent de s'affirmer tant en matière de modélisation que de technologies informatiques. Généralement, plusieurs techniques d'un même domaine apparaissent et chaque éditeur défend alors les atouts de sa technique. Un organisme de normalisation tiers fédère ensuite les standards qui

THESE-MACAO	Juin 2008	78 / 266
	Nicolas FERRY	

ont émergé. L'organisme est principalement un regroupement d'experts ou de sociétés spécialisées dans ce domaine faisant office de référent pour la validation. On peut citer par exemple le W3C, l'OMG, etc.

Ainsi voit-on ce processus s'appliquer à différents niveaux (machines, systèmes d'exploitation, plates-formes d'exécution,...). Suite à la standardisation d'UML, le 'marché' des méthodologies va s'ouvrir sur de nouveaux modèles. Le phénomène des diagrammes DSL (Domain Specific Language) [28] en est d'ailleurs une marque. La tendance globale actuelle étant de revenir vers des modèles plus 'spécifiques' pour favoriser l'efficacité. La recherche axe aussi ses travaux afin d'améliorer les DSL [157]. Sans supprimer les avantages des modèles UML, MACAO rajoute des modèles plus spécifiques à la conception des interfaces homme-machine.

Capgemini a exprimé des besoins en terme de modélisation des IHM. La méthode MACAO a donc été testée sur un projet réel nommé SITOOLS. Ce projet possède une certaine complexité en terme d'écrans. Cela convient bien à la démarche MACAO.

Nous le verrons plus tard dans l'analyse des outils existants, il faut déterminer la plate-forme technique sur laquelle implémenter les futurs outils MACAO. D'un point de vue théorique MACAO n'est pas limitée à une plate-forme particulière en soi. Cependant il est nécessaire de choisir une plate-forme technique qui soit judicieuse pour que la mise en œuvre de l'outil soit la plus efficace et pérenne possible. Ce choix doit correspondre aussi aux attentes de l'entreprise.

Pour arriver à déterminer la plate-forme, le raisonnement est le suivant. Il faut partir des besoins énoncés par Capgemini et référencer les plates-formes utilisées, tout en réalisant le meilleur compromis entre stratégie à long terme, réponse des besoins, facilité de développement, positionnement de MACAO et anticipation de l'évolution des technologies. Le choix s'est orienté vers une plate-forme open-source résolument ouverte vers l'extérieur et qui prône l'échange : Eclipse.

Il a fallu aussi déterminer le contexte général d'évolution par rapport aux autres plates-formes. C'est pour cela qu'une classification a été établie pour répertorier les techniques et les différents domaines. Cette recherche est fonction des éléments trouvés sur Internet et par connaissances internes et/ou externes à Capgemini. L'étude des domaines existants fournie en annexe, a permis notamment de déterminer la plate-forme supportant l'outil de génie logiciel appliquant la démarche MACAO.

THESE-MACAO	Juin 2008	79 / 266
	Nicolas FERRY	

C.I.2 - Les besoins de Capgemini

Le but principal pour Capgemini est de préparer une industrialisation des étapes dans la création de logiciels informatiques avec une grande réactivité aux nouvelles technologies.

La mise en place de l'industrialisation s'oriente en plusieurs chantiers :

- Etude d'un outil qui répond aux critères (besoins exprimés ci-dessous)
- Développement d'un support formation
 A l'heure actuelle, il manque une procédure / formation décrivant l'utilisation de technologies comme : UML, MAVEN [77], CVS, JUnit, ...
 Ces formations peuvent avoir un impact sur plusieurs unités du département.

Les besoins exprimés par Capgemini :

1. Le prix des licences Rational Rose se montre élevé. (Capgemini utilise ce logiciel couramment pour ses développements objets) La recherche de logiciels substitutifs tels que les AGL ou logiciels 'open source' de modélisation / conception UML est un axe d'étude en cours. Rational Rose reste encore utilisé dans la société avec des outils de génération de codes performants qui ont été conçus en interne.
2. La plate-forme 'open source' Eclipse montre un fort intérêt de part sa gratuité et sa capacité modulaire à recevoir différents plugins. L'échange des données est réalisé par la norme XMI (XML Metadata Interchange) qui est normalisée par l'OMG et permet l'échange de données avec Rational Rose notamment. Un portage des anciens modèles semble donc possible.
3. Un outil qui supporte la modélisation d'interface graphique (GUI), la norme MDA et une génération de code sont nécessaires.
4. Il manque un générateur de code vers l'environnement .Net et notamment C#.
5. L'outil devrait supporter le Round-trip engineering [82].
6. La génération de documentation automatique est nécessaire.
7. L'outil doit permettre la formalisation du problème (phase de Requierment)
8. Le support de la traçabilité est nécessaire.
9. Le support de l'intégration continue est nécessaire.
10. Voir les générateurs de tests automatiques « open-source » est souhaitable.
11. Assurer des tests de non-régression.
12. Capgemini souhaite lisser la phase de transition entre les concepteurs et les développeurs.

THESE-MACAO	Juin 2008	80 / 266
	Nicolas FERRY	

C.I.3 - Définition de la thématique globale :

La vision de la plate-forme complète s'organise dans une thématique globale à plus longue échéance que la thèse elle-même.

A terme, il faut fournir un atelier de génie logiciel avec un environnement complet qui permettra de participer à l'industrialisation des sociétés de services en informatique. Dans ce but, une plate-forme MACAO supportant le travail coopératif des différents intervenants doit permettre la mise en oeuvre de la méthode sur différents projets informatiques. Cet environnement reprend des points similaires avec celui présenté par Gilles Halin et Sylvain Kubicki [149].

Dans la méthode MACAO un cycle de vie est défini. Les intervenants sur le projet peuvent prendre des rôles différents comme : chef de projet, analyste, architecte, développeurs, testeurs, qualitatifs, ergonomes,... Chaque rôle doit effectuer des tâches pour produire le logiciel. La méthode MACAO permet de décrire les rôles, les buts, les tâches et les documents entrants et sortants pour chacun. Ainsi les intervenants réalisent les tâches qui leur incombent au travers de leur point de vue ou perspective sur l'atelier de génie logiciel MACAO. Dans une approche moderne, le travail de chacun suit généralement une ou plusieurs représentations propres au métier de l'intervenant qui passent dans la plupart des cas par des modèles de la solution qui ont un point de vue ou une vision du système. Cette étude du processus de la méthode se trouve en détail dans l'annexe de composition de l'AGL MACAO.

La plate-forme gère dans un « repository » l'évolution de la réalisation des projets. La méthode inclut aussi la définition des IHM contrairement à la plupart des autres méthodes, ce qui permettra de modéliser via les diagrammes d'IHM de MACAO la partie interface. Pour la conception de l'IHM, les concepteurs pourront utiliser :

- le diagramme conceptuel du SNI
- les diagrammes logiques SE
- le diagramme intermédiaire des maquettes d'écrans
- le générateur physique pour créer le squelette de code de l'interface

Plusieurs modules de génération pourront être construits en fonction des technologies cibles choisies : par exemple, un module pour générer du Laszlo [76], du WinForm, du Struts, du JSF, etc. La création des modules de génération est laissée à l'entreprise car elle doit maîtriser son savoir-faire.

THESE-MACAO	Juin 2008	81 / 266
	Nicolas FERRY	

Le même raisonnement est applicable sur la conception des objets métiers et les bases de données. De nombreux travaux et outils industriels du commerce proposent déjà des générateurs évolués pour cela. Ces outils peuvent être inclus dans l'atelier MACAO.

La méthode MACAO ne dépendant pas d'une plate-forme technique donnée, il est important de rester indépendant autant que possible des outils pour pouvoir évoluer. Ainsi il est tout à fait envisageable de combiner MACAO avec des logiciels ou modules existants pour gérer le cycle de développement.

C.I.4 - L'insertion de la démarche MACAO

Il semble que Microsoft évoluera vers ses propres modèles de conception délaissant les standards fixés par l'OMG (UML, MDA [52],...). Les outils tout intégrés comme BEA offrent une plate-forme performante de développement mais sont propriétaires et l'insertion d'un plugin pour un éditeur spécifique est un risque générant de la dépendance et un doute en terme de pérennité.

Après étude auprès de plusieurs personnes au sein du groupe ADC (Advanced Development Center) c'est-à-dire le pôle des nouvelles technologies de Capgemini, nous pouvons tirer plusieurs constatations :

Eclipse est une plate-forme « open source » qui présente l'avantage du prix puisqu'elle est gratuite. Ce critère tranche avec les licences généralement appliquées pour l'utilisation de produits. Cette plate-forme montre aussi beaucoup de souplesse grâce à son architecture basée sur un noyau simple de gestion de plugins. IBM a fourni en standard des modules qui peuvent servir de briques réutilisables à la manière d'une API système. Ainsi, des fournisseurs tiers apportent leur contribution et enrichissent les fonctionnalités pour réaliser une plate-forme spécifique à leurs besoins.

Eclipse évolue très vite et profite du support massif d'une communauté grandissante de développeurs. Le suivi implicite d'IBM pour sa plate-forme lui garantit une certaine stabilité face au géant Microsoft.

Eclipse peut être utilisé comme un éditeur, une IDE de développement, et même comme client riche applicatif. En effet, il est possible d'insérer des modules qui construiront une application à part entière. Et c'est là sa force, Eclipse fournit par ailleurs un système de perspectives qui sont composées de différentes vues. Ce système permettra de définir une perspective par rôle de la méthode MACAO. L'analyste aura sa perspective, le chef de projet la sienne, etc. Ensuite l'intervenant placera ses vues du système comme bon lui semble à

THESE-MACAO	Juin 2008	82 / 266
	Nicolas FERRY	

l'intérieur. Une vue pouvant être des tableaux, des diagrammes, du source, etc. Les utilisateurs pourront le configurer au mieux de leurs besoins.

Actuellement, le groupe ADC couvre plusieurs domaines et utilise plusieurs environnements de développement :

- les projets menés avec Eclipse
- les projets IntelliJ + Iplanet avec le projet CMH
- Les projets .NET avec le Portail ou CTOOL ou les PDA
- Les projets avec BEA weblogic
- etc.

Capgemini a décidé d'unifier ces plates-formes dans la mesure du possible et s'oriente vers Eclipse. Eclipse permet de rester indépendant des techniques, ce qui contribue à une certaine souplesse pour s'adapter à la demande du marché et des clients.

Tout cela nous a encouragés à choisir Eclipse comme plate-forme pour le développement de l'atelier de génie logiciel de MACAO.

Dans les propositions présentées à Capgemini, Eclipse a été retenu comme solution pour unifier la plate-forme de développement en discernant trois catégories d'utilisation :

- Une plate-forme **High-cost** (c'est-à-dire pour des projets ayant de forts budgets) basée sur Eclipse utilisant IBM Rational **Atlantic**.
Cette solution couvrira tout ou une partie du cycle de développement en fonction du coût des licences. C'est aussi un choix d'efficacité pour une société qui veut se donner les moyens d'avoir des outils complets et performants en matière de développement de logiciel. Ce choix est corrélé avec l'existant et le passif de certain produit comme Winchill qui nécessite une conception avec du Rational Rose.
Une transition progressive de l'ancien Rational Rose vers Atlantic doit être envisagée.
- Une plate-forme **Low-cost** (pour des projets à faible budget) basée sur Eclipse utilisant des outils **open source**.
Cette solution privilégie des outils open source de type MAVEN, ANT, CVS, UML2... qui permettront d'approcher en terme de fonctionnalités une plate-forme payante pour un prix dérisoire. Il faudra alors définir une plate-forme commune pour les équipes.
- Un outil de modélisation UML spécialisé pour le conceptuel, permettra de faire une approche MDA [33] et un modèle indépendant de la plate-forme.

THESE-MACAO	Juin 2008	83 / 266
	Nicolas FERRY	

Il existe de bons logiciels comme : **Enterprise Architect, Together** ou **Poseidon**. A noter particulièrement Enterprise Architect qui offre un excellent rapport qualité/prix et représente une bonne solution pour de la synchronisation modèle/code sous la plate-forme : Eclipse et Visual studio .NET.

A partir de cette étude, nous avons choisi la plate-forme Eclipse pour supporter l'atelier de génie logiciel de MACAO. L'objectif est maintenant de se concentrer sur la réalisation des modules **centrés** sur la conception des IHM.

Mais tout d'abord, il faut examiner comment prendre en compte la conception des IHM dans le processus de Capgemini. En particulier, Capgemini privilégie et soigne ses relations avec ses clients et utilise pour atteindre l'objectif les « ateliers participatifs » dans les phases amont de capture des besoins. Notre travail participe à l'insertion des modèles d'IHM dans la démarche de l'entreprise.

THESE-MACAO	Juin 2008	84 / 266
	Nicolas FERRY	

Chapitre II : Les ateliers participatifs

Les ateliers participatifs sont des réunions qui font intervenir la maîtrise d'ouvrage, les utilisateurs finaux et Capgemini notamment durant la phase d'analyse afin de spécifier ensemble le logiciel. La réunion est conduite par un animateur qui guide et oriente la séance. Le chef de projet, les architectes et les personnes concernées retranscrivent les besoins exprimés. Généralement, les ateliers participatifs utilisent un long « Brown paper » ayant l'apparence d'un ruban déroulé sur le mur. Toutes les idées et points de discussion sont notés par post-it sur le ruban. Ainsi que tous les participants voient et participent à la notification des besoins.

C.II.1 - Capture des exigences d'IHM

La capture des exigences met généralement en relation la maîtrise d'oeuvre et la maîtrise d'ouvrage afin de les faire converger vers une solution satisfaisante en terme de qualité. Plusieurs types de qualité recherchée peuvent exister : nous accepterons l'hypothèse que les participants recherchent une solution privilégiant la satisfaction générale d'un maximum de personnes.

Ce qui compte au final dans la préparation d'un produit informatique c'est la nécessité d'acquérir deux choses : les données et les traitements. A cela MACAO préconise d'ajouter « l'interface » pour les programmes interactifs. Pratiquement toutes les applications demandées aux SSII ont une interface utilisateur et de plus l'architecture 3 voir N tiers classique intègre dorénavant l'IHM comme couche à part entière. Avec l'arrivée des architectures objets, nous remarquons que l'interface est un même concept qui se retrouve dorénavant à plusieurs niveaux : dans les classes, les composants, les modules, les services, les plugins, la couche IHM bien sûr, etc. Elle est l'élément situé à la frontière entre l'utilisateur et le système d'information, elle a un rôle primordial puisque c'est la première en relation avec l'utilisateur. MACAO souhaite la satisfaction du client et des utilisateurs, c'est pour cela que les utilisateurs sont placés au centre du système. Ainsi MACAO fait participer l'utilisateur et l'intègre dès les premières phases et tout au long du cycle de vie [132]. Elle considère ainsi l'interface comme la composante majeure d'interaction avec lui.

Mais quel est l'intérêt d'introduire les IHM dans les exigences ?

Les IHM se complexifient, non seulement parce que les utilisateurs deviennent de plus en plus exigeants, mais aussi parce que les IHM sont fonctionnellement et techniquement plus diversifiées. Il faut aussi adapter l'IHM suivant les utilisateurs [150][155]. Aujourd'hui, il faut

THESE-MACAO	Juin 2008	85 / 266
	Nicolas FERRY	

compter avec des architectures diverses : des clients dits légers, des clients riches, des clients lourds, mais aussi avec la multiplicité des langages, l'apparition de la multimodalité,... La complexité est double puisqu'elle est liée à la quantité applicative à produire mais aussi à l'architecture et la manière dont celle-ci doit communiquer avec de multiples couches. En effet, la conception d'IHM applique des schémas conceptuels de plus en plus sophistiqués, nous pouvons citer l'inversion de contrôle, les usines de composants, les patrons modèle-vue-contrôleur, et bien d'autres encore. Cela augmente le temps de réalisation d'un prototype apte à valider la plate-forme technique et nécessite des compétences techniques élevées.

Actuellement, la prise en compte de la modélisation des IHM est très peu considérée dans les phases initiales d'un projet. L'étude menée lors de notre série d'entretiens auprès de chefs de projets de Capgemini, a conduit à cette conclusion. Nous avons également constaté que bien souvent la conception de l'IHM est laissée à la seule responsabilité des développeurs. Bien que pour certains projets, des maquettes soient proposées au client afin de valider une ergonomie générale, très peu de réflexion est menée en partenariat avec l'utilisateur final.

Avec MACAO, il est souhaitable de réaliser des entretiens avec les utilisateurs finaux sous forme d'ateliers participatifs dont l'un des buts est de concevoir une IHM adaptée aux besoins ou qui favorise une solution basée sur des compromis. Or le dialogue avec l'utilisateur passe nécessairement par un langage commun, non technique, permettant de modéliser de façon synthétique et claire ses principales exigences : enchaînement des menus, informations à afficher, saisies à réaliser, contrôles à effectuer...

C.II.2 - Ateliers participatifs à Capgemini

Une des « bonnes pratiques » pour la saisie des besoins et exigences des clients est de privilégier le contact de proximité sous forme d'atelier participatif. Les ateliers participatifs sont menés par un animateur, un chef de projet, des experts pour l'architecture, des experts des bases de données,...

Les ateliers participatifs ont plusieurs objectifs, tels que faciliter le dialogue et l'échange avec le client, acquérir et comprendre son besoin, préciser les zones d'ombre ou de flou, faire converger la solution.

Durant la phase d'analyse, un déroulement global possible est d'associer aux ateliers participatifs une approche qui vise à corréliser la conception par les modèles avec l'entretien client qui n'est pas forcément informaticien.

THESE-MACAO	Juin 2008	86 / 266
	Nicolas FERRY	

En effet, l'expérience montre qu'après la présentation de certains diagrammes UML aux clients, le formalisme n'est pas toujours compris. L'acceptation dépend grandement du client notamment si celui-ci a déjà utilisé ou non le formalisme et s'il est convaincu de son efficacité. Mais hormis certains domaines spécifiques, il faut bien avouer que les clients adoptent peu les représentations informatiques souvent éloignées de leur domaine.

Il faut privilégier un maximum le dialogue et l'échange sur ce qu'il connaît de mieux : son métier. La modélisation de la conception doit lui être présentée dans la limite de sa compréhension et de son acceptation. Par contre, un feedback reformulé par le client est nécessaire pour bien converger.

Parallèlement à la diction du client, il est nécessaire de modéliser ses besoins avec des langages et des représentations plus structurés (SNI, UML, Merise,...) pour faire naître une vision et une compréhension progressive du problème. Pour illustrer, nous pouvons imaginer sur un axe horizontal l'acquisition de ce que les SSII appellent le « business process » ou le procédé du client. Ce procédé peut être représenté par plusieurs formalismes, parmi ceux-ci, remarquons les diagrammes de tâches comme par exemple le DCT de MACAO ou BPMN. Les tâches sont actuellement un des moyens pertinents pour la compréhension du besoin [143] [147].

Nous pouvons remarquer que les diagrammes UML des cas d'utilisation sont aussi une vision métier mais trop généraliste. Alors que le tableau des fonctions (ou TDT pour Macao) représente un découpage fonctionnel plus précis. Bien sûr entre ces deux extrêmes, plusieurs critères peuvent conseiller sur le niveau de précision à atteindre. Nous pouvons imaginer cela par la notion de profondeur ou de granularité de l'analyse.

Prenons l'exemple dans lequel nous concevons une architecture standard de type 3-tiers. L'analyse se découperait suivant l'architecture adoptée c'est-à-dire :

- l'analyse de la couche Présentation (ou vue) étudiée par les modèles d'IHM.
- l'analyse de la couche Logique (ou Contrôleur / ou serveur d'application) étudiée par UML (classes, composants, structure composite, interaction pour les Web Services,...)
- l'analyse de la couche Persistance (ou Model / serveur de données) étudiée par (MCD ou classes)

THESE-MACAO	Juin 2008	87 / 266
	Nicolas FERRY	

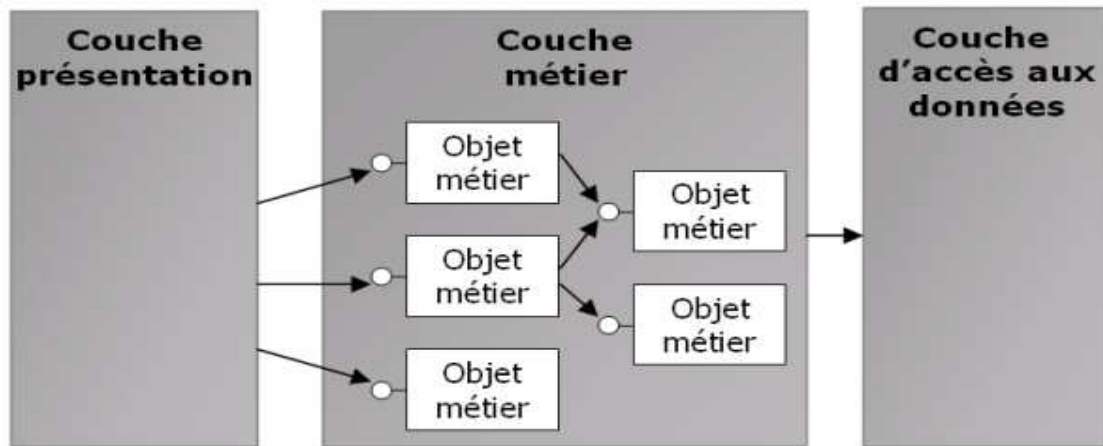


Figure 14 : Représentation d'une application multi-tiers en couches

L'analyse au niveau de chaque couche décrit une partie ou un aspect du système qui peut être à un niveau global ou plus fin suivant les diagrammes utilisés dans chaque axe de l'architecture. L'intérêt principal réside dans le fait de concevoir chaque diagramme comme un élément à part entière qui décrit une partie de la problématique. Le passage d'un modèle à un autre suit le processus global de conception fixé par l'architecte. Le mécanisme de transfert d'un modèle à un autre se fait par transformation de modèle. Le processus de transformation inverse existe avec des langages comme BOTL [18].

Grâce à une plate-forme de transformation, les modèles physiques de chaque couche donnent une partie du codage du logiciel qui est fusionné pour donner l'application finale. Nous verrons plus bas avec le générateur comment cela est possible concrètement, en attendant nous dirons que la plate-forme de transformation doit supporter le mécanisme de tissages des modèles ou d'aspects.

C.II.3 - Découpage en niveaux de l'axe d'analyse des IHM

Soulignons l'importance d'une séparation entre le quoi-fonctionnel et le comment-technologique comme le montre la figure ci-dessous.

THESE-MACAO	Juin 2008	88 / 266
	Nicolas FERRY	

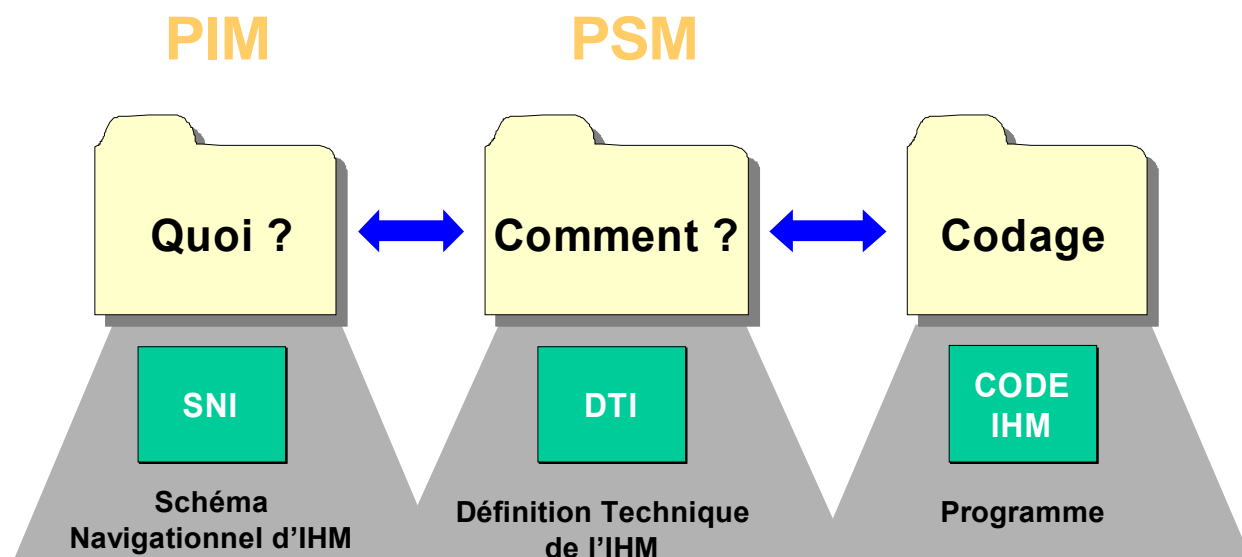


Figure 15 : Les différents niveaux de l'axe IHM

Le niveau conceptuel est géré par le SNI. Le niveau logique est appelé MLI (Modèle Logique des IHM ou DTI pour Définition Technique de l'IHM). Le niveau physique correspond au code de l'IHM. Le niveau logique regroupe lui-même des diagrammes dynamiques : les Schémas d'Enchaînement (SE) et des diagrammes statiques représentant la Définition des Eléments (DE) c'est-à-dire le dessin de la maquette de l'écran.

C.II.4 - Séparation du quoi-fonctionnel et du comment-Technique

Dans la méthode MACAO, les modèles d'IHM sont utilisés lors de l'étape d'analyse globale pour l'acquisition des exigences, lors de la conception globale pour décrire l'architecture de l'IHM, et lors du développement pour réaliser des maquettes et générer le code logiciel de l'IHM.

Pour cela, MACAO propose trois niveaux de modèles qui en font l'originalité. L'intérêt est de pouvoir préciser les frontières de l'IHM entre le « Quoi-Fonctionnel » de la phase d'analyse, le « Comment-Technologique » de la phase de conception et le « Comment-Visuel » de la présentation finale. Le tableau suivant récapitule les frontières :

THESE-MACAO	Juin 2008	89 / 266
	Nicolas FERRY	

Niveau	Schémas / diagrammes	Périmètre
Conceptuel	SNI	Quoi-Fonctionnel
Logique	SEF, SEP, SEM (SE)	Comment-Technologique
	DEF, DEP, DEM (DEV)	Comment-Visuel
Physique	Maquettes, Code	Le résultat (Codage)

Tableau 4 : les schémas de la méthode MACAO suivant le niveau d'approfondissement de l'IHM.

Ainsi au niveau conceptuel le SNI définit la navigation globale dans une IHM abstraite. Le niveau logique raffine le niveau précédent et permet de prendre en considération des domaines technologiques spécifiques. Ainsi, le SEF (Schéma d'Enchaînement des fenêtres) et le DEF (DEfinition de Fenêtres) sont-ils adaptés à la représentation d'une IHM de type fenêtré (client lourd et riche) alors que le SEP (Schéma d'Enchaînement des pages Web) et le DEP (DEfinition des pages) sont mieux adaptés à la représentation Web (client léger). Enfin, le SEM (Schéma d'Enchaînement Multimodal) et le DEM (DEfinition Multimodale) sont utilisés pour décrire des IHM faisant intervenir plusieurs modalités telles que le vocal ou le tactile.

Dans MACAO, les schémas de définition des éléments (DE) cités plus haut gardent encore une représentation abstraite. En effet, bien que spécialisés pour un domaine technologique particulier, ils utilisent une symbolique qui est une représentation de la plate-forme cible et/ou des thèmes graphiques. Ces modèles précisent les attributs visuels (ou autres dans le cas du multimodal) propres à chaque objet graphique.

Enfin, les maquettes du niveau physique ont pour objectif de présenter à l'utilisateur une vision réelle des objets de l'IHM. Elles sont fortement couplées au code puisqu'elles sont le résultat applicatif.

Le passage d'un niveau à l'autre se fait au moyen d'outils et de langages de transformation de modèles permettant de générer *in fine* le squelette du code source relatif à l'IHM.

C.II.5 - Architecture globale de l'axe IHM

Comme nous venons de le voir la modélisation de l'axe IHM a été découpée en étages où des diagrammes spécifiques jouent un rôle de définition par paliers. Le passage d'un diagramme à un autre se fait par le mécanisme de transformation de modèle.

Les travaux de l'équipe de Jean-Claude Tarby de Lille et notamment de Xavier lePallec sur les PMI [165][166] démontrent qu'il est possible de concevoir des métamodèles raffinés en utilisant des métamodèles successifs et incrémentaux. Ils ont utilisé une syntaxe d'opérateurs

THESE-MACAO	Juin 2008	90 / 266
	Nicolas FERRY	

pour composer un métamodèle incrémenté. Dans notre travail, nous avons repris le découpage des niveaux pour établir les métamodèles de chaque niveau.

La transformation de modèle fait partie du Model Driven Engineering ou Ingénierie Dirigée par les modèles. Son principe repose généralement sur un moteur qui va interpréter des règles comme avec ATL [15], MIA Software [50], AndroMDA [10], CODAGEN [26], etc. ou un langage de transformation par patrons comme OpenArchitectureWare.

Ces transformations sont le liant entre les modèles des différents niveaux. A priori, l'évolution vers de nouvelles technologies et donc vers de nouveaux modèles adaptés peut s'intégrer en écrivant les transformations respectives.

Nous avons défini le fonctionnement d'un transformateur simple. Le transformateur doit connaître en entrée les métamodèles du modèle source et du modèle cible. Il faut alors transmettre un modèle source au transformateur qui officie pour générer le modèle cible.

Ainsi en reprenant l'illustration des trois couches précédentes, nous pouvons détailler l'architecture globale de génération de l'axe vertical de l'IHM. Le Schéma Navigationnel des IHM au niveau conceptuel peut être transformé en Schéma d'Enchaînement (partie dynamique) puis enrichi avec la Définition des Elements (partie statique) avant que le code de l'IHM ne soit entièrement généré.

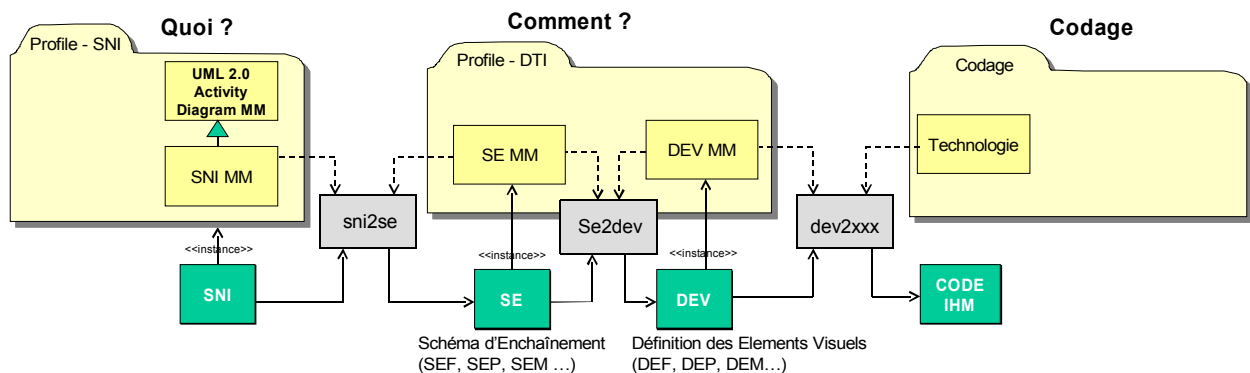


Figure 16 : Architecture de l'axe IHM avec transformateurs pour le passage de diagrammes de niveaux différents. Ici MM signifie Métamodèle. Ce schéma montre la conception d'une IHM à partir de modèles de niveaux d'abstraction différents.

THESE-MACAO	Juin 2008	91 / 266
	Nicolas FERRY	

Des travaux similaires ont été entrepris tels que ceux de l'équipe de Gaëlle Calvary [179] portant sur la plasticité des IHM. La plasticité est définie comme suit :

« En informatique ambiante, les objectifs de l'utilisateur peuvent émerger opportunément tout comme varie le contexte d'usage dans lequel il se trouve. En conséquence, les Interfaces Homme-Machine doivent être composables dynamiquement par l'utilisateur final et capables de s'adapter à des variations à l'utilisation, les anticiper, les apprendre, etc. On appelle Plasticité cette capacité d'adaptation. »

Par rapport à l'approche de MACAO pour les IHM, nous pouvons citer un certain nombre de ressemblances avec les travaux de Gaëlle Calvary et de son équipe :

- Les IHM sont centrées par rapport à l'utilisateur et ses besoins,
- Les métamodèles décrivent les modèles pour les transformations MDE,
- La conception des IHM respecte les qualités « d'utilisabilité » de la norme ISO 9126,
- Les transformations peuvent être effectuées à plusieurs niveaux d'abstraction,
- Les transformations sont vues comme du savoir-faire (expérience).
- Des outils permettent la génération sur des plates-formes d'implémentation de nature différente.

Par contre, MACAO se distingue essentiellement sur quelques points.

Dans MACAO c'est l'utilisateur qui décide de la structure de l'IHM or très souvent, il existe des différences entre les utilisateurs selon leurs besoins. Pour pallier cela, MACAO propose, lors de la conception, de réaliser un compromis entre les différentes tendances. Mais comme le propose Gaëlle Calvary, il serait beaucoup plus intéressant que l'IHM soit suffisamment plastique pour s'adapter à chaque profil utilisateur.

En cela, le SNI présente l'avantage de pouvoir modéliser de manière abstraite l'IHM suivant un métamodèle (méta-IHM) qui devrait faciliter la prise en compte de sa plasticité. Muni d'opérateurs adéquats, les transformations vers les niveaux inférieurs permettraient à l'IHM de s'adapter automatiquement à chaque profil utilisateur.

Un deuxième type de différence concerne la décomposition de la structure. MACAO utilise un modèle de tâches pour décrire le processus utilisateur à partir duquel est construit un modèle abstrait d'IHM, le SNI, avant de définir un niveau logique.

Ici l'approche est de créer des entités et des fonctions exportées (niveau FCA) à un diagramme de tâches (au niveau DC). MACAO n'a pas de niveau au-dessus des modèles organisationnels

THESE-MACAO	Juin 2008	92 / 266
	Nicolas FERRY	

(Tâches) puisque les utilisateurs (Rôles) produisent des tâches (Fonctions) en manipulant des documents (Données) dont certaines tâches peuvent être interactives (IHM). Aussi une application est-elle découpée en 3 perspectives majeures : IHM, Contrôleur, Données (applications n-tiers).

Enfin, MACAO ajoute la notion de fusion de différents modèles pour produire la solution finale à partir des différentes composantes de la structure. Nous soulignons le fait que cette fusion peut être réalisée par la voie du tissage de modèles. Cette approche est détaillée au chapitre C.V.4 : La fusion de modèles.

Pour UsiXML [176] de l'équipe de Jean Vanderdonckt, la démarche adoptée est assez proche de MACAO en ce qui concerne l'objectif principal qui veut que le modèle initial ne soit pas technique afin de pouvoir être utilisé par des non informaticiens et être indépendant de toute plate-forme de développement. Cependant, il existe un aspect important qui différencie leur approche de la nôtre : dans UsiXML [177], le modèle initial utilise la notion de widget pour décrire les besoins d'IHM. Cette définition est prématurée car un besoin élémentaire tel qu'un menu (c'est un exemple) peut se traduire par différents types de widget (menu déroulant, menu contextuel, ensemble de boutons, ensemble d'hyperliens...) dont on n'a pas forcément connaissance au moment de la conception initiale.

C'est pourquoi, dans le SNI, le terme d'unité de dialogue (UD) a été préféré car il est une notion plus large et moins portée sur une technologie particulière. En fait dans MACAO la modélisation initiale de UsiXML se retrouve, mais décomposée en deux modèles : le SNI et le SEF (ou le SEP pour les pages WEB, ou le SEM pour les IHM multimodales).

C.II.6 - Bilan

Pour la réalisation d'un projet informatique, nous avons rappelé l'importance de l'entente et de la convergence des participants pour l'établissement d'un compromis vers une solution commune satisfaisant au mieux l'ensemble des participants. Nous avons aussi rappelé le rôle primordial de l'utilisateur final et sa participation active dans l'équipe. Nous avons soutenu que pour des projets significatifs, la conception de l'IHM doit être considérée au même titre que la couche métier ou la couche de données par l'ensemble des intervenants dès les premières phases du projet. Pour cela, nous avons présenté le mode de fonctionnement que favorise Capgemini avec ses clients avec les ateliers participatifs. Ensuite, nous avons proposé une manière de procéder en atelier qui introduit les modèles d'IHM. Celle-ci permet notamment de recueillir les informations pertinentes en accord avec l'architecture globale choisie pour l'application.

THESE-MACAO	Juin 2008	93 / 266
	Nicolas FERRY	

L'application est conçue en suivant une décomposition de l'architecture de l'axe IHM qui est découpé en trois niveaux. Il faut veiller à séparer le conceptuel acquis avec le client, de la logique pour l'ingénierie et le physique présenté en retour à l'utilisateur final. Nous définissons la frontière de ces niveaux par la réponse aux questions du Quoi-fonctionnel, du Comment-technologique et du comment-Visuel, de la réalisation concrète. Macao introduit un modèle à chaque niveau. Chaque modèle est informatiquement géré par son métamodèle respectif. Dès lors, il est possible de définir ces métamodèles comme des raffinements et des incréments des précédents. Le lien étant fait au niveau des métamodèles, les moteurs de transformation de l'ingénierie dirigée par les modèles permettent de réaliser les différentes mutations des représentations. Un transformateur est un élément de base de cette transformation. La composition de plusieurs transformateurs chaînés comme présentés dans l'architecture globale permet de générer les transformations de l'axe IHM. Il est important de noter que ce principe peut aussi être profitable à d'autres branches de l'architecture logicielle.

L'étape suivante consiste à réaliser l'architecture globale de l'axe IHM. Il faut à présent construire le premier élément qui permette d'éditer graphiquement les diagrammes conceptuels de type SNI.

Chapitre III : L'éditeur graphique de modèle SNI

C.III.1 - L'éditeur de SNI : VisualSNI

VisualSNI est un éditeur graphique pour modèles SNI capable de modéliser la plupart des interfaces homme-machine grâce à son haut degré d'abstraction. Il vise à outiller la méthode MACAO pour la conception informatisée. Il comble le manque d'outil pour ce genre de travail en offrant une vue globale de la navigation dans les IHM, les droits d'accès, et la couverture des fonctionnalités par prototype.

VisualSNI se présente sous la forme d'un plugin pour Eclipse [37]. Il a été construit à partir des briques fondamentales EMF [71], GEF [66] et Draw2D. L'éditeur supporte toutes les fonctionnalités de bases comme la gestion de fichier SNI, la création/modification/suppression d'unités de dialogues, l'annulation de commandes, l'impression, etc. Il est basé sur le métamodèle du SNI créé spécialement pour répondre à la particularité de la conception d'IHM. L'éditeur se présente comme une partie de l'ensemble de la plate-forme MACAO sur la branche de développement des interfaces.

III.1.1 - Objectifs :

VisualSNI doit être capable de gérer les diagrammes SNI, de les sauvegarder et de les recharger au format standard XMI 2.0 [61]. Il utilise pour cela les mécanismes des modules tiers d'Eclipse qui permettent d'enregistrer suivant la norme XMI 2.0.

Il fournit aussi un éditeur graphique composé d'une zone de conception et d'une palette de composants d'édition.

L'éditeur sera géré en temps réel. Des travaux empiriques ont démontrés que la plupart des langages visuels nécessitant une lecture incrémentale, des corrections d'erreurs à la volée, et une génération en préservant la sémantique pouvaient être réalisés en temps réel [153].

III.1.2 - Périmètre fonctionnel

Dans l'étude du processus MACAO, nous avons défini les principaux « cas d'utilisations » d'une équipe projet. Nous décrivons ici comment se place l'utilisation de l'éditeur VisualSNI dans les cas d'utilisation de chaque intervenant.

THESE-MACAO	Juin 2008	95 / 266
	Nicolas FERRY	

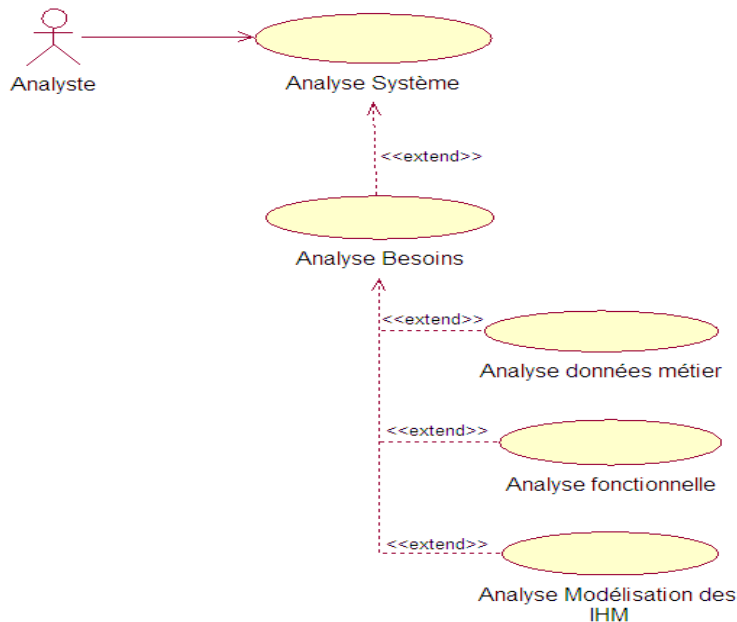


Figure 17 : Schéma des principaux cas d'utilisation du rôle Analyste avec inclusion de la partie IHM

L'analyste utilise le SNI dans la phase de recueil des besoins lorsqu'il modélise l'IHM. Souvent, il aura recours au mode « Esquisse » c'est-à-dire la conception à la volée avec le client dans les ateliers.

THESE-MACAO	Juin 2008	96 / 266
	Nicolas FERRY	

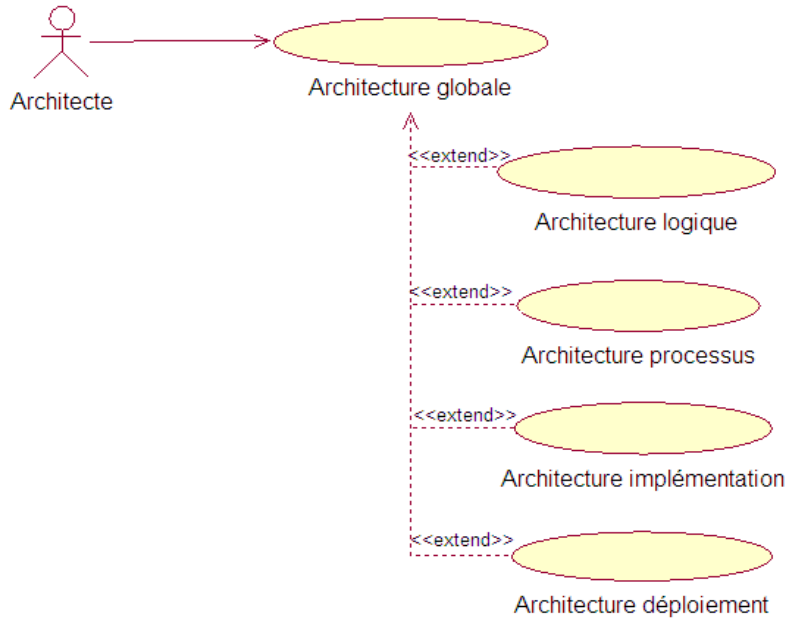


Figure 18 : Schéma des principaux cas d'utilisation d'un architecte avec inclusion de la partie IHM

Pour l'architecte, lors de la conception de l'architecture globale de l'application ou l'architecture détaillée celui-ci pourra utiliser l'outil VisualNSI pour la conception de SNI en mode global.

THESE-MACAO	Juin 2008	97 / 266
	Nicolas FERRY	

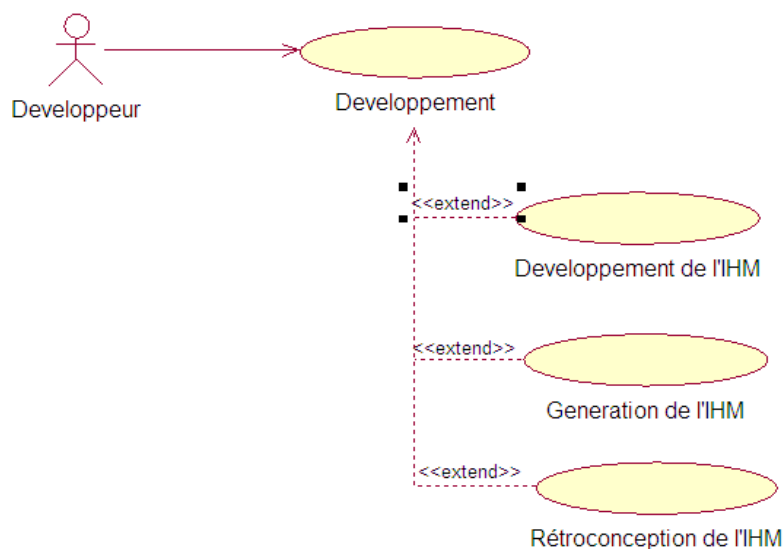


Figure 19 : Schéma des principaux cas d'utilisation du rôle Développeur tenant en compte la partie IHM

Le développeur utilisera les schémas produits par les entretiens et la conception affinée de l'architecte pour le développement de l'IHM finale. Celui-ci pourra avoir recours à la génération automatique, pour la maquette ou pour générer le squelette du code. Enfin idéalement, pour des étapes de mises à jour du modèle après développement, il devra pouvoir utiliser la transformée inverse pour mettre à jour le modèle.

III.1.3 - Description détaillée de chaque fonction prévue dans l'éditeur

Fonction F01 : La création d'un nouveau diagramme

Cette fonction utilise un assistant de création qui va inviter l'utilisateur à saisir les informations nécessaires à la création du nouveau diagramme. Dans l'assistant, l'utilisateur renseigne les informations sur le SNI. L'utilisateur saisit le nom et le chemin du fichier de manière classique à Eclipse. L'extension du fichier *.sni est proposée par défaut.

A la fin de l'assistant sur validation, le nouveau diagramme SNI vierge est créé dans les ressources du projet.

Une fois la planche créée, l'utilisateur peut saisir des informations complémentaires du SNI. Pour rappel, ces informations correspondent au Qui, Où (quel projet), Quand (date de création), Pourquoi (Le problème), Comment (Solution)...

THESE-MACAO	Juin 2008	98 / 266
	Nicolas FERRY	

Fonction F02 : Création d'UD élémentaires

L'édition du diagramme s'opère de manière standard à Eclipse au travers d'une vue : l'éditeur graphique du SNI. Celle-ci comporte deux parties : un espace de création et une palette d'outils.

La palette se compose de quatre types d'outils : création de liens, création de nœuds, création d'unités de dialogues élémentaires et enfin création d'unités de dialogue composées.

C'est dans la catégorie des outils pour UD élémentaires que sont présentées les différentes UDE de bases de MACAO. L'utilisateur choisi l'unité de dialogue qu'il souhaite créer et la "dépose" dans l'espace de création.

Fonction F03 : Création d'UD composées

La catégorie des outils pour UD composées présente les différentes unités de dialogues composées avec la boîte de groupage. L'utilisateur choisi la boîte de groupage qu'il souhaite créer et la "dépose" dans l'espace de création.

Fonction F04 : Création de Nœuds

Les nœuds sont des objets qui permettent des raccordements entre les unités de dialogues. On trouve par exemple les objets de type Décision et Jonction au sens UML. Le mécanisme de création est similaire aux fonctions précédentes.

Fonction F05 : Création de Liens de raccordement

Les liens permettent de relier les objets de type Nœuds ou UD ensemble. Le raccordement se fait à des endroits précis appelés ports. Un lien relie un port d'entrée à un port de sortie. Les ports sont transparents pour l'utilisateur qui ne s'en soucie pas, ils servent cependant à vérifier la cohérence générale des règles de création du SNI. Les liens pourront contenir plusieurs sous points pour créer des "polylignes" afin de garantir un routage visuellement agréable. Les liens sont implicitement orientés dans leur sens de création d'un port OUT vers un port IN.

Fonction F06 : Editer/Modifier les propriétés d'un objet

L'éditeur graphique permet aussi à l'utilisateur de sélectionner un objet du SNI. La vue Properties (standard d'Eclipse) récapitule les différentes caractéristiques de l'objet et permet de les modifier lorsque cela est autorisé. La modification de propriétés graphiques entraîne un rafraîchissement graphique de l'éditeur. Les autres propriétés influencent les vues suivant leur domaine.

Afin de simplifier l'édition des propriétés d'un objet, un assistant est créé pour chaque objet visible du SNI afin de pouvoir éditer les paramètres plus intuitivement. Cela revient au même que l'édition via la vue Properties.

THESE-MACAO	Juin 2008	99 / 266
	Nicolas FERRY	

Fonction F07 : Déplacement d'objets

L'utilisateur a la possibilité de sélectionner un ou plusieurs objets graphiques et de les déplacer à sa guise dans le schéma. Cette opération est réalisée par le mécanisme de Drag And Drop inclus dans Eclipse.

Fonction F08 : Redimensionnement d'un objet du SNI

L'utilisateur peut aussi redimensionner la taille des objets afin de créer des proportions différentes entre objets.

Fonction F09 : Suppression d'un objet du SNI

Un objet ou des objets sélectionnés peuvent être supprimés du SNI. La politique de suppression retenue consiste à supprimer l'objet sélectionné ainsi que tous les liens qui lui sont rattachés.

Fonction F10 : Le panoramique

L'utilisateur aura également la possibilité de déplacer la fenêtre de vision de l'éditeur dans une vue d'ensemble synthétisée du diagramme.

Fonction F11 : Imprimer le diagramme

En cliquant avec le bouton droit de la souris sur le fond du diagramme, l'utilisateur fait apparaître un menu déroulant avec des fonctions générales aux diagrammes. Il y a trois types d'impression :

- L'impression d'une planche qui lui permet de sortir le schéma sur document papier.
- L'impression des planches sélectionnées.
- L'impression de toutes les planches en une seule fois.

Fonction F12 : Chargement/Ouverture d'un diagramme

Soit en cliquant sur nouvel objet (fonction new de Eclipse) soit en double-cliquant sur une ressource de type fichier SNI, (format XMI 2.0) le système recharge un diagramme existant et ouvre un éditeur graphique.

Fonction F13 : Sauvegarde d'un diagramme

L'utilisateur peut sauvegarder un diagramme. La sauvegarde s'opère en format XMI 2.0 suivant la norme de Eclipse.

Fonction F14 : Copier / Couper / Coller

L'utilisateur peut créer de nouvelles UD avec le mécanisme de Copier / Coller. De même, il peut déplacer des éléments entre diagrammes de cette façon.

THESE-MACAO	Juin 2008	100 / 266
	Nicolas FERRY	

Fonction F15 : Undo / Redo

Chaque commande de l'utilisateur est empilé dans une pile des traitements. Chaque commande peut être défaire ou refaire suivant son ordre de création.

Fonction F16 : Contrôler et générer les références

Les étiquettes peuvent pointer sur d'autres planches de SNI. Les références externes qui pointe sur une planche peuvent être calculées automatiquement afin de connaître les références entrantes.

THESE-MACAO	Juin 2008	101 / 266
	Nicolas FERRY	

C.III.2 - Maquette de l'éditeur

A travers cette maquette est représentée la plateforme Eclipse. Dans cette configuration, le plugin utilise une perspective standard (Java dans l'exemple) avec un certain nombre de vues ouvertes.

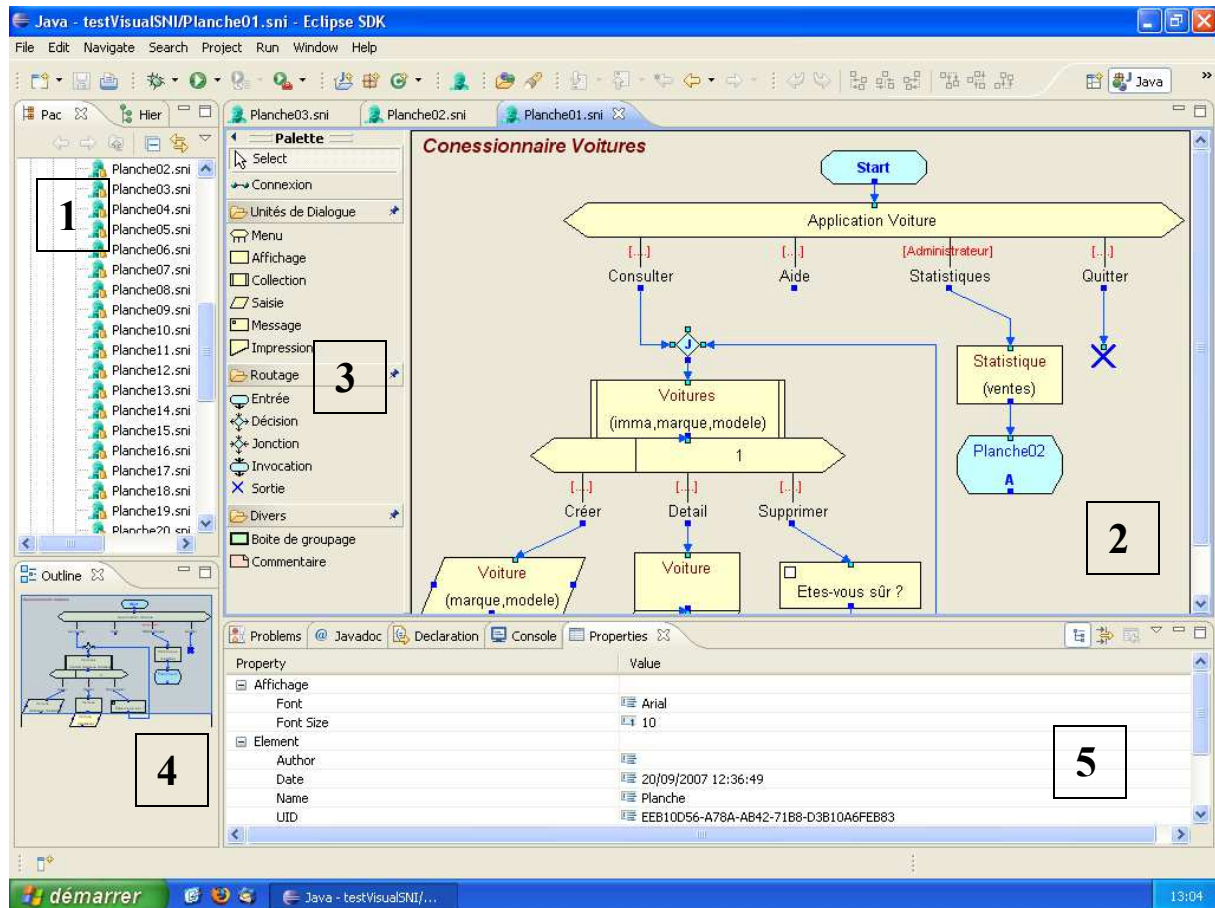


Figure 20 : Insertion de VisualSNI dans Eclipse. La vue sur les paquetages et les ressources N°1 permet de voir la liste des planches de SNI présente dans le projet. La vue N°2 présente l'éditeur graphique de SNI et sa palette associée N°3. La vue N°4 outline permet d'avoir une vision globale du schéma afin de naviguer plus rapidement à l'intérieur. Enfin, la vue n°5 des Properties récapitule et permet de modifier les propriétés des objets.

THESE-MACAO	Juin 2008	102 / 266
	Nicolas FERRY	

C.III.3 - Manipulation de SNI

Nous allons dans ce chapitre mettre en oeuvre un exemple de SNI afin d'expliquer la création de planches pour un projet qui a été développé à Capgemini. Celui-ci montrera comment se manipule l'éditeur pour réaliser un SNI.

L'utilisateur a la possibilité de créer les unités de dialogues à partir d'une palette d'outils. Ces unités, rappelons le, permettent de modéliser une IHM, aussi complexe soit-elle, en utilisant grosso modo six types d'Unités de Dialogue Élémentaires (UDE). Nous les retrouvons en jaune dans la palette de l'éditeur.

Une palette d'outils comportant tous les symboles nécessaires est proposée par VisualSNI. La palette est composée de quatre parties :

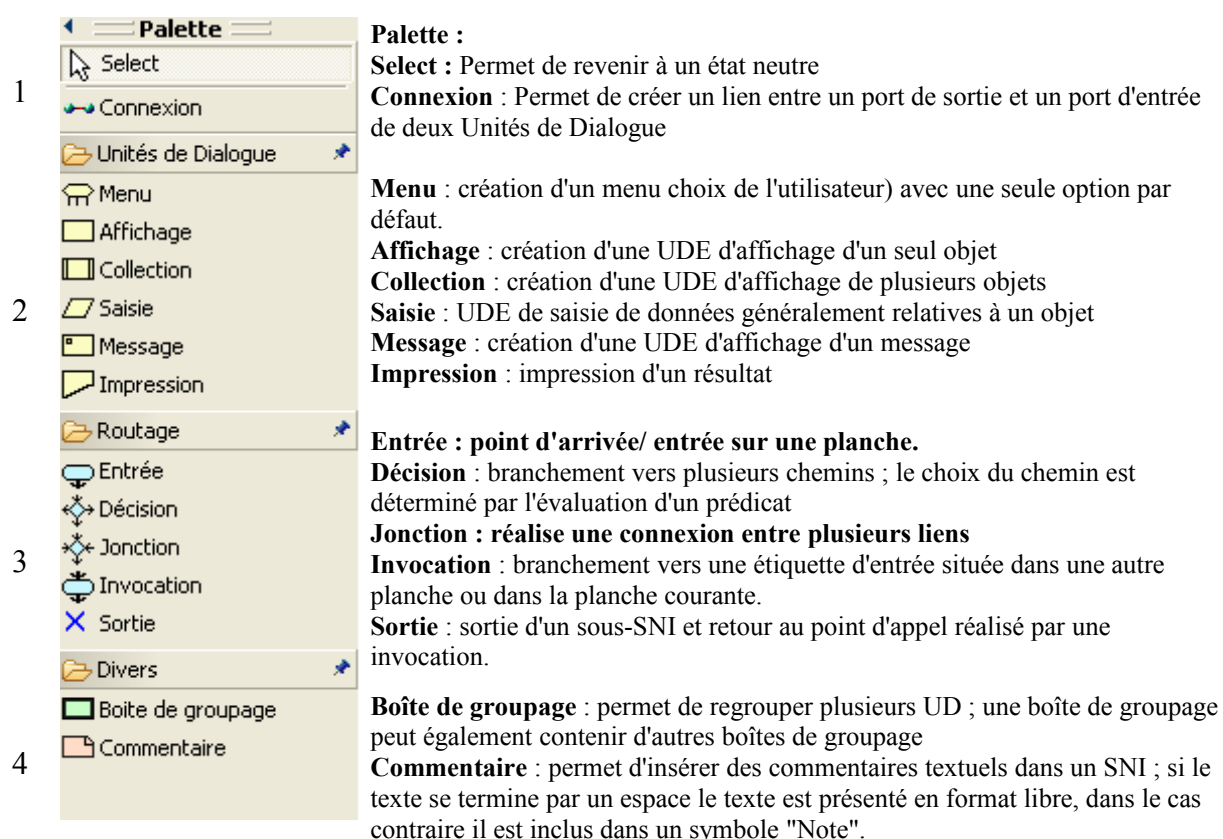


Figure 21 : La palette d'édition permet de créer les objets du SNI

III.3.1 - Exemple de SNI : le projet Sicli

La société Sicli, leader français pour la vente, l'installation et la maintenance d'extincteurs, désenfumage, etc. a souhaité équiper ses techniciens de PDA mobiles pour faire le suivi de leurs interventions et dépannages sur les sites clients. Capgemini a réalisé un logiciel de suivi des interventions.

Sicli a été l'un des tous premiers projets à utiliser le schéma Navigationnel des IHM de MACAO. Nous avons réalisé sur un scénario nominal un SNI complet pour montrer aux équipes de Capgemini un diagramme d'IHM associé à un projet concret. Le schéma qui est présenté ici est restreint pour des questions de place.

III.3.2 - Construction du SNI

Nous allons construire le SNI de l'application Sicli étape par étape. Le scénario adopté est celui d'un technicien qui se connecte sur l'application du PDA, qui consulte son planning des interventions et décide de choisir son intervention pour aller vérifier l'état des extincteurs d'une installation.

La construction d'un SNI débute toujours par la création d'une première planche. Chaque planche possède un intitulé par défaut qu'il est possible de modifier. Il est également possible de renseigner diverses informations concernant la planche de SNI en remplissant l'auteur, la date, le projet,... Pour débiter une planche de SNI, il est conseillé de commencer par une étiquette "Entrée" comportant le numéro de planche ou un libellé, celle-ci servira pour être appelée depuis un autre diagramme ou comme racine de l'IHM.

Nous voulons modéliser un écran d'attente de chargement « splash screen » et arriver sur un écran de login. Le splash screen est un écran de base qui est un affichage simple. L'écran de login par contre nécessite l'authentification de l'utilisateur, c'est une saisie d'information avec une demande de saisie du login et du mot de passe.

Ces écrans s'enchaînent normalement, nous les relierons donc avec des liens de connexions simples. Et voici un aperçu de la modélisation à cette étape :

THESE-MACAO	Juin 2008	104 / 266
	Nicolas FERRY	

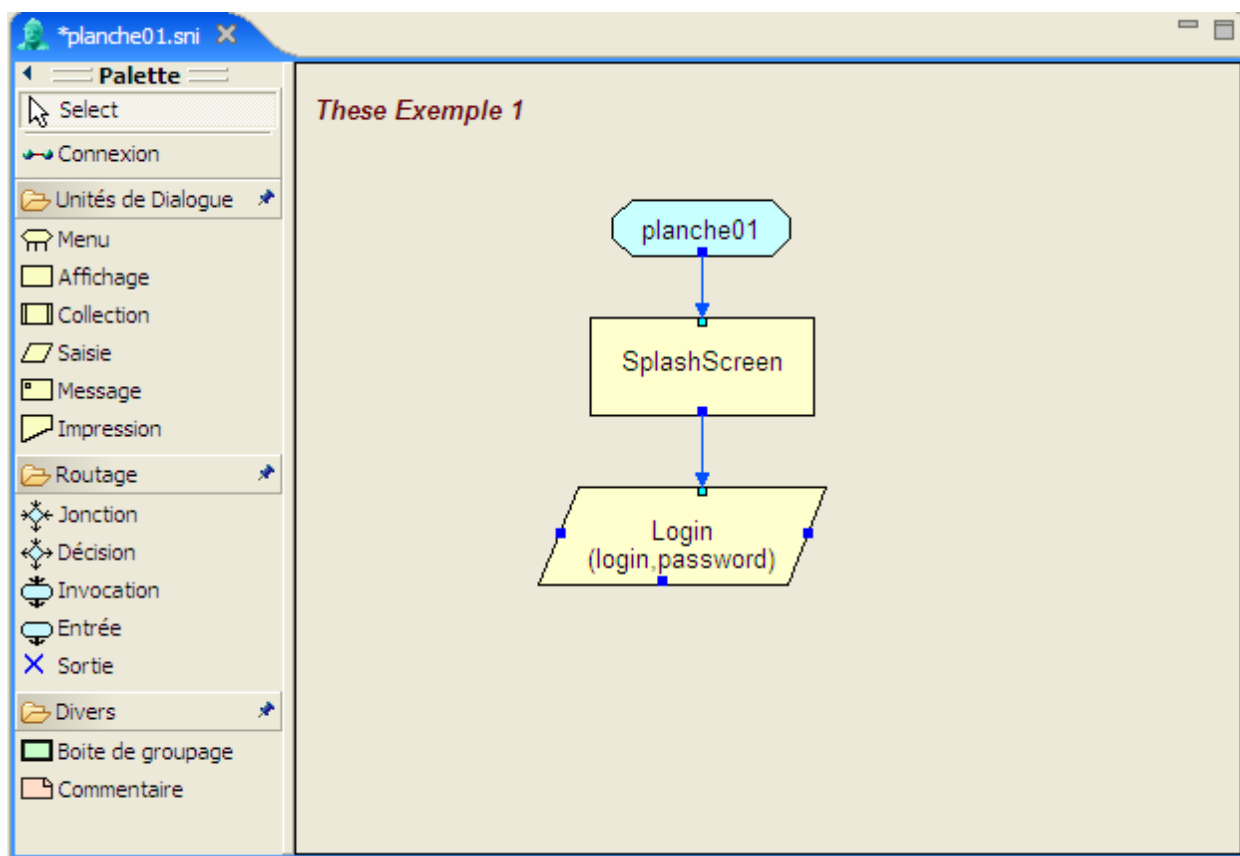


Figure 22 : Construction de l'écran de login

Dans l'étape suivante, nous voulons introduire la possibilité qu'aura l'utilisateur de pouvoir changer son mot de passe et la gestion d'un message d'erreur si la saisie du login/mot de passe est erronée. Enfin, si l'identification est correcte le technicien arrive sur l'écran principal de l'application.

Dans les options de l'écran principal, seul l'accès au planning nous intéresse dans ce scénario. Voici le SNI correspondant :

THESE-MACAO	Juin 2008	105 / 266
	Nicolas FERRY	

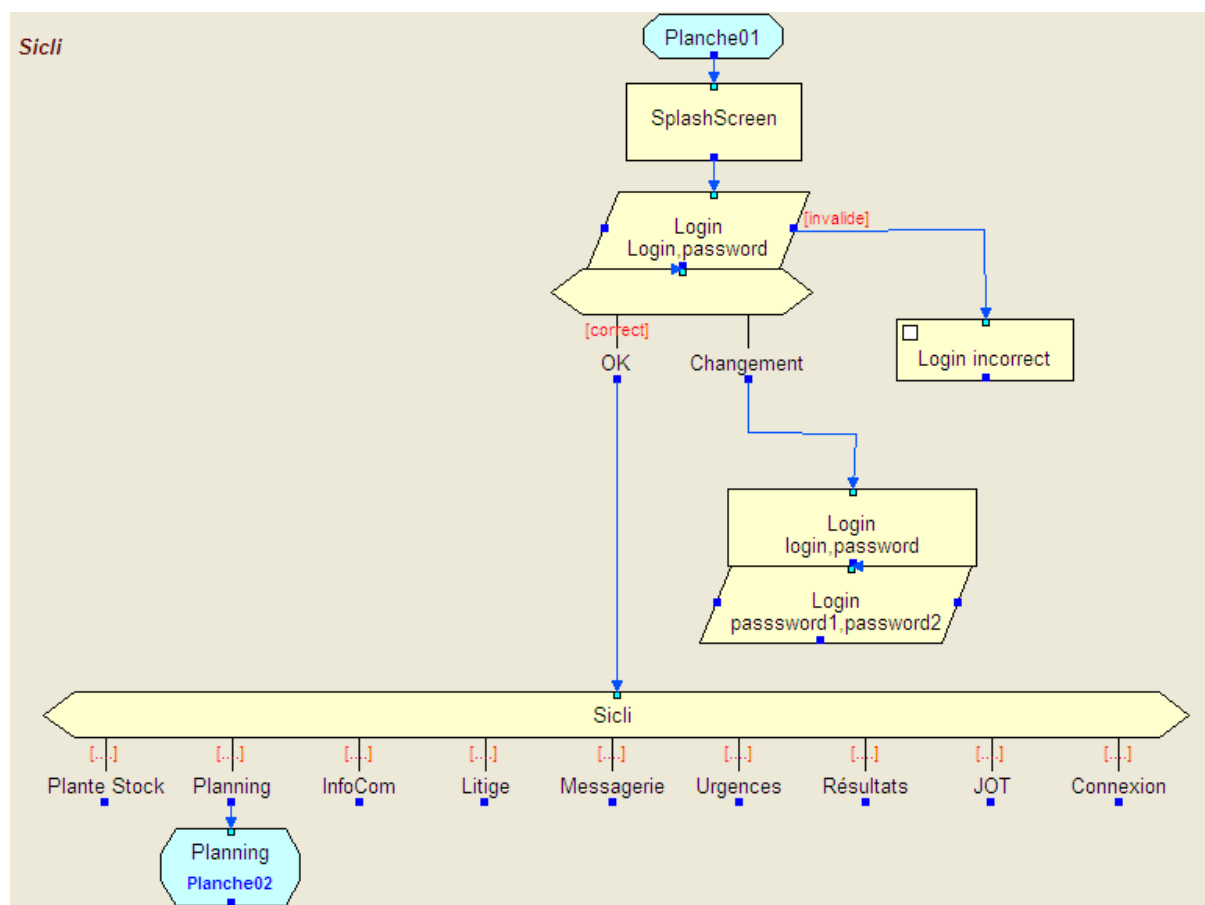


Figure 23: Exemple de construction de l'application « Sicli »

Pour garder de la clarté et pour la "réutilisabilité" de certains écrans, nous découpons le SNI en plusieurs planches. Dans l'écran de planning, le technicien peut consulter ses demandes d'interventions synchronisées à partir de sa maison mère. Il accède à une liste des fiches d'interventions, elles sont dénommées FT (Fiche d'intervention Technique). Le planning lui permet de consulter les FT sur la semaine à venir. Le planning est une unité composée d'une liste de jours (de la semaine) et de la liste des FT pour le jour sélectionné.

Il peut sélectionner une ou plusieurs FT pour définir son "panier" c'est-à-dire l'ensemble des interventions de la journée. Il peut aussi consulter les consignes d'une intervention particulière. L'option de recherche des FT suivant leur proximité ne sera pas traitée dans cet exemple. Voici la planche modélisant ce contexte :

THESE-MACAO	Juin 2008	106 / 266
	Nicolas FERRY	

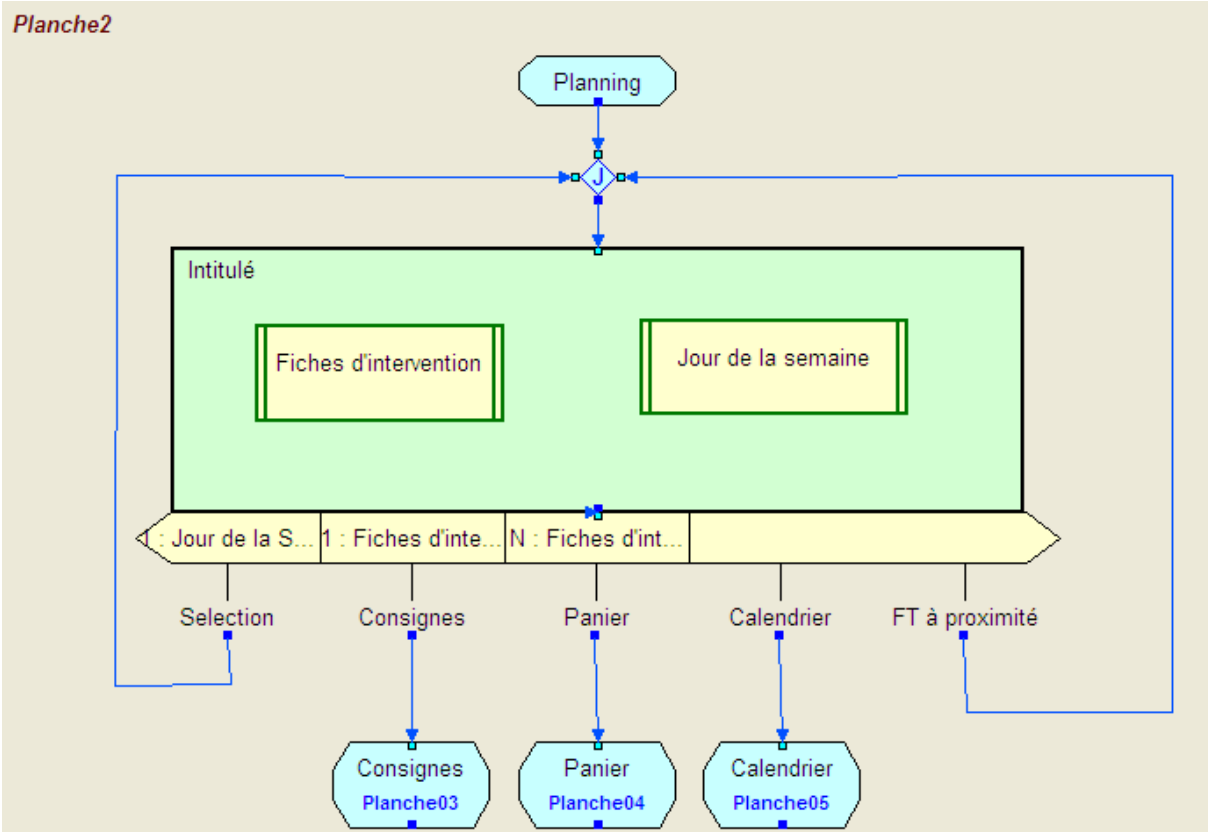


Figure 24 : Exemple de construction de l'application «Sicli»

Le technicien remplit son panier en choisissant N fiches d'interventions pour la journée. Celles-ci peuvent être filtrées suivant leur état : si elles sont en cours, celles qui ont été vérifiées ou toutes. Le technicien peut décider de choisir une intervention dans la liste et en consulte les consignes avant d'intervenir.

THESE-MACAO	Juin 2008	107 / 266
	Nicolas FERRY	

Planche3

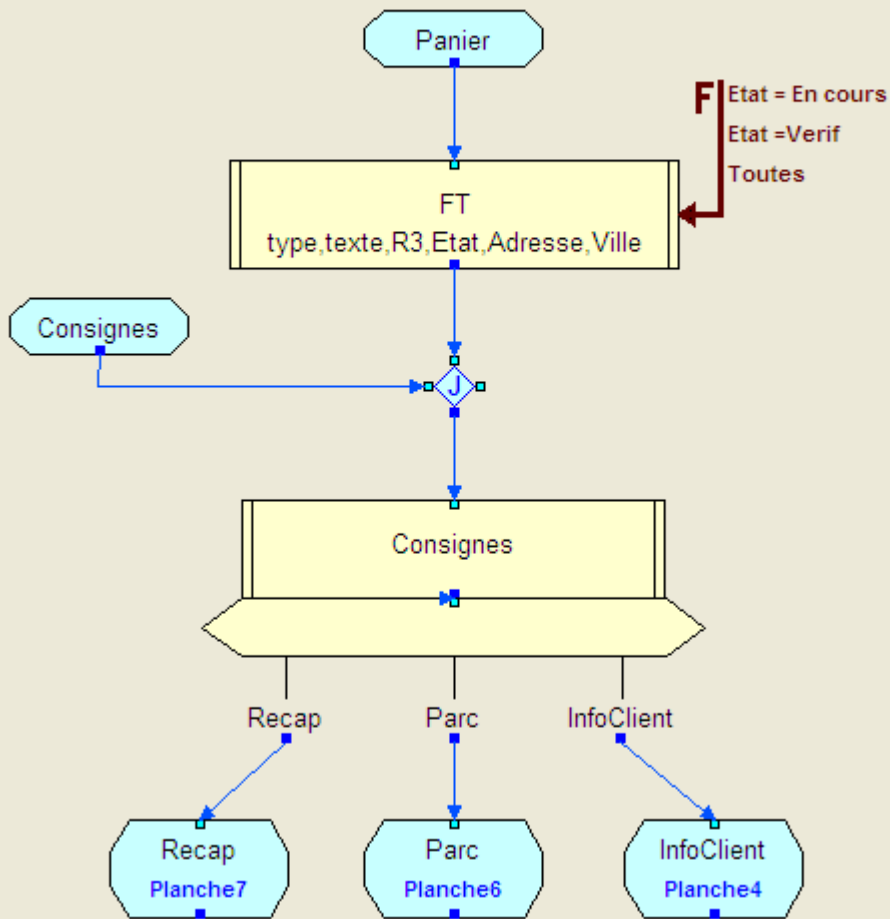


Figure 25 : Planche contenant les consignes

Quand l'intervention a été réalisée, le technicien renseigne son récapitulatif « Recap » où il saisit la description du problème, les différentes informations retournées, etc. Il demande une signature du client dans le cadre d'interventions facturées.

THESE-MACAO	Juin 2008	108 / 266
	Nicolas FERRY	

III.3.3 - Structure XMI des fichiers SNI

En auscultant les fichiers .sni des différentes planches, nous retrouvons les objets du métamodèle qui sont sauvegardés au travers de la structure XMI du fichier :

```
<?xml version="1.0" encoding="ASCII"?>
<sni:SNI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
      xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
      xmlns:sni="http://sni" backColor="0"
      titre="Administration de véhicules">
  <lstObjets xsi:type="sni:Debut" X="130.0" Y="45.0" width="90.0"
    height="30.0" label="Planche 1">
    <outputPorts name="out1" X="0.5" Y="0.96666664"
      linkPort="//@lstObjets.21/@inputPorts.0"/>
  </lstObjets>
  <lstObjets xsi:type="sni:AffListe" X="102.0" Y="146.0" width="143.0"
    height="73.0" dispListe="Véhicules"
    dispAttrib="(immatriculation, marque, modèle)">
    <inputPorts name="in1" X="0.5"/>
    <outputPorts name="out1" X="0.5" Y="0.98630136"
      linkPort="//@lstObjets.3/@inputPorts.0"/>
  </lstObjets>
  ...

```

Figure 26 : Exemple de fichier XMI d'une planche SNI

Examinons à présent comment s'agence cette structure qui transporte l'information des interfaces abstraites. Pour cela, nous allons étudier le métamodèle du Schéma Navigationnel des IHM.

C.III.4 - Structure du métamodèle du SNI

Nous venons de définir les fonctionnalités attendues pour l'éditeur VisualSNI. Celui-ci va manipuler des modèles SNI avec son formalisme. Actuellement, il existe plusieurs approches pour caractériser un langage.

Il existe les langages décrits de manière formelle et ceux décrits de manière informelle ou semi-formelle. Les langages formels utilisent une description de type axiomatique et de type algébrique comme par exemple Z, OBG, Prolog. Les langages de types opérationnels sont la représentation d'un programme sous forme d'opérations. Le programme peut être représenté sous la forme d'une machine à états. Le langage ASM ou le langage B permettent de décrire une machine à états abstraits.

Un langage est défini par sa syntaxe sous forme d'une grammaire ou d'une définition d'un métamodèle.

La sémantique dynamique décrit le comportant. Plusieurs manières permettent de décrire une sémantique dynamique : la sémantique axiomatique qui utilise des invariants, la sémantique dénotationnelle qui utilise une programmation fonctionnelle (Lambda-calcul), et la sémantique opérationnelle qui utilise une programmation impérative.

Pour le SNI, la syntaxe du langage a été définie initialement dans le livre MACAO. La représentation adoptée existait déjà mais il a fallu la concevoir pour qu'elle puisse être adapté à un système informatique. La description syntaxique est réalisée au moyen d'un métamodèle. Le métamodèle est une approche convenant pour décrire les classes du système et pour proposer une syntaxe adaptée.

L'IHM de l'application est liée à la couche qui est appelée M0 dans la représentation de l'OMG. Si nous concevons un modèle de SNI qui est une abstraction de l'enchaînement conceptuel de l'interface, celui-ci se trouve sur la couche M1 des modèles. Son métamodèle [54][55] qui définit la structure de ses éléments se trouve sur la couche M2. Enfin, son métamétamodèle qui définit le système de sérialisation adopté par le parseur de sauvegarde se trouve au niveau M3.

La plupart des structures peuvent être représentées par des métamétamodèles. Actuellement il en existe plusieurs dont deux correspondant au besoin : le MOF et ECORE. Le MOF [56] a été défini par l'organisme de l'OMG, et ECORE a été conçu pour les applications Eclipse utilisant EMF. Le SNI peut être considéré comme une machine à états d'affichage d'éléments visuels. Donc par rapport à l'OMG, le SNI est assimilable à un diagramme d'états-transitions et plus précisément à un diagramme d'activités. Comme il s'avère intéressant de réutiliser les briques de base de EMF pour gérer le métamodèle du SNI, le SNI sera modélisé suivant le métamétamodèle d'ECORE.

THESE-MACAO	Juin 2008	110 / 266
	Nicolas FERRY	

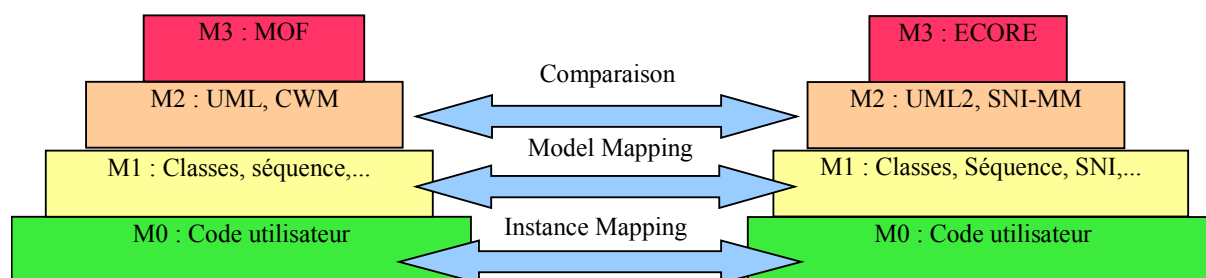


Figure 27 : Ce schéma montre les pyramides des niveaux de modélisation pour l'OMG (avec le MOF) et pour Eclipse (avec Ecore). Il a été montré que des modèles issus des deux métamétamodèles sont transposables à plusieurs niveaux.

Le métamodèle du SNI représente la structuration des classes utilisées pour représenter un modèle SNI. Un modèle SNI donne la vue d'ensemble de l'enchaînement des unités de dialogues d'une application. La sauvegarde d'un modèle SNI se fera au format XMI et reprend la structure même définie par les classes du métamodèle.

Initialement l'OMG avec XMI ne gérât pas l'information de la partie graphique des modèles UML. Ceci a posé bon nombre de problèmes pour l'interopérabilité des modèles entre les différents outils UML. Pour pallier cela, l'OMG a introduit une nouvelle norme appelée XMI-DI pour permettre l'échange de diagrammes entre outils. Seulement, cette norme n'a pas convaincu pour les raisons suivantes :

- le langage XMI-DI n'est pas assez riche pour décrire une représentation graphique d'un modèle.
- Il n'y a pas d'information sur comment faire le rendu concret en XMI-DI des noeuds et liens. Il en résulte qu'il n'est pas possible par exemple de définir un diagramme de Use Case en spécifiant si le rendu d'un cas d'utilisation est une ellipse ou un rectangle.
- XMI-DI ne contient pas d'information permettant de créer un diagramme depuis un modèle, des noeuds ou des liens abstraits.
- XMI-DI n'exprime pas les contraintes d'agencement graphique. Les diagrammes de séquences ont par exemple leur agencement défini par des contraintes exprimées au niveau de la sémantique du modèle.

Dans le cadre du SNI et suivant la norme MVC, les éléments du modèle contiennent les informations persistantes, un contrôleur est dédié à chaque objet, et une Figure est associée à chaque élément du modèle. L'affichage est assuré par le module responsable de la partie visuelle.

Le métamodèle du SNI est découpé en trois parties : la gestion du SNI, l'arborescence des unités de dialogues (partie syntaxique) et l'arborescence de routage (partie sémantique dynamique). Chaque objet du SNI hérite de principes de base : une racine élémentaire commune à tous les objets, les données graphiques, et le mécanisme des connexions. La règle de connexion impose qu'un port d'entrée ne peut être relié qu'à un port de sortie et vice versa.

Nous avons choisi de ne pas faire de schémas récursifs en positionnant la classe de base du SNI en dehors de l'arborescence définie par le patron composite. Ce patron définit les unités de base et les unités composées de manière arborescente. De plus, une unité composée peut être imbriquée dans une autre.

Du point de vue de l'IHM, une interaction se fait soit comme une entrée dans le système soit par envoi vers l'utilisateur. Les catégories d'unités de dialogue sont représentées avec leurs propriétés : Affichage, Affichage de collection, Saisie, Menu...
Les commentaires quant à eux, permettent de rajouter de l'information au système.

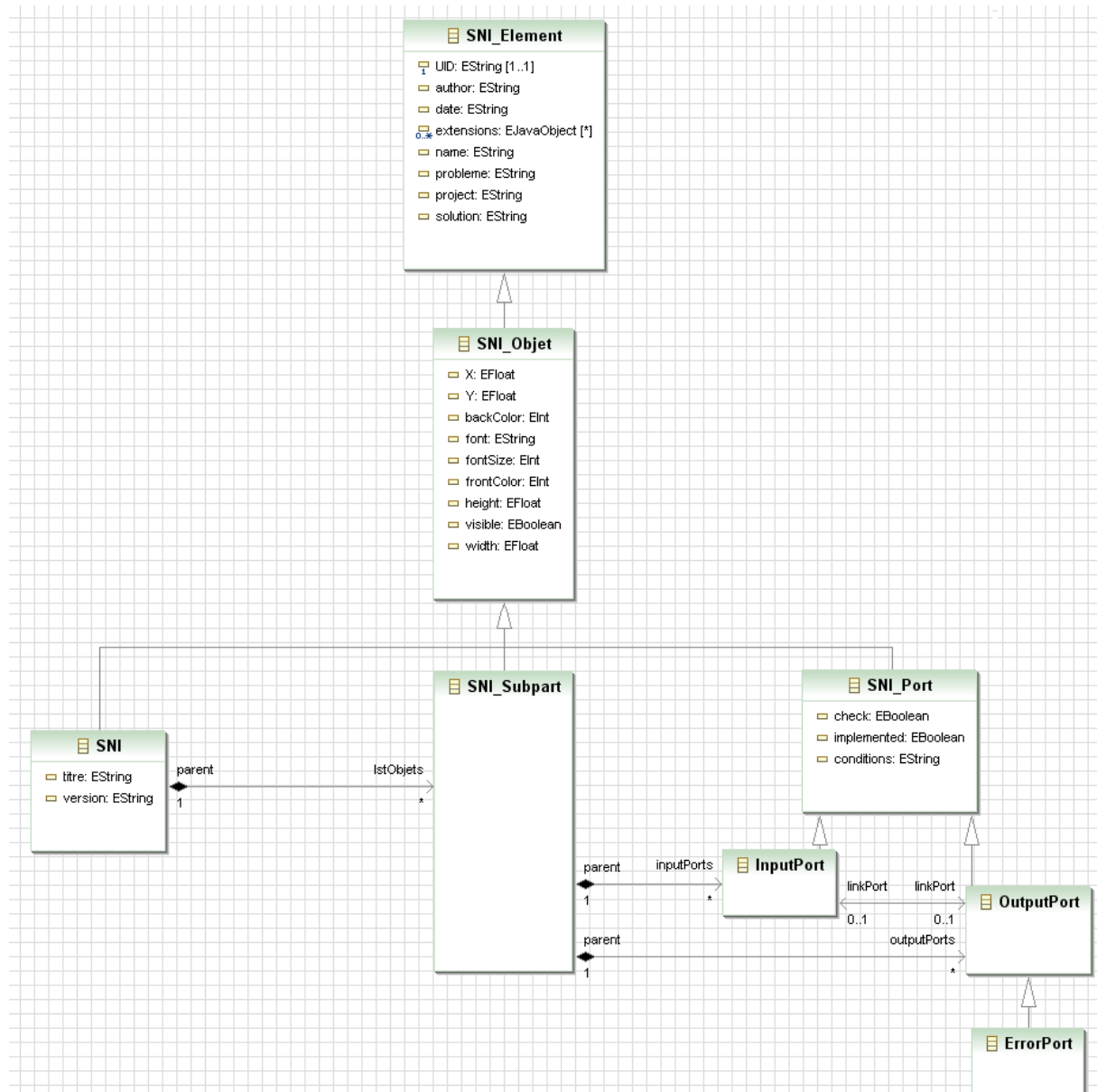
Nous appelons « Nodes » les objets qui participent à la « circuiterie » de routage de tous les objets. Ils participent aux branchements. Nous avons, pour rester homogène avec la contrainte de connexion, défini les liens comme des objets à part entière. Les jonctions et les décisions sont des branchements et débranchements conditionnels mais non commandés par l'utilisateur. Seuls les Menus représentent des choix utilisateurs.

Les étiquettes de début, d'invocation et de retour permettent de faire des connexions inter et extra diagrammes. Dès lors, l'invocation peut être utilisée soit pour factoriser des parties de SNI communes soit pour gagner en lisibilité en répartissant une IHM sur plusieurs planches.

III.4.1 - Métamodèle du SNI – Paquetage Gestion

Ce paquetage gère les données communes à chaque objet du SNI : les données élémentaires, graphiques, et le système d'interconnexion des objets utilisant des ports portant les conditions de navigation.

THESE-MACAO	Juin 2008	112 / 266
	Nicolas FERRY	



SNI_Element : est la classe hiérarchiquement la plus élevée. Elle contient les informations élémentaires dont va hériter chaque objet. Elle regroupe la réponse aux questions Quoi, Qui, Quand, Où, Pourquoi, Comment. Avec les champs :

- UID : Identifiant unique global d'un objet.
- Name : Nom local de l'objet.
- Date : Date de dernière modification de l'objet.
- Projet : Couverture prototype de l'objet dans le projet.
- Problème : Description du contexte applicatif de l'objet.
- Solution : Description de la solution
- Extension : Supporte une collection de champs d'évolution par version

SNI_Objet est la classe qui gère la représentation graphique des objets dans le cadre de l'éditeur. Concrètement, ce sont les paramètres persistants de la partie visuelle des objets d'un SNI. Nous avons défini les paramètres suivants :

- Visible : Indicateur booléen qui indique la visibilité de l'objet dans le diagramme
- X : coordonnée absolue sur l'axe des abscisses de l'objet
- Y : coordonnée absolue sur l'axe des ordonnées de l'objet
- Height : Hauteur de l'objet
- Width : Longueur de l'objet
- FrontColor : Couleur d'écriture d'un objet au format X,B,V,R stocké dans un entier 32bits (Texte & Cadre)
- BackColor : Couleur de fond d'un objet au format X,B,V,R stocké dans un entier 32bits
- Font : Nom de la police de caractères utilisée pour l'écriture des textes de l'objet.
- FontSize : Taille en entier de la police des textes principaux.

SNI_Port est la classe qui définit un port. Un port est un canal de communication vers un autre port. L'accès est restreint en fonction de conditions appelées « préconditions » pour un port d'entrée et « postconditions » pour un port de sortie. Dans le SNI, les conditions sont souvent utilisées pour restreindre les accès des utilisateurs et pour spécifier des conditions d'enchaînement dans l'IHM. Un port contient une expression conditionnelle qui constitue un ensemble de conditions booléennes.

- check : Résultat du calcul de l'expression booléenne.
- implemented : indique si la suite du SNI est implémentée dans le prototype courant.
- Cond : Condition sous forme d'une expression booléenne à évaluer.

Dans le SNI, il existe trois types de ports : les ports d'entrée appelés InputPort, les ports de sortie ou OuputPort et les ports d'erreur ou ErrorPorts.

La connexion entre objets se fait uniquement d'un port d'entrée à un port de sortie. L'accès est fait par le champ :

- linkPort : Lien par référence sur le port correspondant (ou null sinon).

SNI_Subpart est la classe responsable du mécanisme des ports et plus finement de la communication entre objets. En effet chaque objet du SNI possède une liste pour les ports d'entrée et une liste pour les ports de sortie. Les ports sont stockés de l'indice 0 à n dans la liste. Ceci n'a pas d'influence particulière sur le format de sortie, ce qui reste important est que la partie vue et la partie modèle des ports restent cohérentes à indice équivalent.

SNI est la classe d'un diagramme SNI situé dans une planche de présentation. C'est l'objet de base qui est sauvegardé. Celui-ci contient les champs suivants :

- Titre : Titre de la planche
- version : numéro de version des objets
- racine : Etiquette de début principale du SNI
- LstObjets : Liste constituant l'arborescence principale des objets du SNI

Le numéro de version indique avec quelle version du métamodèle il faut travailler pour traiter les objets d'une instance SNI.

Ce qui précède concerne la structure fondamentale de définition du métamodèle. Nous pouvons faire quelques remarques générales. Un Objet du SNI a une structure privée avec SNI_Element, une structure publique avec SNI_Objet et des interfaces avec les autres objets avec SNI_Subpart.

III.4.2 - Métamodèle du SNI – Paquetage Répartition

Les objets se répartissent en plusieurs catégories. Les UD décrivent la partie syntaxique du langage. Les Commentaires permettent d'ajouter des précisions. Les Nodes gèrent la partie sémantique dynamique du langage.

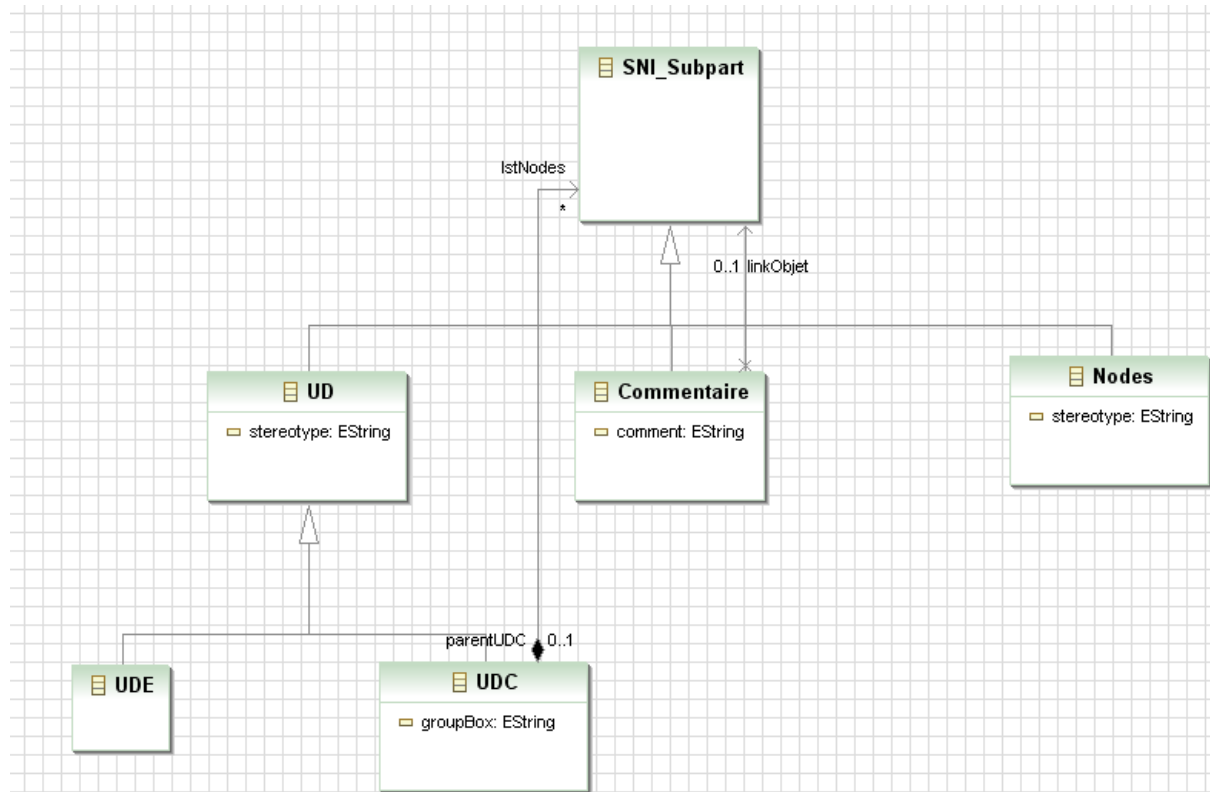


Figure 29 : Métamodèle du SNI - Paquetage Répartition

Les objets d'un SNI peuvent être soit des unités de dialogues UD, soit des noeuds de communication, soit des commentaires. Les noeuds représentent toute la « circuiterie » et branchements dépendant de choix qui ne sont pas en interaction directe avec les utilisateurs. Les unités de dialogues représentent les unités d'IHM en interaction avec l'utilisateur final.

THESE-MACAO	Juin 2008	116 / 266
	Nicolas FERRY	

Commentaire est la classe responsable des champs d'annotation. Celle-ci peut être vue comme un objet qui n'a pas de mécanisme de gestion de ports. En revanche, elle contient une description qui ne peut simplement se formaliser avec le modèle. Elle est constituée de :

- Comment : Le commentaire
- linkObjet : Le lien vers l'objet si le commentaire se rattache à un objet particulier.

THESE-MACAO	Juin 2008	117 / 266
	Nicolas FERRY	

Parmi les UDE de Sorties de type Présentation, nous avons les affichages d'objets, les affichages de collections, et les affichages de messages.

La classe AffObjet est une unité de dialogue qui affiche une information ou une classe constituant un ensemble d'informations à l'utilisateur.

- dispObjet : Nom de l'objet à afficher
- dispAttrib : Nom de ou des Attributs de l'objet affiché

La classe AffListe est une unité de dialogue qui affiche une collection d'objets de type donné.

- dispListe : Nom du type d'objet de la liste à afficher
- dispAttrib : Nom des attributs du type de l'objet
- lstFiltres : Liste des filtres applicables à cette liste.
- lstTri : Liste des Tris applicables à cette liste.

La classe AffMessage est une unité de dialogue qui affiche un message ou une alerte à l'utilisateur.

- Message : contient le message qui est affiché à l'utilisateur.

Nous avons considéré l'impression comme la présentation d'informations persistantes physiquement pour l'utilisateur. Historiquement ce nom avait été donné pour l'opération d'impression papier.

La classe ImpObjet permet de représenter cette persistance.

- impObjet : Nom de l'objet affiché persistant.
- impAttrib : Nom de ou des Attributs de l'objet affiché persistant

Pour les UDE en Entrées de l'IHM, nous trouvons les saisies et les menus.

La classe SaisieObjet représente une unité de dialogue de saisie d'information en provenance de l'utilisateur avec les champs suivants :

- recupObjet : Nom de l'objet à saisir
- recupAttrib : Nom de ou des informations sur l'objet à saisir.

La classe Menu représente un choix à faire par l'utilisateur. Un Menu contient des groupes de choix classifiés suivant une expression conditionnelle. Chaque groupe contient lui-même plusieurs options possibles pour l'utilisateur. La structure est la suivante :

- dispObjet : Nom de l'objet affiché pour le choix

Le groupe est composé des champs :

- condLabel : L'expression booléenne de la condition sous forme d'une chaîne de caractères à évaluer.

L'option :

- mustExist : Indique que l'objet de la cible doit exister pour accéder à l'UD correspondante.
- Name : Nom de l'option

III.4.4 - Métamodèle du SNI – Paquetage Nodes

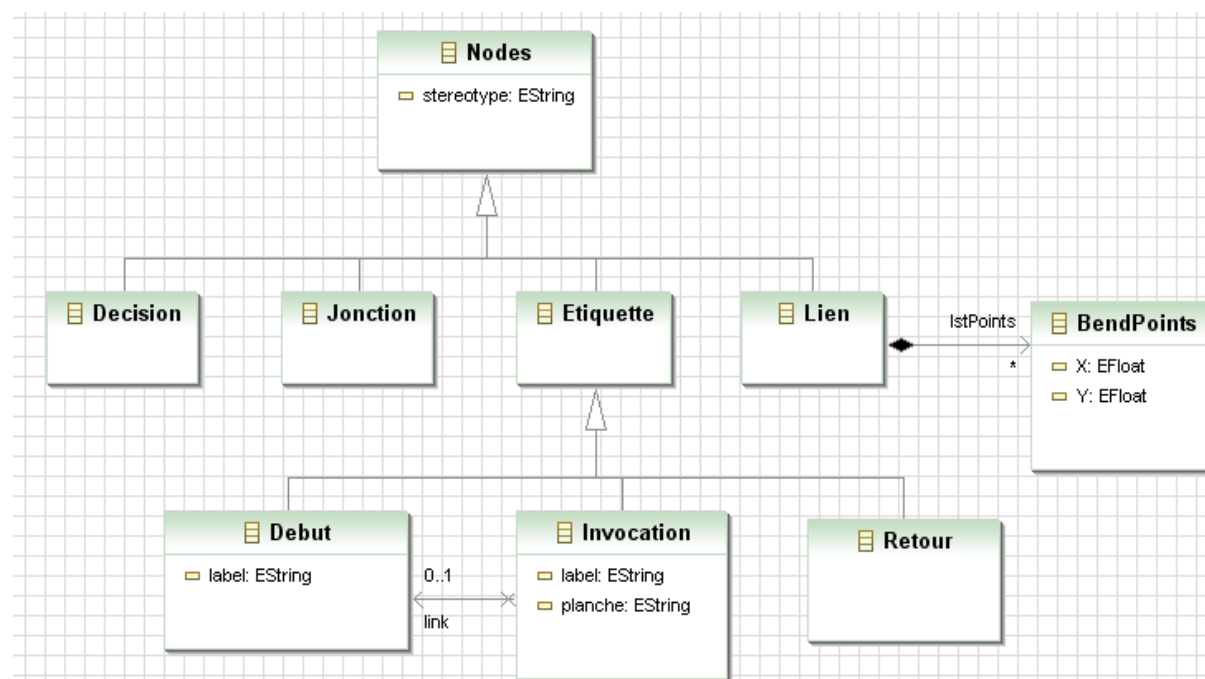


Figure 31 : Métamodèle du SNI - Paquetage Nodes
Les noeuds définissent les objets du SNI participant à la navigation. La logique d'enchaînement des affichages est gérée par l'empilement d'appels aux UD.

Les noeuds sont des enchaînements et des branchements qui ne sont pas en interaction directe avec l'utilisateur. Nous trouvons parmi eux : la Décision qui est un débranchement multiple conditionnel, la Jonction qui est un point de regroupement de plusieurs chemins possibles et les Etiquettes qui sont des débranchements permettant des sauts inter et intra SNI.

La classe Debut est un point de départ dans un diagramme. Elle peut être vue comme une arrivée depuis un autre diagramme. La classe Retour définit un retour à l'étiquette appelante.

La classe Invocation est un appel qui lie cette étiquette à une étiquette de Debut. C'est donc un saut vers une étiquette de Debut. La présence d'une étiquette de Retour conditionne le débranchement sur l'objet suivant la dernière Invocation.

Le Lien est la classe responsable des liens entre objets du SNI. Le lien comporte lui aussi des ports si bien que les ports restent la connexion bas niveau entre objets. Le lien est constitué de points de brisure pour la ligne (Bendpoints). C'est points sont stockés en cordonnées absolues.

- IstPoints : La liste des points entre les deux extrémités.

C.III.5 - Retours d'expériences en contexte réel

Nous avons cherché à évaluer le logiciel en utilisation réelle. Pour cela, VisualSNI a été présenté lors de TP d'IHM à plusieurs groupes d'étudiants. L'exercice a consisté à faire la rétro conception d'applications connues et de les modéliser avec VisualSNI.

L'éditeur graphique a aussi été présenté à Capgemini Sud aux membres de la communauté Méthodologies. Nous avons présenté VisualSNI, son rôle, ses objectifs et les fonctions majeures supportées. Cette réunion préalable vise à sensibiliser le noyau dur des architectes avant de diffuser le logiciel plus largement au groupe.

La conception de VisualSN satisfait les critères "d'utilisabilité", de robustesse et d'efficacité recommandés dans la philosophie de Constantine et Lockwood dans leur ouvrage « Software for Use » [9].

C.III.6 - Bilan

Dans ce chapitre, nous avons décrit l'éditeur graphique de SNI dans sa plate-forme. Nous avons présenté ses spécifications en définissant son rôle, ses objectifs, son architecture et ses fonctions majeures. Une maquette générale a présenté l'aspect et la disposition adoptée dans l'environnement Eclipse.

Par la suite, nous avons cherché à montrer avec un exemple pratique comment construire un SNI étape par étape en révélant les subtilités de l'éditeur. Nous avons construit une rétro conception de l'application du projet Sicli réellement développé par Capgemini.

Nous sommes également entrés dans l'architecture interne de l'outil avec la description du métamodèle du SNI, un des objectifs principaux de cette thèse. Cette description détaillée met en lumière le format d'échange XMI. Le métamodèle est par ailleurs un élément public en lecture et qui contribue à l'ouverture du format vers l'extérieur.

Enfin, nous avons donné le retour d'expériences avec les bêta-testeurs qui nous ont permis par leurs manipulations de corriger de nombreux bugs. Les prototypes les plus avancés ont été présentés à une cellule d'architectes experts de Capgemini afin de valider les intérêts et le placement de VisualSNI.

Rétrospectivement, la conception détaillée interne de l'éditeur a été très enrichissante sur le plan technique car en rapport étroit avec la plate-forme Eclipse et ses modules additionnels.

THESE-MACAO	Juin 2008	122 / 266
	Nicolas FERRY	

Chapitre IV : La conception de l'éditeur graphique de SNI

C.IV.1 - La plate-forme Eclipse comme socle

Ce chapitre est une vue plus technique et qui rentre plus en profondeur quant au choix pris sur la structure, la conception et la réalisation de l'éditeur. Comme nous l'avons vu dans la « recherche de la cible », Eclipse a été retenu comme socle à la plate-forme MACAO pour son soutien par une communauté massive open-source et son ouverture avec des modules supplémentaires.

Eclipse est souvent présenté comme un environnement de développement qui, en son coeur, intègre une plate-forme de gestions de modules ou plugins [70]. Le noyau d'Eclipse gère d'ailleurs la logistique entière des plugins. Eclipse peut ainsi être vu comme un noyau évolutif et paramétrable.

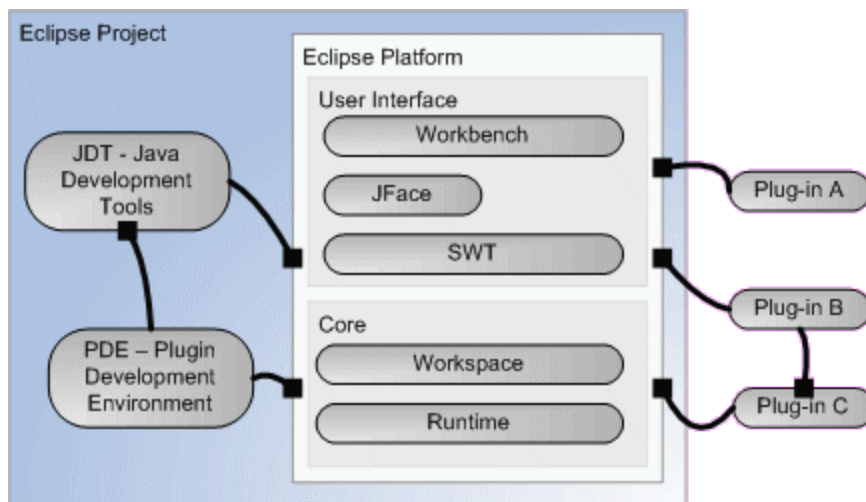


Figure 32 : Structure en modules d'Eclipse. La plate-forme est constituée uniquement de modules qui s'emboîtent les uns les autres.

Si nous souhaitons enrichir la plate-forme existante et son IDE, il faut créer un plugin. Le système de plugin Eclipse fournit une approche assez simple pour le développement de plugins avec la plate-forme PDE (Plugin Développement Environnement). VisualSNI comme tout plugin contient le fichier « plugin.xml » qui décrit l'ensemble des caractéristiques du

THESE-MACAO	Juin 2008	123 / 266
	Nicolas FERRY	

plugin ainsi que les interfaces entrantes et sortantes, les dépendances entre plugins et les points d'extensions sur lesquels le plugin se connecte pour interagir avec les plugins environnants.

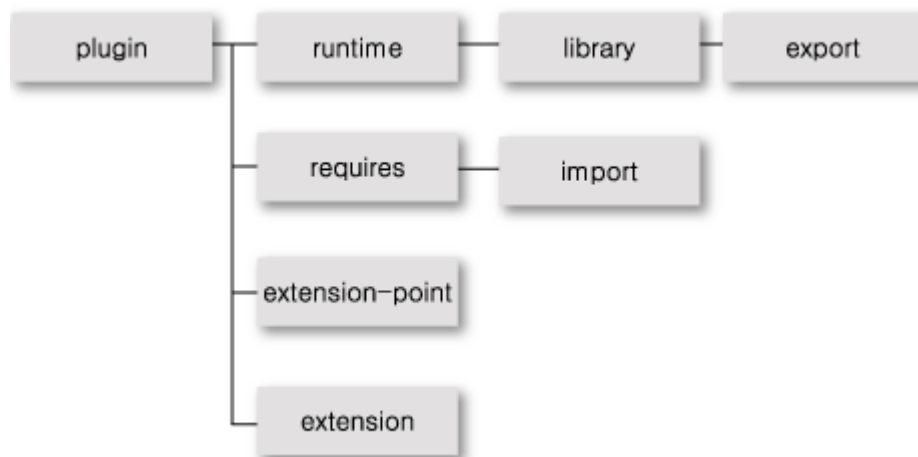


Figure 33 : Structure adoptée par tous les plugins Eclipse

C.IV.2 - Création d'un plugin pour l'éditeur

Nous avons développé un plugin spécifique pour éditer un diagramme SNI. D'autres éditeurs, comme ULC [69], sur Internet peuvent servir de modèle. Dans le cadre de notre approche, l'implémentation de MACAO correspond à l'ensemble de la plate-forme traitant de l'outil de génie logiciel pour la méthode MACAO. Ce plugin contribue au développement logiciel de la méthode sur l'axe d'IHM. Par conséquent et pour suivre le standard de nommage utilisé dans le monde Eclipse, nous l'avons appelé : **org.macao.ihm.sni**

A titre d'exemple, l'éditeur de diagramme SEF prendra le nom de : **org.macao.ihm.sef**

Ce module doit être capable de gérer l'édition graphique de SNI comme décrit fonctionnellement précédemment.

Concrètement, l'éditeur de SNI est un `GraphicalEditorWithFlyoutPalette` qui correspond à une vue au sens Eclipse composée d'une zone pour la palette des outils d'édition et d'une zone pour l'édition. La gestion de l'éditeur se compose de plusieurs vues :

- la vue graphique (la zone d'édition)
- la vue de la palette (les outils de création)
- la vue globale pour la navigation rapide

THESE-MACAO	Juin 2008	124 / 266
	Nicolas FERRY	

- la vue des propriétés

Il définit aussi entre autre :

- un domaine d'édition global
- un gestionnaire des événements de touches
- un gestionnaire de suivi des ressources
- le gestionnaire du modèle
- le SNI édité

L'utilisateur agit sur le système et envoie ses ordres au travers d'actions. Eclipse gère les actions de l'utilisateur et référence l'ensemble des actions dans un registre global qui les recense ainsi que leur identifiant. Dans un éditeur de base, il existe trois sortes d'actions : une liste pour les actions référençant un objet de l'éditeur (editpartActions), une liste pour les actions opératives empilées (stackAction), une liste pour les actions au niveau de l'éditeur (editorAction).

L'éditeur implémente un mécanisme de pile pour la gestion des commandes. Les actions sont créées à partir des manipulations de l'utilisateur et sont empilées. Toutes actions référencées peuvent être ainsi défaisables et refaisables à loisir. L'éditeur doit à son initialisation, créer et définir ses actions et les référencer correctement dans le registre.

L'éditeur gère les fonctions presse-papier copier/coller par l'intermédiaire du clipboard du système. Celui-ci permet de réaliser des copies entre différents éditeurs. En effet, le module GEF présentait un bug ou plutôt un manque car il ne nous a pas permis de coller en dehors du domaine d'édition. Le problème a été corrigé en passant par le clipboard de plus haut niveau.

Une méthode introspective getAdapter permet à Eclipse de retrouver un contrôleur donné.

Comme tout éditeur les fonctions de sauvegarde et de chargement synchronisent les accès et fixe l'état du modèle. La présence dans la pile de commandes exécutées indique l'état modifié du fichier. Eclipse montre un fichier en cours de modification en rajoutant une petite * après le nom du fichier. La sauvegarde met à jour le fichier ressource SNI et met à jour le point de marquage dans la pile des commandes. Le fichier SNI est associé au chargeur XML pour interpréter le format standard XMI du métamodèle. En cas d'erreur, l'exception décrit l'erreur et la ligne dans le fichier causant la défaillance.

Pour ce qui est des vues : la vue de la palette a été découpée en quatre grands groupes :

- Le pointeur à défaut (sélection) et les connexions
- Les unités de dialogues

THESE-MACAO	Juin 2008	125 / 266
	Nicolas FERRY	

- Les unités de routage
- les unités diverses

Pour la vue graphique, c'est un « ScrollingGraphicalViewer » qui a été utilisé. Il s'agit du composant classique qui gère une vue avec les fonctionnalités de déplacement vertical et horizontal. Ensuite le viewer sera relié avec l'arborescence des objets par un pont appelé le « RootEditPart ». Dans notre plugin, nous avons choisi un ScalableFreeFormRootEditPart. Cela signifie que la surface du composant sera extensible dans toutes les directions y compris avec des coordonnées négatives. Le contrôleur EditPart du diagramme du SNI sera la racine de l'arborescence du graphe.

Mais ceci nous amène à définir un « EditPart » :

Dans Eclipse la notion de « EditPart » est employée pour gérer des objets sélectionnables et dont les propriétés sont éditables. Ce terme est repris dans GEF pour définir le « Contrôleur » d'un objet dans le pattern MVC. Ainsi le pont RootEditPart reliera le viewer avec son contenu. Plus tard, chaque objet créé imposera la création de son contrôleur qui sera ajouté à l'arborescence des EditPart. La création d'un EditPart est du rôle de l'usine à EditPart qui est paramétrée pour le viewer.

Le viewer gère aussi notamment le menu contextuel et l'intercepteur des touches clavier. Enfin, le viewer est rattaché à son domaine d'édition car par un système d'onglets il est possible de présenter plusieurs autres éditeurs pour un domaine d'édition d'une vue particulière.

Cette description définit les mécanismes du fonctionnement de base de l'éditeur. Celle-ci est plus ou moins commune à de nombreux éditeurs. Maintenant attardons nous sur la structure avancée.

C.IV.3 - Structure adoptée

Le plugin a été structuré avec une architecture de type MVC (Modèle-Vue-Contrôleur). Cette architecture de base, inventée par Trygve Reenskaug en 1979 qui travaillait sur Smalltalk dans les laboratoires de Xerox PARC, convient bien à notre cadre pour plusieurs raisons :

Tout d'abord nous avons besoin de dissocier les trois composantes dans la mesure où l'utilisateur a la possibilité de combiner plusieurs vues (au sens Eclipse) d'un même diagramme. Nous avons inventorié d'ores et déjà l'éditeur principal et sa palette, sa vue globale [159], et les propriétés d'un objet graphique. Ainsi nous avons plusieurs vues et donc autant de couches de présentation associée. La modification dans une vue d'un élément du système doit être mise à jour dans les autres vues et inversement. Le contrôleur définit un

THESE-MACAO	Juin 2008	126 / 266
	Nicolas FERRY	

comportement particulier et les traitements correspondant à la vue associée. Enfin, le modèle est lui unique. Ce découpage est similaire aux notions de graphes combinés utilisés par Stéphane Huot et Al. [137].

Nous avons décidé de relier à un élément dans la couche du modèle, un contrôleur unique. Un cas particulier concerne la vue des propriétés puisqu'elle utilise un contrôleur très simple auto généré par EMF.

Comme cette structure est souvent utilisée en programmation Eclipse [65] de nature multi-vues, elle a l'avantage d'être disponible avec des outils de bases sous forme de modules standards. Nous parlerons des plugins suivants :

- la partie Modèle est faite avec EMF pour gérer le métamodèle.
- la partie Contrôleur est faite avec GEF pour la gestion des EditPart graphiques.
- la partie Vue est faite avec Draw2D pour le dessin sur canevas graphiques.

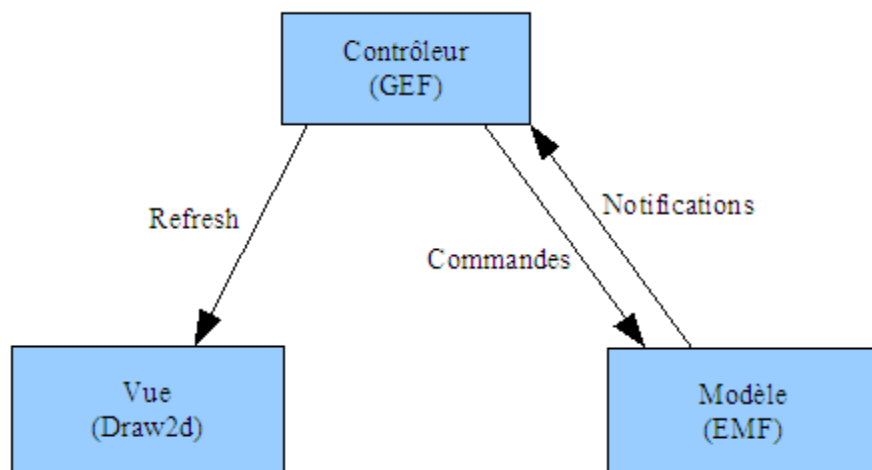


Figure 34 : Le pattern MVC a été repris dans une version légèrement modifiée. Toutes les actions sont envoyées au contrôleur qui traite en émettant des commandes. Les commandes modifient le modèle et elles sont annulables. Toutes modifications explicites ou implicites d'objets notifient leurs contrôleurs. Si bien que le contrôleur peut demander un rafraîchissement des vues concernées. Dans cette version la vue ne lit pas directement les données. Elle met en cache les paramètres du modèle fourni par le contrôleur.

L'éditeur graphique doit gérer le métamodèle de la sémantique définie par le SNI. Celui-ci sera créé et géré avec le plugin EMF (Eclipse Modeling Framework) qui s'occupera de la partie Model. Par abus de langage, nous appellerons Model de l'éditeur dans le cadre de MVC, le métamodèle définissant la grammaire du SNI.

THESE-MACAO	Juin 2008	127 / 266
	Nicolas FERRY	

C.IV.4 - Description du Model avec EMF

L'élaboration du métamodèle nous a permis de concevoir l'ensemble des classes du modèle. Il est lui-même défini en adoptant le métamétamodèle ECORE qui est utilisé dans Eclipse et les plugins utilisant EMF.

A ce jour, OMG définit son propre métamétamodèle appelé MOF pour définir les métamodèles des diagrammes UML. La description est faite par une pyramide qui représente les 4 niveaux de couches. Du code « réel » au niveau M3 jusqu'au métamétamodèle MOF en M0.

Sous Eclipse nous allons utiliser EMF [31] qui utilise le métamétamodèle ECORE pour décrire les éléments d'un diagramme SNI. Celui-ci est défini autour d'éléments hiérarchiquement simples.

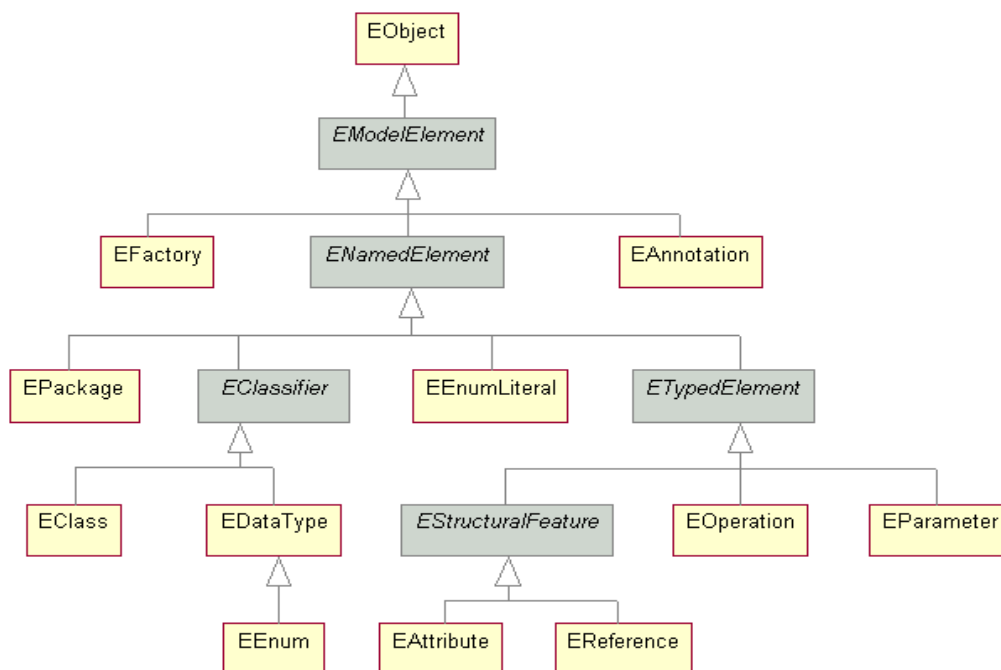


Figure 35 : Hiérarchie des éléments du métamétamodèle ECore

THESE-MACAO	Juin 2008	128 / 266
	Nicolas FERRY	

The diagram illustrates the MVC (Model-View-Controller) architecture for a vehicle management application. It is divided into three main sections:

- Classes Java:** On the left, a stack of documents represents the Java classes. An arrow points from this stack into the central application box.
- Central Application Box:** A large light blue rectangle containing three yellow boxes:
 - EMF (Modèle):** The top box, representing the data model.
 - GEF (Contrôleur):** The bottom-left box, representing the controller.
 - Draw2D (Vue):** The bottom-right box, representing the view.Arrows indicate the flow of control and data within the application:
 - A double-headed arrow connects **EMF (Modèle)** and **GEF (Contrôleur)**, labeled **Traitement des événements** (Event Processing).
 - A single arrow points from **GEF (Contrôleur)** to **Draw2D (Vue)**.
- External Data and Views:**
 - Fichiers.SNI:** A cylinder icon labeled **XMI** is connected to the **EMF (Modèle)** box by a thick arrow pointing from the model to the files.
 - SNI:** A yellow box labeled **SNI** is connected to the **Draw2D (Vue)** box by a thick arrow pointing from the view to the SNI box. Below the SNI box, three document icons represent different views: **Nouveau** (New), **Imprimer** (Print), and **Détail** (Detail). Below these icons, the text **Véhicules mis en vente** (Vehicles for sale) is displayed.

THESE-MACAO	Juin 2008	130 / 266
	Nicolas FERRY	

C.IV.5 - Le contrôleur avec GEF

Le plugin GEF [32] pour Graphical Editor Framework permet de concevoir un éditeur graphique sous Eclipse adoptant le patron de conception MVC.

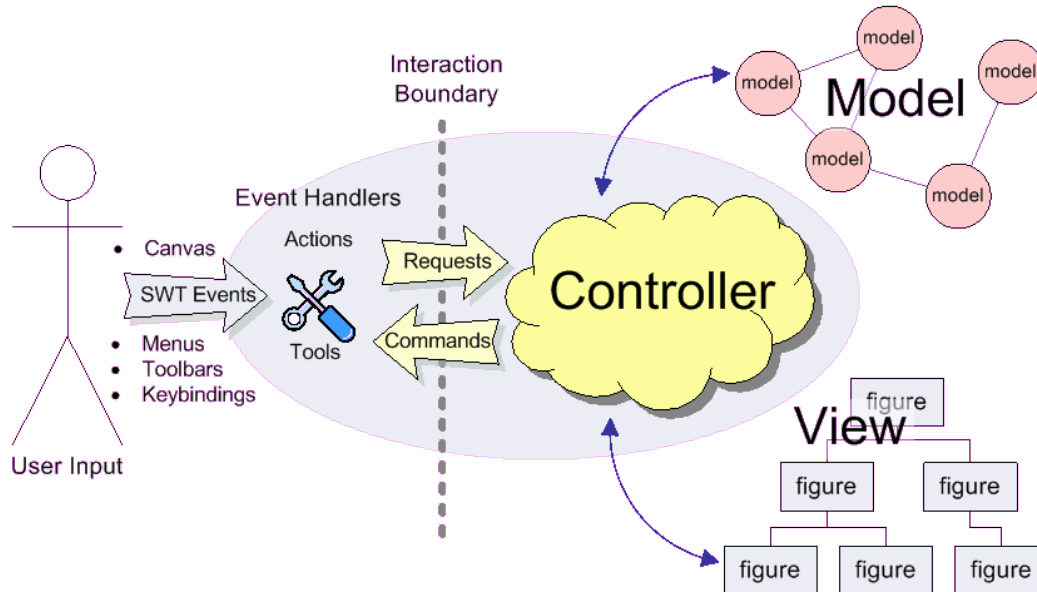


Figure 39 : Description générale de MVC faite par GEF (Graphical Editor Framework), l'utilisateur manipule des outils pour réaliser des actions. Les actions émettent des requêtes au contrôleur concerné pour mettre à jour le modèle et rafraîchir la vue.

L'approche de GEF [68] est de fournir les traitements les plus couramment utilisés pour la création d'éditeurs graphiques. Son API [67] fournit les principaux concepts nécessaires :

- Création d'objet, déplacement d'objet, changement de taille, suppression d'objet
- Connections de liens, gestion de points de brisure pour le routage des liens.
- Routeur avec algorithme du plus court chemin, Manathan ou autres
- Annulation et relance de commandes (undo/redo)
- Edition directe de libellés sur le graphique
- Vue globale et Vue zoom – panoramique
- Palette d'outils
- Règles et guides horizontaux/verticaux
- Accrochage d'un objet sur la grille
- Interception des touches claviers
- Drag & Drop

THESE-MACAO	Juin 2008	131 / 266
	Nicolas FERRY	

Dans GEF, le contrôleur (appelé Editpart) a un rôle central puisqu'il gère un élément du graphe et son interaction avec le système. Son rôle est de créer et maintenir la vue graphique de la figure, de créer et gérer l'arborescence de ses contrôleurs enfants (pour les UD composées d'autres UD) ainsi que les contrôleurs des liens qui en dépendent. Et enfin il doit supporter les mises à jour et notifications avec le modèle. Mais vu le nombre de requêtes qu'il est nécessaire de gérer, celui-ci délègue avantageusement les traitements à des politiques d'édition (EditPolicies) dédiés à des requêtes précises.

Pour cela, le contrôleur installe ses politiques d'édition auxquelles il va s'abonner. Chaque politique d'édition est en charge d'intercepter ses types de requêtes, de les accepter ou non en vue de réaliser le traitement correspondant. Il existe plusieurs politiques de bases fournis avec GEF qu'il est possible d'enrichir voir de compléter en les dérivant. Certaines requêtes peuvent être traitées par plusieurs politiques d'édition. Si nous reprenons l'image de l'engrenage, les politiques d'édition sont les petites roues de l'engrenage. Les politiques d'édition interceptent les requêtes destinées à leur contrôleur pour créer les commandes de modification du modèle qui seront préalablement empilées avant d'être exécutées ou annulées.

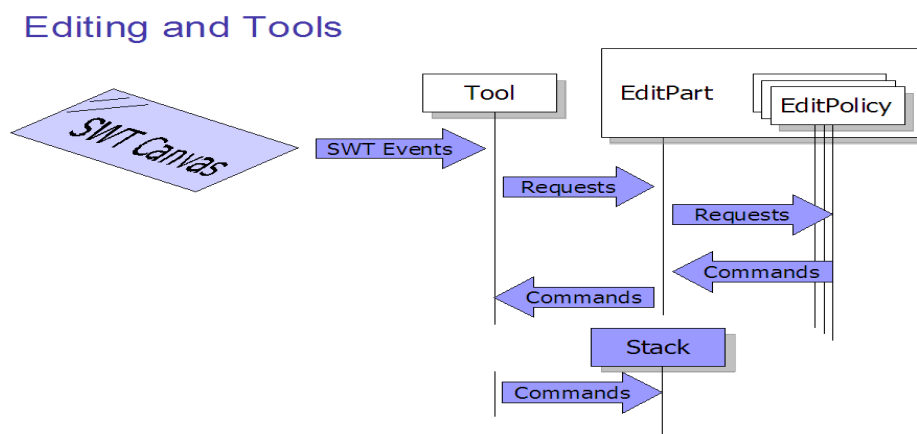


Figure 40 : Figure montrant le cheminement des requêtes. Les événements de la souris sont interprétés par l'outil en cours d'utilisation. Celui-ci crée les requêtes de façon adéquate et les transmet aux contrôleurs sélectionnés qui lui même les transmet à ses EditPolicies. Les EditPolicies créent les commandes qui seront empilées pour exécution. Les requêtes proviennent des outils alors que les commandes ont pour but de modifier le Modèle.

Voici la liste qui associe les outils, les requêtes, les actions, et les politiques d'édition :

THESE-MACAO	Juin 2008	132 / 266
	Nicolas FERRY	

Creation

Tools	Requests	Edit Policies and Roles	Actions
CreationTool	REQ_CREATE Create	CONTAINER_ROLE LAYOUT_ROLE TREE_CONTAINER_ROLE	CopyTemplateAction PasteTemplateAction
TemplateTransferDropTargetListener TemplateTransferDragSourceListener		ContainerEditPolicy LayoutEditPolicy	

Basic Model Operations (Delete)

Tools	Requests	Edit Policies and Roles	Actions
	REQ_DELETE	COMPONENT_ROLE CONNECTION_ROLE RootComponentEditPolicy	DeleteAction

Selection

Tools	Requests	Edit Policies and Roles	Actions
SelectionTool	SelectionRequest	SelectionEditPolicy	SelectAllAction
MarqueeTool	DirectEditRequest	DirectEditPolicy	
SelectEditPartTracker	REQ_SELECTION_HOVER REQ_OPEN	SELECTION_FEEDBACK_ROLE	
*GraphicalViewerKeyHandler	REQ_DIRECT_EDIT		

Moving and Resizing

Tools	Requests	Edit Policies and Roles	Actions
DragEditPartsTracker	ChangeBoundsRequest	LayoutEditPolicy	AlignmentAction
ResizeTracker	AlignmentRequest REQ_MOVE REQ_CLONE REQ_ADD REQ_ALIGN REQ_ORPHAN REQ_RESIZE	ResizableEditPolicy ContainerEditPolicy	MatchSizeAction

Connection Creation

Tools	Requests	Edit Policies and Roles	Actions
ConnectionCreationTool	CreateConnectionRequest	GraphicalNodeEditPolicy	
ConnectionDragCreationTool	REQ_CONNECTION_START REQ_CONNECTION_END	NODE_ROLE	

Editing Connections

Tools	Requests	Edit Policies and Roles	Actions
ConnectionEndpointTracker	ReconnectRequest REQ_RECONNECT_SOURCE REQ_RECONNECT_TARGET	ConnectionEndpointEditPolicy ENDPOINT_ROLE GraphicalNodeEditPolicy	
		NODE_ROLE	

Bending Connections

Tools	Requests	Edit Policies and Roles	Actions
ConnectionBendpointTracker	BendpointRequest REQ_MOVE_BENDPOINT REQ_CREATE_BENDPOINT	BendpointEditPolicy CONNECTION_BENDPOINTS_ROLE	

Bien évidemment de nouvelles requêtes ont été créées pour gérer les commandes spécifiques d'un SNI comme par exemple les ajouts / suppressions de cases de menu et d'options. Pour cela, nous avons créé un système qui relie directement une action typique du SNI en une requête SNI.

Action	Requête	EditPolicy
ActionSNI	--> REQ_SNI -->	Libre

Le travail principal de GEF est de gérer l'arborescence des contrôleurs et leurs enfants. Pour le SNI, il existe un contrôleur par unité de dialogue (UD). Mais un contrôleur peut couvrir plusieurs éléments dans le modèle. Notamment un menu est composé de groupes eux-mêmes pouvant être composés de plusieurs options. Dans ce cas, un contrôleur unique gère l'unité entière avec tous les objets la composant. L'avantage de cette méthode c'est le gain de place mémoire et la cohérence du point de vue de GEF qui travaille avec la même granularité. La contrepartie est qu'il faut programmer explicitement des comportements avec les composants internes. De manière globale un contrôleur est créé dans son conteneur parent et ainsi de suite, ceci crée l'arborescence des contrôleurs et par corrélation celle du fichier XMI.

C.IV.6 - Vue (Draw2D)

Le système graphique utilisé avec Eclipse repose sur les Widgets SWT [125]. Pour dessiner dessus, nous utilisons un "canevas" de composants légers LWS qui permet de gérer des composants graphiques légers c'est-à-dire qui ne sont pas des composants fenêtres ou assimilés. Les composants graphiques légers simplifient le système d'affichage.

Par dessus l'API de dessin Draw2D permet de dessiner des primitives simples comme des Rectangles, cercles, lignes, appelés Figures. Les Figures sont gérées elles aussi avec une arborescence.

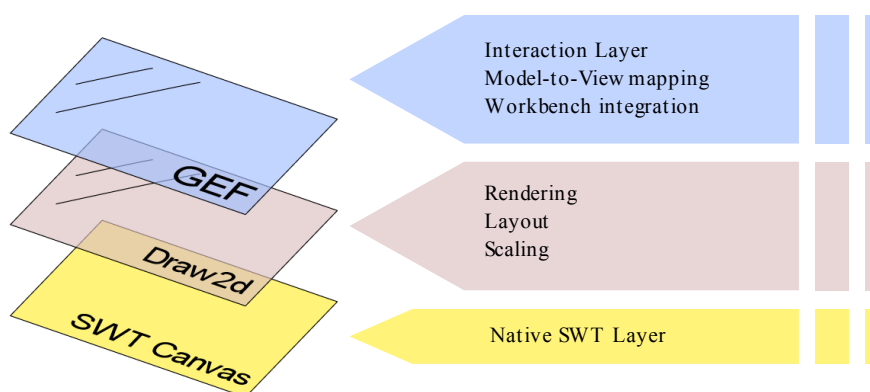


Figure 41 : Figure des couches des API graphiques

THESE-MACAO	Juin 2008	134 / 266
	Nicolas FERRY	

Le système d'affichage incorpore les fonctionnalités classiques :
paint(), repaint(), invalidate(), validate(),...

Dans Eclipse un thread dédié à l'affichage parcourt une file d'objets "Runnable" dont la fonction run() exécute le code de la procédure d'affichage de la figure concernée.

Les affichages d'UD ont été conçus comme spécifié dans le livre de la méthode MACAO. Cependant le système d'affichage a une contrainte dans notre cas bien précise : il « clippe » c'est-à-dire qu'il coupe systématiquement toutes les figures enfants aux bords de la figure du parent. Ceci a posé quelques problèmes en ce qui concerne les UD Menu et Collections. En effet, elles nécessitent l'affichage d'objets externes à la figure de base. Pour une UD de Menu par exemple les options doivent pouvoir être affichées plus bas. Pour les Collections, les filtres et Tris sont représentés en dehors sur les côtés de la figure.

Pour y pallier, il a fallu créer une figure de base non visible enveloppant toutes les figures d'une UD. Ainsi, les options, les textes, les ports sont affichés correctement en dehors de l'UD Figure. Dans l'exemple ci après, nous utilisons le mode debug pour montrer en jaune la forme de l'unité, en bleu la couleur de la figure de base, en rouge la figure porteuse de tous les éléments.

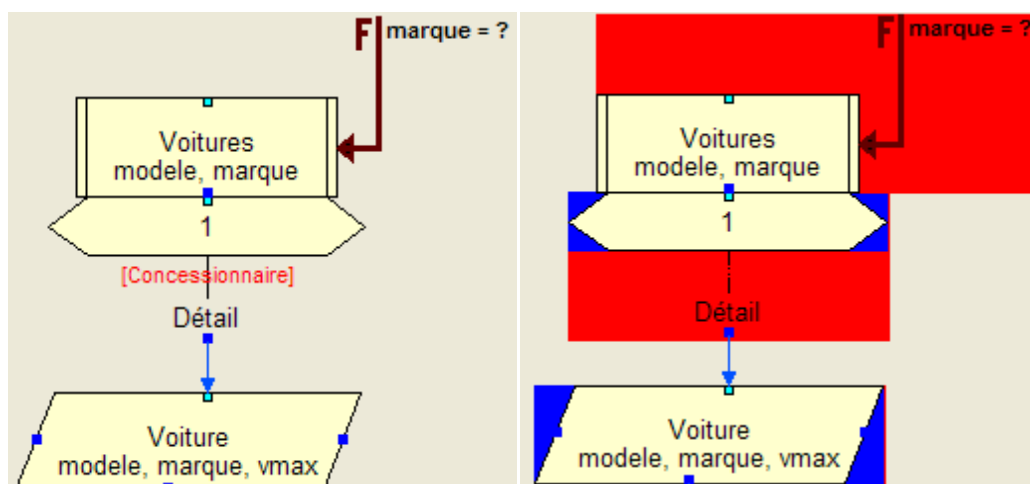


Figure 42 : Figure montrant la structuration graphique des figures du SNI.
A gauche un SNI en mode exploitation, et à droite un SNI en mode debug.

C.IV.7 - Bilan

Dans ce chapitre, nous avons vu que la plate-forme Eclipse se compose d'un noyau qui gère la connexion de modules additionnels. Chaque module s'interconnecte avec la plate-forme sur des points d'extensions et vient enrichir ou utiliser à son tour les fonctionnalités du système. Ces contributions peuvent intervenir aux niveaux du Workbench, de l'interface graphique, etc.

Pour créer un module ou un plugin la plate-forme Eclipse fournit un module qui est dédié à cette tâche. Un module contenant un éditeur graphique basique a tout d'abord été développé. Par la suite il a été ajouté au prototype les différentes vues avec la zone de dessin, la palette d'outils, la navigation rapide et les propriétés. Dans la vue de l'éditeur, nous avons développé les fonctionnalités liées à Eclipse, notamment le mécanisme de pile de commandes qui permet de faire et défaire les actions de l'utilisateur. Une fois l'ensemble des mécanismes fonctionnels de l'éditeur en place, nous avons pu intégrer les paramètres spécifiques au SNI.

Nous avons adopté une structure de type Modèle-Vue-Contrôleur pour l'éditeur graphique. Nous avons présenté dans le chapitre précédent le métamodèle du SNI, celui-ci correspond à la couche Modèle dans cette structure. Ensuite, nous précisons la partie Contrôleur gérée par le module GEF qui va intercepter les actions pour les transformer en commandes qui effectueront des traitements interagissant pour modifier ou non le modèle. Lorsque le modèle est modifié, il prévient son contrôleur en le notifiant des modifications impactées. A son tour, le contrôleur peut alors rafraîchir la ou les vues qui sont la représentation graphique du modèle. Le dessin des Vues est réalisé par le module Draw2d. Ainsi a été mise en place la structure MVC pour l'éditeur graphique.

Nous avons détaillé l'architecture et les mécanismes MVC de VisualSNI. Le prototype a concilié le système de types de commandes de GEF et de EMF qui ne sont pas forcément compatibles. Le système de notification de EMF étant simple et performant, il a été gardé sur la partie Modèle, et GEF a été limité aux contrôleurs et Draw2d pour la partie affichage.

THESE-MACAO	Juin 2008	136 / 266
	Nicolas FERRY	

Chapitre V : Le générateur de maquette JSF

Nous avons présenté l'éditeur graphique de SNI qui permet de créer une modélisation de l'interface homme-machine. Dans la volonté de laisser à Capgemini un outil de génération qui apporte des éléments concrets aux étapes du processus de réalisation, il a été choisi, en partenariat avec l'entreprise, de créer un générateur de maquettes. Le but est de couvrir l'axe IHM dans son ensemble du conceptuel au physique.

Pour reprendre l'hypothèse de départ sur la possibilité de décomposer la conception de l'axe des IHM, nous avons étudié les systèmes par raffinements successifs de modèles. Le langage B a été appliqué au domaine des IHM [146], il définit lui aussi un certain nombre de machines qui vont être raffinées pour obtenir la machine finale. D'un point de vue des modèles, les travaux sur les PMI abordés par l'équipe de Jean-Claude Tarby ont démontré qu'il est possible de construire des métamodèles par raffinements successifs. Ici, nous appliquons ce procédé au domaine de l'IHM car nous soutenons qu'il est possible de modéliser les interfaces de manière abstraite puis de descendre au niveau logique pour rajouter le typage des unités logiques et enfin le niveau physique avec l'ajout de propriétés visuelles pour les composants graphiques. L'unité de dialogue est transformée progressivement en unité logique et enfin en composant graphique. Ce passage d'un niveau à un autre est assuré par des transformations de modèles qui permettent de s'affranchir de la distance sémantiques entre les métamodèles. Un couplage fort entraînerait une certaine rigidité mais une efficacité accrue alors qu'un couplage faible laisserait plus de souplesse [24].

C.V.1 - Les transformations de modèles

Pour passer d'un modèle A à un modèle B et inversement, le mécanisme de transformation de modèles est utilisé. Celui-ci a émergé avec le domaine de l'ingénierie Des Modèles ou Model Driven Engineering en Anglais. L'architecture dirigée par les modèles ou MDA [13][20] (pour l'Anglais Model Driven Architecture) [49] est une démarche de réalisation de logiciel [59], proposée et soutenue par l'OMG c'est une variante particulière de l'ingénierie dirigée par les modèles (IDM, ou MDE [34][4] pour l'Anglais *Model Driven Engineering*). D'autres variantes de l'IDM ont été développées, par exemple par Microsoft avec les DSL (Domain Specific Language).

L'idée fondamentale est que les fonctionnalités du système à développer sont définies dans un modèle indépendant de la plate-forme d'implémentation (*Platform Independent Model*, PIM), en utilisant un langage de spécifications approprié, puis traduites dans un ou plusieurs

THESE-MACAO	Juin 2008	137 / 266
	Nicolas FERRY	

modèles spécifiques à la plate-forme (*Platform Specific Model*, PSM) pour l'implémentation concrète du système.

La traduction du PIM vers les PSM est normalement effectuée à l'aide d'outils automatisés, par exemple des transformations de modèles réalisées avec des outils comme VIATRA, MOLA [47] ou ATL [15]. Ces langages de transformation sont plus ou moins compatibles avec le standard de l'OMG nommé QVT (Query View Transformation) [30][38][45][57][64]. Le passage du PSM à la génération du code est la suite logique de ce traitement [23]. Elle peut être réalisée par des générateurs tels que Acceleo afin de produire tout type de cibles technologiques.

Les travaux actuels autour du MDA [17] tendent à renforcer les prérogatives des modèles et des métamodèles, avec l'utilisation de métadonnées et de méthodologies [36][39][130].

La transformation de modèle est un principe important [40]. La plupart des modèles qui apparaissent pour la modélisation sont cohérents et définissent leur syntaxe et leur grammaire interne. Un modèle est plus ou moins adapté à décrire une problématique et permet de couvrir des besoins différents. L'intérêt de la transformation de modèles réside dans le mécanisme qui va permettre fusionner des modèles de domaines différents. Elle permet de tisser des passerelles et lier des domaines conjoints du génie logiciel entre eux. Chaque modèle aura son importance, le choix d'utiliser un modèle dépendra de son adéquation et de sa perspicacité à décrire une partie du système.

Par exemple, il est envisageable d'écrire des transformations d'un SNI vers un diagramme de classe, d'un SNI vers un diagramme d'activité et inversement. A terme, il serait envisageable que plusieurs modèles de domaines différents participent à la génération finale d'une application comme cela est le cas manuellement.

Les transformations de modèles peuvent s'appliquer pour faire évoluer un diagramme vers un niveau d'abstraction différent mais aussi entre domaines applicatifs [41]. Ce qui place chaque modèle avec un niveau d'autonomie suffisant pour être utilisé individuellement ou dans un cadre collectif.

Il est important de noter que pour établir une structure basée sur un ensemble de modèles, il faut au début créer les transformations entre les modèles. Cette tâche nécessitera un travail en entreprise dédié à ce rôle qui pourrait être appelé architecte MDE.

THESE-MACAO	Juin 2008	138 / 266
	Nicolas FERRY	

C.V.2 - Création d'un générateur avec oAW sur Eclipse

V.2.1 - Description du projet

Lors de la phase d'acquisition des besoins, Capgemini utilise des ateliers participatifs avec les clients et les utilisateurs pour préciser leurs exigences. Au terme des ateliers, des maquettes leur sont présentées comprenant les écrans ou les pages à produire ainsi que la navigation entre ces écrans ou pages. Mais faute de temps seuls les écrans finaux sont présentés alors que souvent les problèmes de compréhension proviennent de l'enchaînement et de la navigation entre ces écrans.

Dans la plate-forme de génération de l'interface homme-machine d'une application, VisualSNI-Generator est le module transformateur qui permet de traduire un modèle SNI en une maquette d'IHM.

V.2.2 - Objectifs

Ce projet vise à outiller la génération des maquettes à partir des diagrammes SNI modélisant la navigation dans l'IHM. Il permet de spécifier les écrans majeurs ainsi que leur enchaînement. Ainsi, le générateur produit une maquette des écrans de l'application suivant l'aspect général choisi. Cette maquette peut être présentée dans les ateliers participatifs afin de préciser la solution à obtenir.

L'objectif est donc de créer un générateur dans une cible technologique existante. Le projet INRA a été choisi comme base pour la génération de son IHM.

Le Centre National de Ressources Génomiques Végétal (CNRGV) est un centre unique en France créé par l'INRA, il est chargé de conserver et de « faire vivre » les ressources génomiques végétales générées par la recherche.

Capgemini a développé pour l'INRA le système d'information du centre (SI-CNRGV). Nous nous sommes inspirés des écrans et d'un cheminement de l'application pour produire un modèle SNI représentant cet enchaînement de manière abstraite. Ensuite le générateur a été construit afin de générer des écrans d'IHM semblables à l'application mais générés suivant le modèle SNI donné.

Le générateur a pour but de construire une application Web (appelée WebApp) qui intègre les pages Web au standard JSF (Java Server Faces) et qui permet la navigation dans la maquette entre les différentes pages. Le choix de la génération au format JSF a été fixé par Capgemini.

Hypothèse simplificatrice : le générateur traitera dans un premier temps uniquement les unités de dialogues élémentaires du SNI.

THESE-MACAO	Juin 2008	139 / 266
	Nicolas FERRY	

V.2.3 - Scénario d'utilisation

Le SNI suivant modélise le scénario d'enchaînement des écrans de l'application INRA pour un scénario d'utilisation nominale.

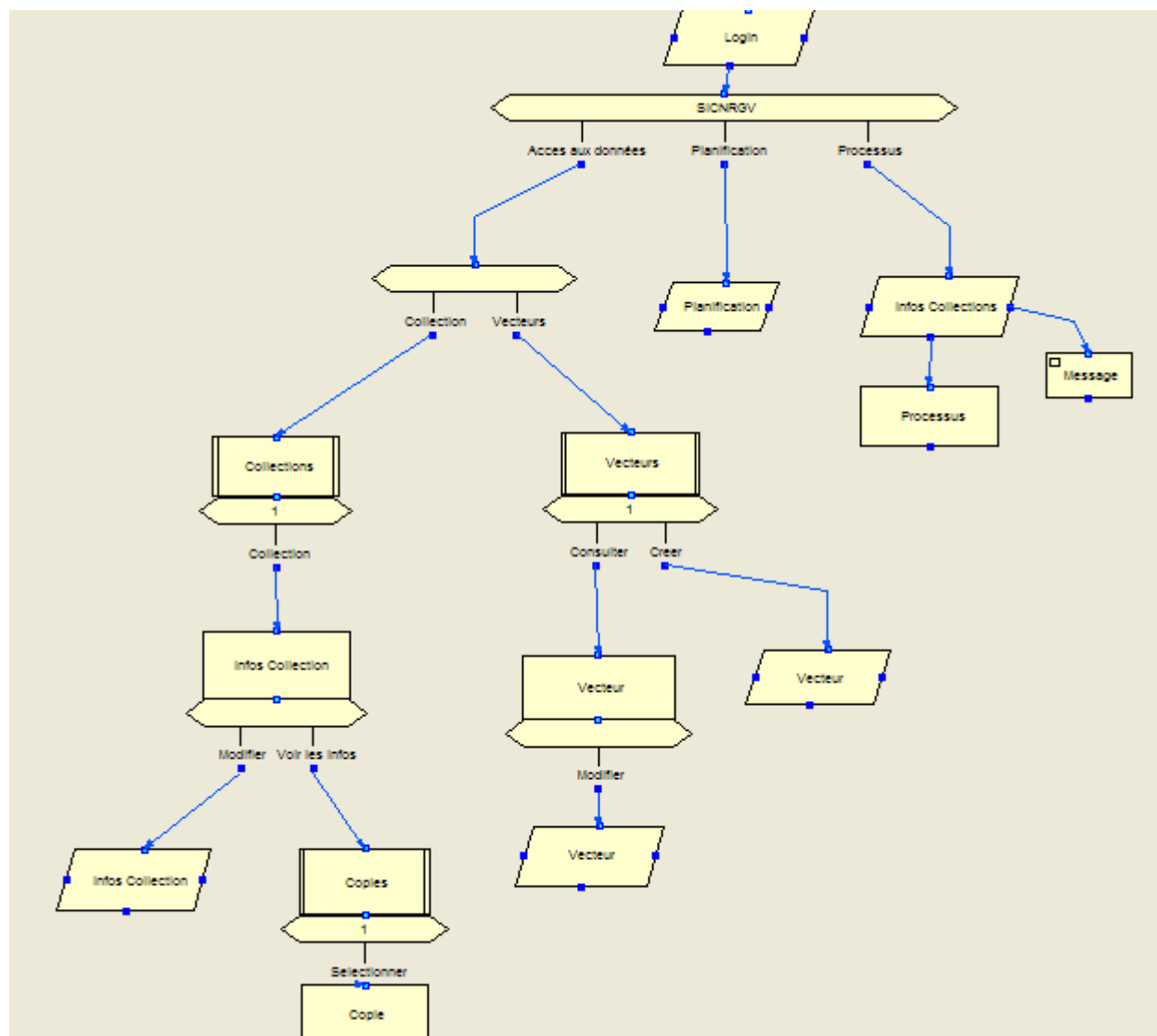


Figure 43 : SNI de l'application INRA pour le scénario nominal de test défini.

THESE-MACAO	Juin 2008	140 / 266
	Nicolas FERRY	

V.2.4 - Périmètre de la prestation

Le générateur est capable de générer à partir d'un modèle SNI, une maquette comportant les écrans et leurs enchaînements. Il gère la partie visuelle de l'application. Ceci correspond à la couche présentation d'une application multi-tiers.

V.2.5 - La description fonctionnelle de la solution

Lors d'un atelier participatif l'équipe projet désire acquérir la compréhension de l'IHM et l'enchaînement fonctionnel demandé par le client. La personne en charge de l'IHM : l'architecte IHM, saisit un SNI global à partir de la connaissance métier préalablement acquise et des souhaits du client. La logique fonctionnelle est donc adaptée visuellement à ce qui doit être présenté à l'utilisateur. Ainsi un modèle SNI est créé en séances, nous appelons cela dans la méthode MACAO : le mode esquisse. La saisie du SNI est faite dans VisualSNI sous une plate-forme Eclipse configurée pour les ateliers. L'enchaînement des écrans est fait directement dans l'outil projeté sur grand écran. La construction est faite en partenariat avec le client.

Une fois l'IHM globale saisie, l'architecte peut générer une maquette avec l'outil. L'outil propose une liste d'écran correspondante au modèle SNI. Pour la génération de la maquette, l'outil parcourt le modèle et applique pour chaque objet du SNI une transformation. Chaque transformation se base sur des "templates" ou patrons ou modèles au sens : des exemples. Ces exemples sont définis avant l'atelier ou à partir des projets précédents. Ils constituent ainsi une base d'expérience d'écrans applicatifs déjà utilisés ou prévus spécialement. La transformation est une substitution qui remplace chaque objet du SNI par son "modèle final de maquette" à obtenir.

Une liste des écrans lui est présentée, il peut assigner les Objets du SNI à des écrans, ensuite il doit choisir parmi les patrons celui qu'il souhaite appliquer. La présence de l'architecte permet de garder de la souplesse dans le processus et dans le choix final pour faire face aux cas litigieux.

Après génération, la maquette peut être montrée au client pour lui donner une idée globale du fonctionnement applicatif ou pour lui présenter graphiquement un retour. Le générateur produit ainsi l'ensemble du squelette de code de la partie IHM qui sera reprise pour le développement.

THESE-MACAO	Juin 2008	141 / 266
	Nicolas FERRY	

V.2.6 - La description technique de la solution

VisualSNI et le générateur sont des plugins Eclipse. Le générateur est séparé de l'éditeur VisualSNI ainsi la plate-forme peut accueillir plusieurs transformateurs différents. La connexion de plusieurs générateurs permet l'interopérabilité générale de l'outil. Il faut bien sûr développer un traducteur pour chaque cible technologique.

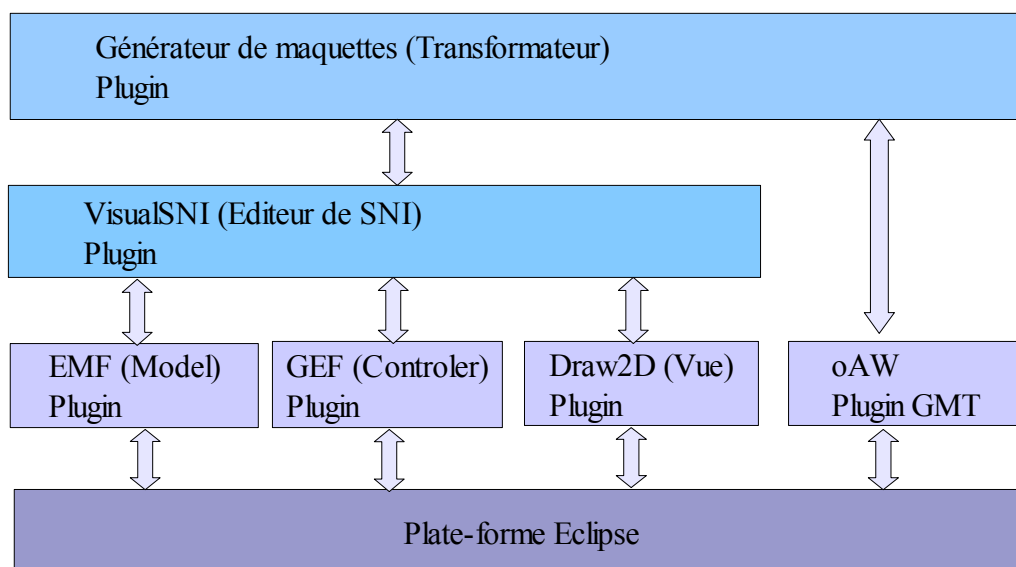


Figure 44 : Architecture des couches VisualSNI et du générateur de maquette. Alors que VisualSNI utilise EMF, GEF et Draw2D. Le générateur s'appuie sur OpenArchitectureWare pour la génération à partir de patrons.

La maquette génère dans la technologie JSF et produit une application Web à part entière. JSF est composé d'un ensemble d'API pour la gestion de l'interface homme-machine avec la gestion d'événements, définition des pages et gestion interne. Il est fourni avec un ensemble de composants graphiques les « Faces » qui étendent l'interface graphique avec des composants plus évolués.

Dans un premier temps, une application Web dite « application blanche » a servi de base pour la construction du modèle de l'application : le générateur reproduisant ce contexte à chaque génération. Ceci a permis de valider la plate-forme cible JSF et d'éprouver la technologie.

V.2.7 - Technologies pour la génération

Pour le générateur lui-même, le projet GMT (Generative Modeling Technologies) [16] est constitué d'un ensemble de sous-projets visant à inclure la génération MDE dans Eclipse. Parmi ces projets nous avons utilisé OpenArchitectureWare [] comme moteur de transformation à base de patrons [14]. Ce module offre l'avantage d'inclure des mécanismes de transformations de modèles à modèles, de modèles vers du Code, et de fournir des langages de transformations et de vérifications de contraintes. C'est à partir du moteur oAW que nous avons construit le plugin du générateur qui sera appelé aussi « SNI2JSF ».

Pour le transformateur, il faut aussi créer les patrons ("templates") correspondant aux pages JSF en les écrivant avec le langage Xpand. Les pages modèles doivent correspondre aux types d'objets du SNI c'est-à-dire un écran de saisie, un écran de présentation simple, un écran de présentation de liste... Le stéréotypage ainsi que les champs d'identification permettent à l'architecte de définir des patrons de pages avec une granularité plus fine. Si plusieurs templates sont disponibles pour un objet dans le modèle source, l'architecte peut choisir le template à appliquer qui lui convient le mieux.

Ainsi le moteur de transformation va effectuer les actions de transformation définies sous la forme d'une machine à état. Le raisonnement est inversé pour les templates qui eux produisent les éléments finaux sous forme de fichiers alors qu'en entrée, nous traitons des unités de dialogues. Les templates génèrent par exemple, les fichiers des pages Web JSF, les fichiers des servlets associés, les fichiers d'enchaînement des pages et les fichiers de paramètres.

Les transformateurs de VisualSNI sont structurés comme suit :

- Le métamodèle du modèle source en fournit en entrée
- Un ensemble de patrons définissent les transformations et la dynamique de la transformation
- Un modèle source (référence) est transmis au transformateur
- La maquette est générée en sortie sous forme de fichiers format l'application Web.

THESE-MACAO	Juin 2008	143 / 266
	Nicolas FERRY	

Le transformateur

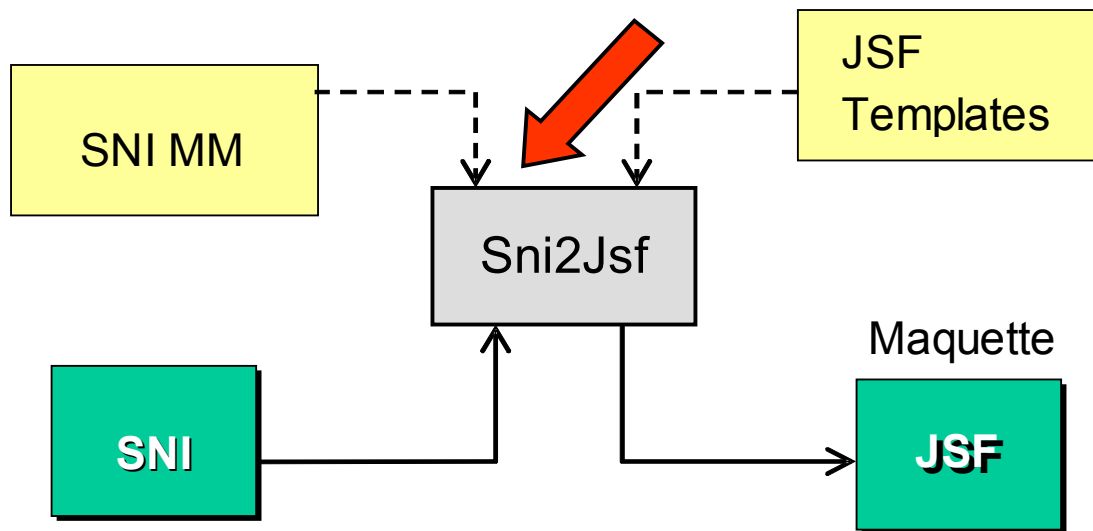


Figure 45 : Un transformateur à base de templates nécessite le métamodèle du SNI (SNI-MM), et un ensemble de templates (patrons) de transformation dans la technologie cible JSF. En passant un modèle SNI source, on obtient une maquette avec l'IHM générée sous forme de pages Web.

C.V.3 - Structure interne du générateur

V.3.1 - Structure d'un nouveau générateur :

Pour concevoir le générateur, les architectes IHM vont créer un nouveau plugin Eclipse. Dans notre exemple, le plugin correspond à SNI2JSF.

Dans la structure interne, nous devons avoir les paquetages suivants :

- metamodel (SNI-MM)
- template (JSF Templates)
- workflow (generator.oaw de SNI2JSF)

Le métamodèle du SNI est exactement celui utilisé par l'éditeur graphique VisualSNI. Les templates définissent les patrons de transformations. Le fichier generator.oaw est le fichier de configuration de l'automate de transformation.

THESE-MACAO	Juin 2008	144 / 266
	Nicolas FERRY	

Voici la structure du fichier generator.oaw :

```
<workflow>
  <property name="model" value="sicnrgv.sni"/>
  <property name="src-gen" value="src-gen"/>

  <!-- Model Source : SNI -->
  <component class="org.openarchitectureware.emf.XmiReader">
    <metaModelFile value="src/metamodel/sni.ecore" />
    <modelFile value="${model}" />
    <outputSlot value="model" />
    <firstElementOnly value="true" />
  </component>

  <!-- Effaçage du repertoire de sortie -->
  <component id="dirCleaner"
class="org.openarchitectureware.workflow.common.DirectoryCleaner"
skipOnErrors="true">
    <directories value="${src-gen}"/>
  </component>

  <!-- Application des templates de Transformation -->
  <component class="org.openarchitectureware.xpand2.Generator">
    <metaModel
      class="org.openarchitectureware.type.emf.EmfMetaModel">
      <metaModelFile value="src/metamodel/sni.ecore" />
    </metaModel>
    <expand value="template::JSF_File::jsp FOREACH
model.lstObjets" />
    <genPath value="${src-gen}" />
  </component>

  <component class="org.openarchitectureware.xpand2.Generator">
    <metaModel
      class="org.openarchitectureware.type.emf.EmfMetaModel">
      <metaModelFile value="src/metamodel/sni.ecore" />
    </metaModel>
    <expand value="template::JSF_Bean::bean FOREACH
model.lstObjets" />
    <genPath value="${src-gen}" />
    <outlet path="${src-gen}">
      <postprocessor class="oaw.xpand2.output.JavaBeautifier"/>
    </outlet>
  </component>

  <component class="org.openarchitectureware.xpand2.Generator">
```

THESE-MACAO	Juin 2008	145 / 266
	Nicolas FERRY	

```

<metaModel
    class="org.openarchitectureware.type.emf.EmfMetaModel">
    <metaModelFile value="src/metamodel/sni.ecore" />
</metaModel>
<expand value="template::JSF_Config::config FOR model" />
<genPath value="${src-gen}" />
<outlet path="${src-gen}">
    <postprocessor class="oaw.xpand2.output.XmlBeautifier"
fileExtensions=".xml"/>
</outlet>
</component>
<!-- etc ... -->

</workflow>

```

Figure 46 : L'automate de génération de oAW préparé pour la génération de template JSF.

Remarques sur le fichier generator.oaw :

La property model définit le fichier SNI source à transformer.

La property src-gen spécifie le répertoire de sortie de la génération

D'autres propriétés peuvent être utilisées pour définir des variables pour le workflow.

Ensuite, les « composants » sont les actions de traitement à réaliser dans l'ordre pour achever la transformation globale. Parmi elles, nous pouvons citer : le chargement du fichier, l'effacement des répertoires de sorties, le lancement des transformations en utilisant les templates, etc.

Il reste à construire les templates de transformations qui sont placés dans le paquetage « templates » dans le répertoire « src » du plugin.

Les templates sont des patrons de codes ou de fichiers à générer qui vont adopter la syntaxe du fichier de sortie tel qu'il serait sans génération. La partie statique correspond au code qui est généré (par exemple du JSF) et la partie dynamique est interprétée par le moteur de transformation.

Les templates sont écrits en langage Xpand2. La description de la syntaxe est très simple et claire. Une description de référence du langage Xpand peut se trouver sur le site d'Eclipse : <http://www.eclipse.org/gmt/oaw/doc/>

Dans le cadre de la technologie JSF, il y a quatre groupes de fichiers à générer :

- Les fichiers métiers Beans qui définissent la classe en lien avec la page Web affichée et qui va gérer son comportement et ses traitements.

THESE-MACAO	Juin 2008	146 / 266
	Nicolas FERRY	

- Les fichiers des pages JSF à proprement parler.
- Le fichier de configuration de l'enchaînement des pages : faces-config.xml
- Le fichier des ressources ou des paramétrages des textes statiques des pages.

Voilà l'arborescence obtenue dans le paquetage du plugin de génération :

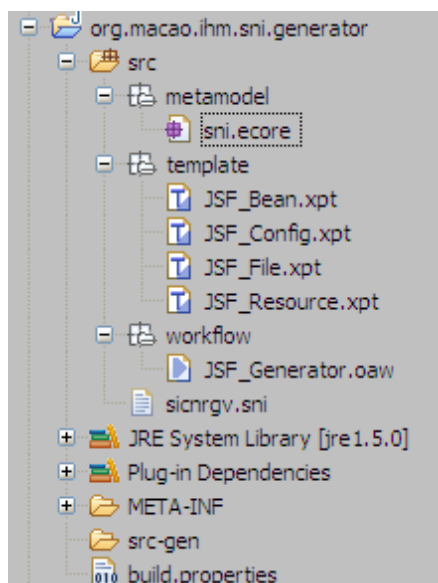


Figure 47 : Création d'un plugin de génération pour visualSNI. Nous retrouvons le métamodèle du SNI, les templates, l'automate de génération, et le modèle SNI source : *sicnrgv.sni*

Pour les fichiers « template », il est nécessaire de développer un fichier de template par type de fichiers à générer. Par exemple, un template JSF pour générer toutes les pages JSF, un template faces-config pour générer le fichier d'enchaînement des pages JSF, etc. Un fichier template est écrit en langage Xpand2 et possède l'extension xpt (exp: MonTemplate.xpt).

La configuration du générateur est prête, il faut lui fournir à présent le fichier SNI source à transformer. Grâce à l'éditeur de SNI, nous avons généré l'arborescence de façon proche de celle employée sur le projet SICNRGV pour l'INRA.

V.3.2 - Lancement d'une génération :

Pour lancer une génération, il faut paramétrer la source et la destination.

Dans le fichier generator.oaw cela se traduit par ces deux valeurs :

<workflow>

THESE-MACAO	Juin 2008	147 / 266
	Nicolas FERRY	

```
<property name="model" value="sicnrgv.sni"/>
<property name="src-gen" value="src-gen"/>
```

Le lancement doit se faire en appelant oAW avec le fichier generator.oaw. Depuis Eclipse, cela se fait par le bouton droit sur generator.oaw puis Run As / oAW Workflow. Les résultats sont inscrits au fur et à mesure dans la console, indiquant le nombre de fichiers générés par traitement de l'automate et/ou les erreurs s'il y en a.

Les fichiers sont générés dans le répertoire de sortie :

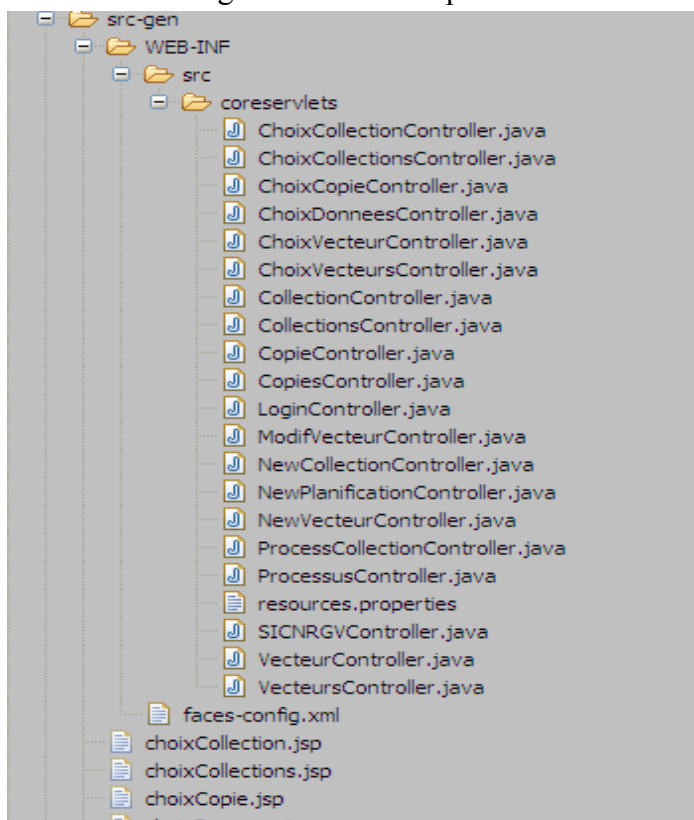


Figure 48 : Fichiers générés directement dans l'application Web. Ici, nous voyons les fichiers des servlets générés pour chaque UD du SNI du SI-CNRRGV de l'INRA, le fichier de configuration faces-config.xml et quelques pages JSF.

THESE-MACAO	Juin 2008	148 / 266
	Nicolas FERRY	

V.3.3 - Création du template des pages JSF :

Le générateur permet de reconstruire des applications suivant des modèles appelés template. Initialement il est nécessaire de créer les modèles à partir d'une application de référence. Tout d'abord, nous avons construit un fichier JSF type sur la plate-forme finale donnant le cadre de ce que nous voulons pour chaque type de fichier.



Figure 49 : Exemple de Page JSF affichée par le navigateur. Les champs login et password sont connectés à leur servlet contrôleur suivant le modèle MVC.

Cette page JSF utilise des balises classiques du HTML. La norme JSF rajoute des extensions au langage sous la forme de nouvelles balises. Ces bibliothèques d'extensions sont appelées « taglib ». Par exemple dans la balise « form » sont créés les composants permettant de faire la saisie du login et du mot de passe. Les informations saisies sont reliées automatiquement au contrôleur appelé « servlet ».

```
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<f:loadBundle basename="coreservlets.resources" var="res"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Login</TITLE>
<LINK REL="STYLESHEET"
      HREF="./css/styles.css">
```

THESE-MACAO	Juin 2008	149 / 266
	Nicolas FERRY	


```

        TYPE="text/css">
</HEAD>
<BODY>
<CENTER>
<IMG src="./images/login.gif">
<BR>
<BR>
    <h:outputText value="#{res.loginTitle}"/>
<P>
<h:form>
    <h:outputText value="#{res.loginLogin}"/>:
    <h:inputText value="#{loginController.login}"/><BR>
    <h:outputText value="#{res.loginPassword}"/>:
    <h:inputText value="#{loginController.password}"/><BR>
    <h:commandButton value="#{res.buttonLabel}"
        action="#{loginController.submit}"/>
</h:form>
</CENTER></BODY></HTML>
</f:view>

```

Figure 50 : Exemple d'une page Web JSF typiquement utilisée pour faire une saisie d'un login. C'est elle qui va servir de modèle pour construire le template. Les composants graphiques sont ici connectés à leur servlet contrôleur par un système de binding.

Il faut identifier les parties statiques, des parties dynamiques qui seront configurées par le générateur. Les parties dynamiques marquées arbitrairement « en rouge » sur la page sont remplacées par le code de substitution en Xpand2.

Un fichier template commence généralement par importer le métamodèle afin de pouvoir dans l'éditeur de template d'Eclipse proposer les objets du métamodèle et de vérifier les variables avec un typepage statique.

```

«IMPORT sni»      // Importation du métamodèle

// Définition de la template jsp pour l'objet AffObjet d'un SNI
«DEFINE jsp FOR AffObjet»
«IF name != null»      // assurance que le nom de l'objet existe
«FILE name+".jsp"»      // Crée un fichier avec le nom de l'objet +jsp
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<f:view>
<f:loadBundle basename="coreservlets.resources" var="res"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>

```

THESE-MACAO	Juin 2008	150 / 266
	Nicolas FERRY	

```

<TITLE><<dispObjet>></TITLE>          // Utilisation d'une variable du metamodelle
<LINK REL="STYLESHEET"
      HREF="./css/styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<CENTER>
// Condition dépendant d'un attribut
<<IF dispObjet.toLowerCase() == "planification">>IMG
src="./images/planification.png"><<ENDIF>>
<BR>
<BR>
      <h:outputText value="#{res.<<name.toFirstLower()>>Title}"/>
<P>
<h:form><<IF dispAttrib == null>>
      <h:outputText value="#{res.<<name.toFirstLower()>>Var1}"/>:
      <h:inputText value="#{<<name.toFirstLower()>>Controller.var1}"/><BR>
      <h:outputText value="#{res.<<name.toFirstLower()>>Var2}"/>:
      <h:inputText value="#{<<name.toFirstLower()>>Controller.var2}"/><BR> <<ELSE>>
// Boucle sur une liste d'attributs
<<FOREACH dispAttrib.replaceFirst("\\(", "").replaceAll("\\)", "").split(",")
AS attr>>
      <h:outputText
value="#{res.<<name.toFirstLower()>><<attr.toFirstUpper()>>}"/>:
      <h:inputText
value="#{<<name.toFirstLower()>>Controller.<<attr.toFirstLower()>>}"/><BR>
<<ENDFOREACH><<ENDIF>>
      <h:commandButton value="#{res.buttonLabel}"
action="#{<<name.toFirstLower()>>Controller.submit}"/>
</h:form>
</CENTER></BODY></HTML>
</f:view>
// Terminaison des commandes XPAND
<<ENDFILE>>
<<ENDIF>>
<<ENDDDEFINE>>

```

Figure 51 : Un template pour chaque type d'UD va définir la transformation à opérer.

Nous définissons par granularité du générateur, la couverture entre une unité de dialogue du SNI et le code généré de l'application. La granularité d'un générateur peut varier suivant les besoins. Il peut couvrir un template pour un type d'unité de dialogue du SNI. Il est possible de descendre jusqu'à un niveau plus fin en définissant : un template pour une unité de dialogue et un type d'écran (login, affichage principal, de liste, saisie de formulaire, etc.). Ou bien encore de descendre à un template par écran. Cette granularité permet de gérer tous les cas au prix de devoir recréer un template pour chaque écran.

THESE-MACAO	Juin 2008	151 / 266
	Nicolas FERRY	

Bien évidemment, plus la maquette utilise une granularité large, plus les différents écrans présents dans la maquette se ressembleront. Inversement plus il faudra créer de template par écran plus la maquette sera précise et adaptée.

En l'occurrence, il ne faut pas perdre de vue que le gain d'un générateur de maquette ne se fait pas sur une seule application mais sur un ensemble d'applications. Chaque application terminée permet de construire des templates riches qui sont directement réutilisables par le générateur. C'est le gain le plus important puisqu'il permet de pérenniser l'expérience de l'entreprise en la proposant à nouveau sous forme de maquette.

C.V.4 - La fusion multi-modèles

V.4.1 - La fusion de modèles

Ce générateur se concentre principalement sur la partie IHM. Une évolution serait de couvrir la génération inter-domaines, c'est-à-dire accepter la conception utilisant plusieurs modèles différents. Pour cela, il y a deux approches. L'une consiste à établir un Mégamodèle [22]. Un mégamodèle est un modèle qui prend en compte la génération à partir de plusieurs points de vue et qui permet d'en fusionner les concepts. Ce n'est pas simplement un métamodèle mais plutôt la recherche du plus grand commun dénominateur de plusieurs modèles.

Une autre approche consiste à utiliser la programmation orientée-aspect comme la continuité de la modélisation regroupant plusieurs points de vue. En effet, la programmation orientée-aspect représente une manière élégante de fusionner les modèles en accord avec l'architecture du logiciel. En fait, la modélisation par les modèles représente un point de vue humain sur le système alors que la programmation orientée-aspect est une vue système au travers d'aspects (Sécurité, Journalisation, etc.). Des lors, il suffit d'utiliser des aspects plus larges comme l'IHM, les composants/objets [140], et les données pour faire fusionner des concepts issus de modèles différents. Les points de coupures et les points de jonctions peuvent être utilisés pour tisser au niveau physique la génération de plusieurs modèles. En effet, chaque modèle va générer dans le code final la partie qui le concerne qui sera insérée aux points de jonctions spécifiés par les architectes.

VisualSNI supporte déjà la dernière approche car le moteur de transformation oAW du projet Eclipse GMT qui utilise le langage Xpand2 gère en natif les concepts de points de coupures et points de jonctions de la programmation orientée-aspect. Il est donc possible de faire fusionner différents modèles provenant de domaines différents avec cette technologie.

THESE-MACAO	Juin 2008	152 / 266
	Nicolas FERRY	

V.4.2 - XPand2

XPand2 est un langage de transformation qui a l'avantage d'offrir une syntaxe simple et pourtant très efficace. XPand2 est un langage qui permet aussi de faire de la programmation orientée-aspect. Il inclut les concepts de point de jonction et de point de coupure qui permettent d'insérer un « greffon » c'est-à-dire une partie de programmation qui concerne un domaine ou aspect du problème. Aujourd'hui, les fruits de notre recherche se focalisent sur la couche de Présentation. La construction d'application informatique comprend bien sûr de nombreuses autres couches toutes aussi importantes. A terme, la conception assistée du MDE sera plus complète puisqu'elle couvrira la génération de toutes les couches (données, logique, présentation, autres...).

Pour enrichir ces propos, il paraît intéressant de souligner que pour le MDE il faudra générer les couches de l'application en utilisant des modèles sur les couches de la structuration logicielle, dans la mesure où ces vues sont pratiquement orthogonales ce choix permet de séparer les principes. En pratique cela signifie qu'il faut un rôle « architecte » pour créer chaque modèle du domaine qui sera pris en charge par une ou plusieurs personnes.

Dans l'approche orientée-aspect, le système décorrèle les préoccupations en aspects. Le système est alors vu comme un tissage de plusieurs aspects. Par exemple on trouve souvent : l'aspect de la sécurité, l'aspect de la journalisation, l'aspect de l'authentification,...

Dans l'ingénierie des modèles, nous utilisons la notion de point de vue pour décrire une partie du système adaptée à un rôle donné. Les deux notions aspects et points de vue semblent pouvoir s'aligner dans le domaine informatique. Les points de vue étant en conception une représentation du système suivant des critères spécifiques. Les aspects semblent être une considération depuis le système. Ainsi, les points de jonctions sont une façon de relier plusieurs points de vue physiquement au système.

Dans le cas de l'ingénierie des modèles cela équivaut à ce que les concepteurs utilisent plusieurs modèles, de représentation du système, tangents entre eux ou non et à fusionner leur vision pour produire le système final. Typiquement, c'est dans cette dernière phase que la programmation orientée-aspect trouve sa place car elle offre, du point de vue du système, un moyen de fusionner des concepts de domaines différents.

Il paraît intéressant d'utiliser la programmation orientée-aspect pour faire converger des points de vue des concepteurs humains plutôt que purement techniques avec un découpage d'aspect à

THESE-MACAO	Juin 2008	153 / 266
	Nicolas FERRY	

posteriori sur des éléments du système. Autrement dit cela ne sert pas à grand chose de concevoir un aspect sécurité par exemple si personne n'est en charge de la concevoir.

C.V.5 - L'utilisation des patrons comme base de connaissances

La transformation de modèles peut se faire par traduction directement du modèle conceptuel vers un modèle logique puis vers un modèle physique. Les patrons utilisés par le moteur de transformation contiennent l'équivalent des informations rajoutées à chaque niveau.

Dans le principe une première application est réalisée. Elle sera reprise dans le patron de conception comme base pour la génération. Les parties statiques communes ne seront pas changées tandis que les parties dynamiques seront interprétées par le moteur de transformation.

Dès lors, un patron fonctionnel donne une structure qui pour la référence donnée produira des données adaptées. Pour enrichir un objet, il est possible de concevoir des patrons différents.

Les patrons peuvent ainsi être vus comme l'expérience de l'entreprise. Plus il y aura de projets réalisés, plus il y aura potentiellement une source large de choix de patrons pour chaque élément modélisé.

C.V.6 - Bilan

Ce chapitre a présenté un module de génération du code IHM qui, à partir d'un modèle SNI est capable de générer des maquettes dans une technologie particulière. Des standards et des outils liés à la transformation de modèles ont été présentés succinctement. Parmi ceux-ci le générateur utilise OpenArchitectureWare comme le moteur de transformation et participe au système d'échange communicatif interne et externe à l'axe IHM.

Le cahier des charges du générateur a été établi puis la spécification fonctionnelle et enfin une description technique de la solution. Cette étude a permis de définir le transformateur en étudiant les liens de corrélation entre modèles à partir de leurs métamodèles. Elle a notamment permis de spécifier la cible de génération de l'outil ainsi que de définir un prototype utilisant le moteur oAW pour générer les transformations.

Pour analyser le comportement de l'outil, un projet mené à Capgemini pour le client INRA a été repris. Ce projet a été simplifié sur un scénario d'utilisation qui réduit en terme de quantité et de complexité le nombre d'écrans. Il permet de vérifier la correspondance entre la maquette et le projet. Avec ce prototype, nous avons prouvé le concept et la faisabilité d'une génération

THESE-MACAO	Juin 2008	154 / 266
	Nicolas FERRY	

de l'IHM. Ceci ouvre de nombreuses perspectives quant à l'implémentation des générateurs pour tous les niveaux de l'axe IHM.

Nous avons vu aussi comment créer un générateur en utilisant la syntaxe de XPAND2. Enfin, nous montrons le lien avec la programmation orientée-aspects notamment son avantage pour faire fusionner et converger plusieurs modèles pour la génération globale de l'application. Le tissage de modèle issu des prémices des liens de l'IDM et de AOP devrait devenir une voie pour la conception des futures applications.

Un autre intérêt, pour l'entreprise cette fois, réside dans la structure même de ce générateur à base de template en récupérant, dans le patron de conception considéré comme base de connaissance, l'expérience et le savoir-faire de l'entreprise dans une technologie et un type d'écran donné.

Ce générateur devrait pouvoir servir à Capgemini pour réaliser des maquettes de technologies différentes d'une même application basée sur leur expérience précédemment acquise.

THESE-MACAO	Juin 2008	155 / 266
	Nicolas FERRY	

Partie D - Résultats

THESE-MACAO	Juin 2008	157 / 266
	Nicolas FERRY	

Chapitre I : Conclusion

Le génie logiciel apporte des solutions pour la création de logiciels dont l'écriture est la spécificité des Sociétés de Services en Ingénierie Informatique. Ces sociétés produisent chaque jour des logiciels dans des domaines variés et l'une de leurs problématiques est de surmonter les difficultés et les risques inhérents lors de leurs projets. Le travail d'une SSII fait l'objet d'une remise en cause permanente, un nouveau projet apporte son lot de risques dû à ses technologies nouvelles et ses intervenants avec leurs compétences diverses. C'est un grand défi pour le monde de l'informatique qui doit faire face à une complexification exponentielle et à une grande richesse d'innovation. Le génie logiciel tente de répondre à ces problématiques en proposant des approches qui veulent réduire un maximum ces difficultés.

Parmi les solutions proposées nous avons vu que les « méthodologies » visaient à construire un procédé et à pouvoir le répéter pour atteindre le résultat recherché. Nous avons décrit un bref historique des types de démarches et méthodes existantes. Nous avons présenté la méthode MACAO qui est née du besoin de proposer une synthèse des meilleures méthodes actuelles du génie logiciel. Elle est apparue à peu près au même moment que le Rational Unified Process. Elle préconise un processus, et promulgue sous forme de conseils ses recommandations en favorisant les compromis à tous les niveaux. MACAO est une méthode participative et interactive originale qui apporte des concepts novateurs notamment par le fait de placer l'utilisateur final au centre de l'étude, par son analyse détaillée de l'aspect IHM, par sa réalisation de prototypes incrémentaux, par ses règles de non régression entre prototypes.

Nous avons étudié en particulier le cycle de vie des différentes méthodologies car il définit les principales étapes pour la réalisation d'un logiciel. Nous avons alors cherché à étudier les cycles de vies utilisés à Capgemini pour des projets objets standard afin de déterminer les critères avantages et inconvénients de chacun. Cette étude a permis notamment de dégager qu'aucune méthode n'était « parfaite », au mieux une méthode résout certains problèmes particuliers. Ce qui est important c'est de la choisir au mieux pour s'adapter à un problème donné.

Le génie logiciel propose d'utiliser la « modélisation » pour représenter des problématiques de manière spécifique afin de mieux les contrôler. C'est dans cette optique que MACAO présente un ensemble de diagrammes spécialisés pour l'IHM. Mais, et c'est le problème, tant que la méthode ne possèdera pas de plate-forme convenable et d'outils performants pour l'utiliser, son utilisation restera pour des projets assez limités.

THESE-MACAO	Juin 2008	158 / 266
	Nicolas FERRY	

L'un des travaux de cette thèse a consisté à apporter le support informatique aux modèles d'interfaces homme-machine de la méthode MACAO. Pour cela, nous avons adopté un découpage en trois niveaux avec le niveau conceptuel, logique et physique. Ce découpage est connu pour sa réussite dans plusieurs autres domaines informatiques. Nous avons alors présenté les diagrammes pour chacun des niveaux.

Ainsi, nous avons décrit le schéma navigationnel des interfaces avec ses objectifs, sa syntaxe, et ses caractéristiques. Puis nous avons parlé du schéma d'enchaînement des fenêtres et de la représentation visuelle associée avec les dessins des écrans.

Il faut remarquer que le passage d'un modèle de haut niveau à un niveau inférieur manipule les mêmes concepts qui évoluent par raffinement. A contrario, le passage d'un modèle de niveau inférieur vers un niveau supérieur constitue une abstraction qui masque de l'information. En fait, nous rajoutons ou enlevons de la sémantique à chaque étape de transformation.

Pour la réalisation d'un projet informatique, il est important de rappeler l'importance de l'entente et de la convergence des participants pour l'établissement d'une solution commune satisfaisante. De même le rôle de l'utilisateur final est primordial car il faut sa participation active dans l'équipe. Pour des projets significatifs, la conception de l'IHM doit être considérée au même titre que le métier ou les données, par l'ensemble des intervenants dès les premières phases du projet. Pour cela, nous nous sommes basés sur le mode de fonctionnement préconisé par Capgemini avec l'emploi d'ateliers participatifs avec le client. L'acquisition des exigences d'IHM peut être faite en parallèle en atelier pour prendre en compte l'aspect IHM. Celle-ci permet notamment de recueillir les informations pertinentes en accord avec l'architecture globale choisie pour l'application. Dans les applications informatiques classiques, cela correspond aux couches de l'IHM, des processus métiers et des données.

Dans le cadre de l'étude de l'axe IHM, nous défendons la nécessité de séparer le conceptuel acquis avec le client, la logique pour l'ingénierie et le physique présenté en retour à l'utilisateur final. Pour cela il faut clairement définir la frontière de ces niveaux par la réponse aux questions du Quoi-fonctionnel, du Comment-technologique et Visuel, et de la réalisation concrète. MACAO introduit un modèle pour chaque niveau. Et chaque modèle est informatiquement géré par son métamodèle. Dès lors, il est possible de définir ces métamodèles comme des raffinements et des incréments des précédents. Le lien étant fait au niveau des métamodèles. L'existence des moteurs de transformation de l'ingénierie dirigée par les modèles permet de réaliser les différentes mutations des représentations. Après avoir brièvement décrit un transformateur comme un élément de base de cette transformation, nous avons finalement couvert l'architecture globale de transformation pour l'axe de l'IHM.

THESE-MACAO	Juin 2008	159 / 266
	Nicolas FERRY	

A partir de là, la construction du premier prototype de VisualSNI a pu être démarrée. Tout d'abord, la plate-forme Eclipse a été présentée ; celle-ci se compose d'un noyau qui gère des modules additionnels. Nous avons utilisé les capacités d'Eclipse pour construire un premier module d'éditeur graphique de SNI. Celui-ci a été ensuite amélioré et structuré avec une architecture de type Modèle-Vue-Contrôleur grâce aux modules GEF, EMF et Draw2d. Dès lors, les différentes parties ont été créées progressivement en intégrant le métamodèle puis le contrôleur et enfin les représentations graphiques par itérations successives.

La structure d'Eclipse se prête bien à ce type d'agencement et son système de vues est très efficace dans ce cadre. Nous avons détaillé l'architecture et les mécanismes dans les trois domaines Modèle-Vue-Contrôleur.

Enfin pour démontrer la faisabilité de génération sur l'axe IHM, nous avons développé un prototype qui traduit directement un SNI vers une maquette exploitable dans la technologie JSF. Ainsi nous couvrons le processus de génération dans sa globalité. La prochaine étape sera de construire les éditeurs graphiques des niveaux intermédiaires pour suivre le procédé de création complet. Les transformations associées devront être réalisées pour le passage entre modèles.

Pour le générateur, un transformateur a été défini et étudié ainsi que les liens de corrélation des modèles à partir de leurs métamodèles. Nous avons présenté succinctement des standards et des outils liés à la transformation de modèles. Le module OpenArchitectureWare a été utilisé pour le moteur de transformation à cause de son langage très efficace.

Le comportement de l'outil a été validé en reprenant le projet INRA mené à Capgemini. Ce prototype nous a prouvé que le concept d'une génération d'IHM était possible de bout en bout. Ceci ouvre de nombreuses perspectives quant à l'implémentation des générateurs pour tous les niveaux de l'axe IHM.

Nous avons montré un autre intérêt pour l'entreprise, qui cette fois résidait dans la structure même de ce générateur à base de templates. Celui-ci utilisant des patrons de conception, ils peuvent être vus comme une base de connaissance de l'expérience et une synthèse du savoir-faire de l'entreprise dans une technologie donnée. Ce générateur devrait pouvoir servir à Capgemini pour réaliser des maquettes de technologies différentes d'une même application basée sur les expériences précédemment acquises.

THESE-MACAO	Juin 2008	160 / 266
	Nicolas FERRY	

Chapitre II : Bilan et Perspectives

Le domaine des interfaces homme-machine est très peu considéré dans la création de logiciels qui sont par nature de plus en plus complexes en terme d'architecture et d'exigences des utilisateurs. A l'avenir, il deviendra primordial de modéliser les IHM au même titre que le sont les données [44] et les traitements.

En regardant de près, il est évident que le domaine des interfaces va subir des transformations majeures dans l'avenir. L'essor d'Internet a permis au Web de se démocratiser en générant des applications en client léger où l'affichage est déporté sur des postes clients. Ceci a permis de retrouver une architecture à trois, voire à n-tiers. La tendance actuelle va vers le client riche qui permettra de retrouver ce que cherche le Web depuis ses débuts : une interface aussi puissante et élaborée que les clients lourds mais en mode déporté. Les RIA (Rich Internet Application) sont considérées comme la prochaine révolution de l'interface du Web.

Actuellement, l'interface relie fortement la structuration de ses composants avec leur aspect visuel. Il est intéressant de souligner qu'il existe de plus en plus de tentatives pour décorréliser ces deux aspects. Le premier avantage est de pouvoir différencier les compétences du graphiste (ou designer [141]) qui peut soigner et adapter la présentation à l'utilisateur, de l'architecte IHM qui s'intéresse à la structuration informatique [162]. Cette tendance avait déjà fait son apparition avec les « css » mais elle semble prendre de plus en plus d'importance dans l'avenir. Nous constatons l'émergence de plusieurs langages conçus à ces fins et basés sur XML.

Les perspectives sont nombreuses puisque grâce à VisualSNI la plupart des difficultés ont été aplanies et la technologie éprouvée. La prochaine étape consiste à réaliser les éditeurs du niveau logique. Un ensemble de règles de transformation est en cours de réalisation pour transformer un SNI en SEF.

Le générateur de maquette peut être grandement amélioré notamment sur trois points. Le premier point concerne la couverture des unités de dialogues du SNI. Actuellement, le générateur n'interprète que des éléments simples, dans l'exemple JSF nous pourrions étendre l'analyseur pour prendre en compte les unités de dialogues composées.

Une deuxième amélioration possible concerne les patrons qui ne sont pas paramétrables pour une génération donnée. Il faudra pouvoir donner le choix à l'architecte IHM pour qu'il puisse saisir, pour une unité donnée, le patron correspondant à l'écran de maquette qu'il souhaitera générer. Ceci ferait gagner de l'intérêt à l'outil et éviterait d'avoir une maquette trop uniforme.

THESE-MACAO	Juin 2008	161 / 266
	Nicolas FERRY	

Une fenêtre de choix permettrait de présenter pour chaque unité de dialogue du SNI une liste de patrons parmi lesquelles il faudrait choisir un patron à appliquer. Les patrons seraient apparentés à une base de connaissances du savoir-faire de l'entreprise. Ceci nécessite, pour un processus qui n'est pas entièrement automatisé, de gérer les nouveaux patrons en phase de finalisation du projet pour les rédiger et les insérer dans la base de connaissances.

Un troisième point d'amélioration concerne la phase de génération qui souvent nécessite d'ajouter de l'information au modèle initial. L'écueil à éviter pour une génération paramétrée est de vouloir remonter les paramètres au niveau du modèle initial. Ceci a plusieurs inconvénients : les informations de « tag values » rajoutées au modèle après coup, sont équivalentes pour introduire des informations d'un niveau inférieur à un niveau abstrait, autrement dit cela dénature le modèle initial. Ensuite que se passe-t-il si plusieurs générations de technologies différentes doivent être produites ? Il peut y avoir un chevauchement des « tag values remontées au niveau du modèle ». A l'inverse placer les informations dans le modèle logique ne résout pas le problème. Pour ces raisons nous conseillons de simplement placer les paramètres de génération au niveau de la transformation elle-même. Par exemple, le choix d'un patron pour une unité de dialogue équivaut à paramétrer le type ou le patron à appliquer à cette unité. Un autre exemple pourrait être le choix de la couleur dominante dans le patron appliqué. Nous le voyons tous ces paramètres doivent le plus possible rester au niveau de la transformation.

Pour terminer, il faut souligner que le générateur requière un architecte qui connaisse à la fois le codage de la technologie cible et la structure du métamodèle du SNI. Le temps de formation est rapide puisque une seule journée est nécessaire pour former un architecte à la création de templates pour VisualSNI. Car l'écriture du langage de transformation permet de faire des choses puissantes assez simplement et rapidement.

La plate-forme MACAO a désormais ses premiers modules disponibles, ils permettent de mettre en oeuvre la méthode MACAO, pour ceux qui le souhaitent, de manière outillée. Le métamodèle du SNI est ouvert vers l'extérieur et offre la possibilité de réaliser des passerelles vers et depuis d'autres modèles ou programmes. VisualSNI s'intègre comme un outil pour les architectes désireux de modéliser des IHM que ce soit pour des phases d'avant-ventes, ou dans des ateliers participatifs, ou dans le processus de création.

Nous avons décidé avec la collaboration de Capgemini de porter le projet dans le monde open-source. Ce n'est pas le premier projet à Capgemini qui est amené en open-source puisque le projet MDA Workbench, qui présente une plate-forme de transformation de modèles de type MDA, avait été un précurseur. L'idée défendue à Capgemini est que le projet

THESE-MACAO	Juin 2008	162 / 266
	Nicolas FERRY	

doit passer par l'extérieur pour être profitable en retour à l'entreprise. D'autres entreprises ont fait le pas du MDA [25].

Le module VisualSNI ainsi que VisualSNI-Generator ont été placés dans le domaine public sur SourceForge. VisualSNI a passé plusieurs séries de bêta-tests auprès de personnes choisies et a subi les utilisations répétées des étudiants de l'IUT de Blagnac, de Master informatique et d'école d'ingénieurs. Le produit est stable pour une utilisation industrielle et quotidienne. Nous maintenons ses modules et espérons à présent créer une communauté de personnes autour de ce projet afin de créer une dynamique pour poursuivre ses évolutions en fonction de nouvelles demandes des industriels.

Cette contribution offre plusieurs perspectives :

En fournissant librement la structure du SNI et son métamodèle documenté, nous espérons ouvrir sur des échanges et des communications dans différents domaines de l'ingénierie du logiciel. Une communauté de développeurs va progressivement se former autour de MACAO et des outils comme VisualSNI. Ils vont ainsi pouvoir contribuer à leur tour, et garantir une certaine pérennité en le maintenant et en suivant ses évolutions futures.

Enfin, le projet open-source est sous licence GPL. Ceci garantit que le travail fourni ne pourra pas être redistribué sans livrer les sources d'origine. Avec la GPL v2 de 1991, les maquettes et les produits des générateurs ne sont quant à eux, pas soumis aux restrictions de fourniture des sources. Cela veut dire qu'ils peuvent être utilisés par Capgemini sans contrainte particulière.

Au terme de cette année, mon directeur de thèse Jean-Bernard Crampes va prendre sa retraite et monter une société de consultant en informatique. Avec les outils développés dans le cadre de cette thèse, il a la possibilité de conseiller, de diffuser et d'appliquer la méthode MACAO auprès des entreprises qui le souhaitent. L'open-source lui garantira une maintenance et une évolutivité en fonction des membres impliqués dans le projet.

THESE-MACAO	Juin 2008	163 / 266
	Nicolas FERRY	

Chapitre III : Contribution

Dans cette thèse, mon travail s'est porté sur la conception théorique et pratique des modèles d'IHM de la méthode MACAO permettant d'utiliser celle-ci en milieu industriel.

Cette thèse souligne l'importance de placer l'utilisateur final au centre du système. L'IHM étant à la frontière entre l'homme et la machine, nous insistons sur l'importance de prendre en compte celle-ci au même titre que les données et les traitements dans la conception d'un logiciel informatique.

Nous avons cherché comment insérer la démarche MACAO sur le marché en étudiant la cible potentielle et son environnement. Cette étude menée dans une démarche industrielle a fait émerger à de multiples niveaux les grandes forces en présence. Elle a permis aussi de préciser le but qui en résumé précise que l'implantation des outils devait se faire en Java sur la plateforme Eclipse. Cette plate-forme est supportée par la communauté open-source qui pour nous représente un gage de stabilité et de liberté pour les développements de nos outils.

Vis-à-vis de l'existant, rappelons que MACAO propose un processus, des modèles et des documents. Nous avons cherché à préciser ses fondements théoriques en élargissant son cadre, pour cela nous avons défini une taxonomie des éléments, des rôles, et des documents principaux du processus. A partir de cette classification et du livre MACAO, nous avons rédigé un processus global simplifié qui définit les tâches majeures à réaliser et l'échange des documents majeurs de la démarche. Ensuite, grâce à l'expérience acquise en entreprise, nous avons répertorié l'ensemble des tâches des intervenants d'une équipe informatique. Nous avons alors constitué des fiches de description qui récapitulent les buts, les tâches, les documents entrants et les documents sortants pour l'application de la méthode MACAO. Cette base théorique a permis de mieux cerner MACAO dans son ensemble. L'étude d'opportunité nous a permis enfin d'entériner le choix de la technologie cible entre un outil avec une approche Web et un outil sous Eclipse.

Notre avons aussi été amenés à comparer les avantages et inconvénients des différents cycles de vie utilisés en entreprise. Nous avons pu ainsi établir une cartographie des cycles de vie adaptés avec le type de projets et de clients.

A partir d'expériences passées en entreprise, il semble intéressant de défendre l'élaboration de l'IHM dès les phases de conception. Les ateliers participatifs récoltent généralement les données et les traitements de la solution informatique à concevoir. Nous pensons qu'il y a un gain évident à capturer les besoins et les attentes des utilisateurs en terme d'IHM.

THESE-MACAO	Juin 2008	164 / 266
	Nicolas FERRY	

Nous avons collaboré au sein de notre équipe ISYCOM. Le fruit de ce travail a permis d'établir des passerelles entre les diagrammes de tâches et les diagrammes d'IHM. Les diagrammes de tâches étant par leur nature ceux qui conviennent le mieux pour décrire le domaine métier du client. Ce palier préliminaire ouvre des perspectives intéressantes pour une acquisition du métier avant de partir sur une étude abordant l'IHM, l'architecture, et la structuration des données.

Nous avons précisé le processus d'analyse global de la méthode MACAO pour prendre en compte cet échange avec les utilisateurs. Ce processus plus détaillé dans les phases d'analyse et de conception globales, aide au recueil, à la préparation et l'élaboration des modèles d'IHM.

Nous avons spécifié un découpage en trois niveaux de l'axe IHM pour concevoir des applications avec le niveau conceptuel, logique et physique. En soulignant l'intérêt de séparer le « Quoi-fonctionnel », le « Comment-Technologique » et le « Comment-Visuel » qui constituent les trois domaines et leurs frontières.

Notre contribution implique la définition d'un métamodèle qui est activement contributeur dans l'outil VisualSNI puisqu'il définit le format des fichiers SNI. Nous avons créé aussi les métamodèles théoriques pour le SEF, le SEP et les maquettes.

Cette répartition de l'axe de l'IHM, nous a amenés à prévoir le système de communication entre les différents niveaux. Nous utilisons les transformations de modèles comme celles appliquées sur les PMI et les projets de MDE pour transcrire d'un niveau à l'autre. Cette approche offre une grande souplesse et permet aussi d'envisager des échanges avec d'autres modèles de domaines différents. Cet interopérabilité nécessitera de construire des transformateurs adaptés entre les différents domaines.

Bien sûr un tel système de communication a des conséquences, notamment celle de prévoir l'émergence d'un nouveau rôle que j'appelle « architectes MDE » dont le rôle serait justement de créer et écrire les transformations de modèles. C'est un point important car s'il est admis en recherche, dans le monde de l'entreprise, je pense qu'il ne l'est pas. Les architectes réalisent des modèles mais n'ont pas encore conscience, je crois, que dans le procédé industriel il faille écrire les transformations entre ces modèles. C'est un facteur important dans l'industrialisation de l'entreprise.

Suite à l'évolution courante des IHM la tendance actuelle, nous fait penser que la séparation architecture de l'IHM et de la représentation du contenu graphique sera propice et c'est déjà plus ou moins le cas, à faire émerger le rôle de graphiste ou de designer en anglais. Si

THESE-MACAO	Juin 2008	165 / 266
	Nicolas FERRY	

Capgemini emploie déjà un certain nombre de graphistes sur certains projets, le monde de la recherche prend conscience de l'importance de ce rôle qui apporte un complément et de la souplesse face à la technique.

Concrètement notre contribution se manifeste par des modules additionnels pour la plateforme Eclipse. Le module « VisualSNI » permet de concevoir les Schémas Navigationnels des IHM de niveau conceptuel de la méthode. C'est un éditeur graphique complet au niveau des fonctionnalités requises primordiales.

Le second module constitue une partie du système communicatif qui permet de transformer un modèle SNI en une maquette informatique. Ce prototype fait la preuve du concept et permet de définir une première base pour générer des maquettes d'IHM en plusieurs technologies cibles depuis un diagramme. Cette génération directe d'un diagramme SNI conceptuel en maquette requiert de rajouter une énergie signifiante qui est contenue dans les patrons de génération.

Ce générateur ouvre un cadre d'utilisation progressif de la génération automatique dans un procédé industriel. Notamment nous pensons après réflexion avec l'entreprise que le générateur peut servir dans un premier temps, dans les phases d'avant-vente où des propositions maquettées sont soumises aux clients. Un générateur peut permettre de présenter rapidement une offre dans une technologie donnée. Dans un deuxième temps, il peut être utilisé dans les ateliers participatifs pour s'adapter à la demande et proposer des modifications du modèle et générer des maquettes quasiment sur l'instant. Enfin, il peut être adapté dans le processus de l'entreprise pour les transformations successives des modèles vers le code applicatif.

Les générateurs de codes automatiques permettent d'industrialiser le processus de création mais ils le rigidifient par la même occasion. Il ne faut pas oublier que la majorité des projets sont nouveaux et repartent de rien. L'utilité de ces générateurs provient du fait qu'il existe une certaine redondance au niveau des projets et c'est cette redondance que les patrons de générations peuvent encoder. La proportion de cette redondance dépend des projets mais dans l'absolu elle suit la règle des 80-20. Aussi ces 20% restant sont très difficilement supprimables à moins de faire des logiciels pour un marché de niche. Ce n'est certainement pas la majorité des projets d'une SSII comme Capgemini, ce qui veut dire qu'il faut garder une place importante aux personnes qui de toute façon doivent produire et gérer la partie renouvelée.

Pour les ateliers participatifs je précise que pour une architecture globale donnée, la définition de ses composantes architecturales sont des vues orthogonales dans un espace ayant autant de dimensions que de composantes par rapport au processus métier.

THESE-MACAO	Juin 2008	166 / 266
	Nicolas FERRY	

Le corollaire est de concevoir la programmation orientée aspect comme un moyen de faire fusionner les différentes composantes avec leurs modèles en un ou plusieurs ensembles de code informatique.

Glossaire des sigles

Sigles	Définition
ANRT	Association Nationale de la Recherche Technique.
CIFRE	Les Conventions Industrielles de Formation par la Recherche sont instruites et gérées par l' ANRT pour le compte du Ministère chargé de la Recherche.
DEV	Diagramme de niveau Logique. Les schémas de Définitions des Elements Visuels décrivent l'apparence des IHM. Ils sont comme les SE dépendants de la technologie employée. Et de la même façon, il faut distinguer les DEP, DEF et DEM. Ils sont très utilisés dans les étapes de maquettage en développement RAD.
DSL	Un langage dédié est créé pour résoudre certains problèmes spécifiques dans un domaine particulier, et n'a en principe pas vocation à résoudre des problèmes en dehors de ce contexte. En comparaison, on crée des langages généralistes pour résoudre des problèmes dans beaucoup de domaines. Les langages généralistes sont nécessairement "Turing Complete". Avantages : <ul style="list-style-type: none"> • Les langages dédiés permettent d'exprimer des solutions avec les tournures idiomatiques au niveau d'abstraction du domaine traité. En conséquence, les experts du domaine eux-mêmes peuvent comprendre, valider, modifier, et souvent même développer des programmes en langage dédié. • Les langages dédiés facilitent la documentation du code. • Les langages dédiés améliorent la qualité, la productivité, la fiabilité, la maintenabilité, la portabilité et les possibilités de réutilisation.

THESE-MACAO	Juin 2008	168 / 266
	Nicolas FERRY	

	<ul style="list-style-type: none">• Les langages dédiés permettent la validation au niveau du domaine. Aussi longtemps que les éléments du langage sont sûrs, toute phrase écrite avec ces éléments peut être considérée comme sûre.	
Eclipse	<p>Eclipse IDE est un environnement de développement intégré libre, extensible, universel et polyvalent.</p> <p>La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.</p>	
Ecore	Ecore est le métamétamodèle utilisé par EMF. C'est un sous-ensemble du MOF qui permet de définir et gérer des métamodèles.	
EMF	Eclipse Modeling Framework (EMF) est un Framework de modélisation et de simplification de génération de code pour la construction d'outils et d'autres applications basés sur une structure de modèle de données . EMF gère la partie Model du pattern MVC.	
GEF	Graphical Editing Framework est un environnement pour le développement de visuels graphiques (la partie View du MVC + un support de la partie Controller) avec abstraction totale du Model mais possibilité de le modifier via une infrastructure qui organise une serie d'implémentations du pattern 'Command'. Cette infrastructure est aussi utilisée pour le fonctionnement interne de GEF.	
IDM	L' IDM (Ingénierie dirigée par les modèles) est le domaine de l'informatique mettant à disposition des outils, concepts et langages pour créer et transformer des modèles .	
JSF	<p>Java Server Faces (abrégié en JSF) est un framework Java, pour les développeurs d'application Web, utilisant l'architecture J2EE.</p> <p>JSF a été mis au point par Craig McClanahan, le père de Struts</p>	
THESE-MACAO	Juin 2008	169 / 266
	Nicolas FERRY	

	<p>et de Catalina, le conteneur de servlet de Tomcat. La première version a vu le jour en 2003. JSF est un framework backend et frontend.</p> <p>Sur le frontend, il améliore les JavaServer Pages en apportant de nouvelles bibliothèques de composants (ajax4jsf, richfaces...). Les composants JSF sont comme des composants JSP, ils génèrent du HTML et du JavaScript.</p> <p>Sur le backend JSF apporte d'énormes améliorations, en fait JSF est basé sur le pattern MVC2 et apporte la notion d'inversion de contrôle (ou IOC), c'est-à-dire qu'il permet à certaines classes Java (appelées alors des JSF Managed Beans) de n'être créées que lorsque c'est nécessaire (par JSF). La navigation ne se fait plus de page en page par des URL mais de vue en vue par des événements.</p>
JSP	Le JavaServer Pages ou JSP est une technologie basée sur Java qui permet aux développeurs de générer dynamiquement du code HTML , XML ou tout autre type de page Web . La technologie permet au code Java et à certaines actions prédéfinies d'être ajoutés dans un contenu statique.
MACAO	La méthode d'analyse et de conception d'applications orientées objet (dite MACAO) est fondée sur une démarche participative par prototypage incrémental (processus itératif) permettant aux utilisateurs d'intervenir très tôt dans le processus de développement du logiciel .
MAQ	Maquette présentée à l'utilisateur.
MDA	L' architecture dirigée par les modèles ou MDA (pour l'Anglais Model Driven Architecture) est une démarche de réalisation de logiciel, proposée et soutenue par l'OMG. C'est une variante particulière de l' ingénierie dirigée par les modèles (IDM, ou MDE pour l'Anglais <i>Model Driven Engineering</i>). D'autres variantes de l'IDM ont été développées, par exemple par Microsoft (DSL Tools).

THESE-MACAO	Juin 2008	170 / 266
	Nicolas FERRY	

	Le principe de base du MDA [46] est l'élaboration de modèles indépendants de plates-formes (<i>Platform Independent Model</i> , PIM) et la transformation de ceux-ci en modèles dépendants de plates-formes (<i>Platform Specific Model</i> , PSM) pour l'implémentation
MDE	terme anglophone pour <i>Model Driven Engineering</i> pour Ingénierie dirigée par les modèles (voir IDM).
Merise	Merise est une méthode d'analyse, de conception et de gestion de projet complètement intégrée, ce qui en constitue le principal atout. Elle a fourni un cadre méthodologique et un langage commun et rigoureux à une génération d'informaticiens français.
Métamodèle	Un métamodèle peut être défini comme un modèle d'un langage de modélisation. Un métamodèle sert ainsi à exprimer les concepts communs à l'ensemble des modèles d'un même domaine.
Modèle	En informatique, un modèle a pour objectif de structurer les données, les traitements, et les flux d'informations entre entités.
MOF	<p>Le Meta-Object Facility (MOF) est un standard de l'OMG s'intéressant à la représentation des métamodèles et leur manipulation. Le langage MOF s'auto-définit.</p> <p>Le standard MOF est situé au sommet d'une architecture de modélisation en 4 couches:</p> <ul style="list-style-type: none"> • M3: le métamétamodèle MOF (couche auto descriptive) • M2: les métamodèles • M1: les modèles • M0: Le monde réel <p>Le langage UML est décrit par un métamodèle conforme au MOF. Ainsi un modèle UML peut être sérialisé en XMI [58] Mais il y a également de nombreux autres métamodèles situés au même niveau que UML. On peut citer par exemple les</p>

THESE-MACAO	Juin 2008	171 / 266
	Nicolas FERRY	

	<p>métamodèles CWM, SPEM, SysML, etc.</p> <p>UML est utilisé pour concevoir des applications objets dans différents domaines comme l'aéronautique [161].</p>
MVC	<p>Le Modèle Vue Contrôleur (MVC) est une architecture et une méthode de conception pour le développement d'applications logicielles qui sépare le modèle de données, l'interface utilisateur et la logique de contrôle. Cette méthode a été mise au point en 1979 par Trygve Reenskaug, qui travaillait alors sur Smalltalk dans les laboratoires de recherche Xerox PARC.</p>
oAW	<p>openArchitectureWare (oAW) est un framework de génération MDA/MDD conçu en Java. Il possède des modules pour l'analyse syntaxique de modèles, une famille de langages de vérification de contraintes et de transformations de modèles, et de génération de code.</p>
OCL	<p>OCL (Object Constraint Language) est un langage informatique d'expression des contraintes utilisé par UML. OCL [60] est une contribution d'IBM à UML 1.1. OCL permet de décrire des invariants dans un modèle, sous forme de pseudo-codes :</p> <ul style="list-style-type: none"> • pré et post-conditions pour une opération • expressions de navigation • expressions booléennes, etc... <p>OCL est utilisé dans la définition du métamodèle UML.</p>
RAD	<p>La méthode RAD, acronyme de <i>Rapid Application Development</i> (développement rapide en français) est une méthode de développement de logiciels où le cycle de développement est plus court que celui des méthodes Cascades. La méthode RAD fut initialement développée par James Martin pendant les années 1980. L'objectif est d'obtenir un applicatif adéquat à partir d'un prototypage impliquant l'utilisateur final.</p>
SE	<p>Diagramme de niveau Logique. Les Schémas d'enchaînement décrivent techniquement une dynamique d'IHM. Les objets</p>

	<p>sont spécifiques et les événements sont dépendants d'une technologie.</p> <p>En règle générale, il existe un type de diagramme par famille technologique :</p> <ul style="list-style-type: none"> – Clients légers à base de pages WEB seront postfixés par P – Clients lourds à base de fenêtre seront postfixés par F, – Applications Multimodales seront postfixées par M – etc. <p>Il faut donc distinguer les diagrammes SEP, SEF et SEM suivant la technologie. Si une nouvelle famille technologique apparaît, un nouveau type de diagramme sera créé.</p>
SNI	Diagramme de niveau Conceptuel, le Schéma Navigationnel des IHM modélise l'interaction homme-machine indépendamment d'une technologie.
TTUP	Two Tracks Unified Process est un processus proposé par Valtech (consulting), présenté dans le livre : Pascal Roques, Franck Vallée (2004) UML2 en action [86] Elle a pour objectif de prendre en compte les contraintes de changement continu imposées aux systèmes d'information des organisations. Elle est basée sur l'acquisition de deux composantes majeures pour gérer les évolutions fonctionnelles et les évolutions techniques.
UD	Unité de Dialogue. Élément de conception d'une IHM conceptuel.
UDC	Unité de Dialogue Composée. Assemblage d'unités de dialogue permettant de créer une structure d'IHM par composition d'éléments.
UDE	Unité de Dialogue Élémentaire. Brique de base pour la création d'une IHM conceptuelle.

THESE-MACAO	Juin 2008	173 / 266
	Nicolas FERRY	

UP	<p>Processus unifié (ou UP en anglais pour Unified Process) est une méthode de prise en charge du cycle de vie d'un logiciel et donc du développement, pour les logiciels orientés objets. C'est une méthode générique, itérative et incrémentale, contrairement à la méthode séquentielle Merise (ou SADT).</p> <p>UP vient compléter la systématique des modèles UML. Elle est le résultat final d'une évolution de l'approche d'Ericsson qui est au fondement d'une des premières méthodes de développement pour applications orientées objets, la méthode Objectory Process (1987). Objectory Process (version 1 à 3.8 en 1995) a elle-même servi de base à la société Rational pour la création de Rational Objectory Process (1997) (version 4.1), parente directe de RUP en 1998.</p>
----	---

THESE-MACAO	Juin 2008	174 / 266
	Nicolas FERRY	

Index des illustrations

<i>Figure 1 : Les étapes d'une approche en cascade.....</i>	<i>23</i>
<i>Figure 2 : Les étapes d'une approche en spirale.....</i>	<i>24</i>
<i>Figure 3 : Méthodes ascendantes.....</i>	<i>26</i>
<i>Figure 4 : Méthodes descendantes ou systémiques.....</i>	<i>27</i>
<i>Figure 5 : Les étapes du processus MACAO.....</i>	<i>41</i>
<i>Figure 6 : Etape de développement par prototypage successif.....</i>	<i>44</i>
<i>Figure 7 : Exemple de composition par juxtaposition.....</i>	<i>61</i>
<i>Figure 8 : Exemple de composition d'une UDC par groupage.....</i>	<i>61</i>
<i>Figure 9 : Exemple d'une IHM d'un concessionnaire de véhicule.....</i>	<i>65</i>
<i>Figure 10 : Représentation d'une fenêtre dans le SEF.....</i>	<i>69</i>
<i>Figure 11 : SEF de l'administration de la classe "Véhicule".....</i>	<i>69</i>
<i>Figure 12 : Maquette d'écran d'une saisie d'étudiant avec sa légende.....</i>	<i>71</i>
<i>Figure 13 : Maquette d'écran d'une page Web pour gérer des contrats</i>	<i>72</i>
<i>Figure 14 : Représentation d'une application multi-tiers en couches</i>	<i>88</i>
<i>Figure 15 : Les différents niveaux de l'axe IHM.....</i>	<i>89</i>
<i>Figure 16 : Architecture de l'axe IHM avec transformateurs.....</i>	<i>91</i>
<i>Figure 17 : Schéma des principaux cas d'utilisation du rôle Analyste.....</i>	<i>96</i>
<i>Figure 18 : Schéma des principaux cas d'utilisation d'un Architecte.....</i>	<i>97</i>
<i>Figure 19 : Schéma des principaux cas d'utilisation du rôle Développeur.....</i>	<i>98</i>
<i>Figure 20 : Insertion de VisualSNI dans Eclipse.....</i>	<i>102</i>
<i>Figure 21 : La palette d'édition permet de créer les objets du SNI.....</i>	<i>103</i>
<i>Figure 22 : Construction de l'écran de login.....</i>	<i>105</i>
<i>Figure 23: Exemple de construction de l'application « Sicli ».....</i>	<i>106</i>
<i>Figure 24 : Exemple de construction de l'application «Sicli».....</i>	<i>107</i>
<i>Figure 25 : Planche contenant les consignes.....</i>	<i>108</i>
<i>Figure 26 : Exemple de fichier XMI d'une planche SNI.....</i>	<i>109</i>
<i>Figure 27 : Ce schéma montre les pyramides des niveaux de modélisation.....</i>	<i>111</i>
<i>Figure 28 : Métamodèle du SNI - Paquetage Gestion.....</i>	<i>113</i>
<i>Figure 29 : Métamodèle du SNI - Paquetage Répartition.....</i>	<i>116</i>
<i>Figure 30 : Métamodèle du SNI - Paquetage UDE.....</i>	<i>118</i>
<i>Figure 31 : Métamodèle du SNI - Paquetage Nodes.....</i>	<i>120</i>
<i>Figure 32 : Structure en modules d'Eclipse.....</i>	<i>123</i>
<i>Figure 33 : Structure adoptée par tous les plugins Eclipse.....</i>	<i>124</i>
<i>Figure 34 : Le pattern MVC a été repris dans une version légèrement modifiée.....</i>	<i>127</i>
<i>Figure 35 : Hiérarchie des éléments du métamétamodèle ECORE.....</i>	<i>128</i>
<i>Figure 36 : Métamétamodèle ECORE.....</i>	<i>129</i>

THESE-MACAO	Juin 2008	175 / 266
	Nicolas FERRY	

Figure 37 : La génération des classes du modèle pour l'éditeur.....	130
Figure 38 : le pattern MVC de VisualSNI.....	130
Figure 39 : Description générale de MVC faite par GEF.....	131
Figure 40 : Figure montrant le cheminement des requêtes.....	132
Figure 41 : Figure des couches des API graphiques.....	134
Figure 42 : Figure montrant la structuration graphique des figures du SNI.....	135
Figure 43 : SNI de l'application INRA pour le scénario nominal de test défini.....	140
Figure 44 : Architecture des couches VisualSNI et du générateur de maquette.....	142
Figure 45 : Un transformateur à base de templates.....	144
Figure 46 : L'automate de génération de oAW.....	146
Figure 47 : Création d'un plugin de génération pour visualSNI.....	147
Figure 48 : Fichiers générés directement dans l'application Web.....	148
Figure 49 : Exemple de Page JSF affichée par le navigateur.....	149
Figure 50 : Exemple d'une page Web JSF typiquement utilisée.....	150
Figure 51 : Un template pour chaque type d'UD va définir la transformation à opérer....	151

Index des tableaux

Tableau 1 : Tableau des interviews menées auprès de chefs de projets.....	34
Tableau 2 : Documentation produite à chaque étape du processus.....	47
Tableau 3 : Chronologie d'utilisation des diagrammes d'UML + MACAO.....	48
Tableau 4 : Les schémas de la méthode MACAO.....	90

THESE-MACAO	Juin 2008	177 / 266
	Nicolas FERRY	

Partie E - ANNEXES

THESE-MACAO	Juin 2008	178 / 266
	Nicolas FERRY	

INTRODUCTION

Les annexes suivantes traduisent l'étude de l'atelier et du processus MACAO.

L'étude commence par un sondage trouvé sur le forum du site Developpez.com. Ce sondage montre, en 2004, les pourcentages d'utilisation du langage UML dans les entreprises. On trouve aussi les logiciels les plus utilisés pour modéliser UML et les bases de données.

Dans l'annexe 2, nous faisons une critique par rapport à l'existant. A Capgemini Sud, le profil suit cette lignée puisqu'à l'ADC la communauté des architectes utilise principalement l'outil Rational Rose pour certains projets objets. Cette critique fait aussi remonter les commandes majeures qu'il est nécessaire de fournir avec MACAO. Ceci afin de pouvoir déterminer une plate-forme opérationnelle.

Dans l'annexe 3, nous fournissons la spécification de l'étage de génération des IHM telle qu'imaginée au début de la phase d'analyse. Cette spécification a été précurseur dans le système de génération présenté dans cette thèse.

A la demande de François Tricot, l'annexe 4 décrit l'étude de la cible et l'insertion de l'outil MACAO dans son contexte et face aux outils industriels concurrents.

L'annexe 5 est la décomposition du processus MACAO afin qu'il soit informatisable dans l'atelier de génie logiciel. L'étude décrit les rôles, les étapes, les documents à produire, pour mettre en oeuvre la méthode MACAO sur un projet.

L'annexe 6 est une étude de faisabilité réalisée afin de permettre de tester deux technologies d'implémentation à savoir le Web et Eclipse.

L'annexe 7 référence les cas d'utilisation d'une équipe MACAO, elle classe les actions principales de chaque intervenant.

L'annexe 8 montre les cas d'utilisation concernés pour l'élaboration d'une IHM d'une application.

Enfin, l'annexe 9 est une ébauche des paquetages et éléments présents pour définir dans sa globalité l'atelier de génie logiciel MACAO.

THESE-MACAO	Juin 2008	179 / 266
	Nicolas FERRY	

Chapitre I : Sondages du forum Développez :

J'utilise UML :

Pour certains projets : processus partiel	24% [35]
Pour tous mes projets : processus partiel	18% [27]
Pour tous mes projets et de A à Z	11% [17]
C'est quoi UML ?	10% [15]
Rarement	8% [13]
Jamais	6% [10]
Pour certains projet et de A à Z	5% [8]
Jamais UML, par contre parfois MERISE	4% [6]
Sans opinion	4% [6]
Jamais UML, par contre toujours MERISE	3% [5]
Jamais UML, autre (précisez)	2% [3]

Total : 145 votants

Quel outil de modélisation UML utilisez -vous ?

[IBM] Rational Rose / XDE	33% [35]
[Gentleware] Poseidon UML	14% [15]
[Borland] Together / ModelMaker	11% [12]
[Eclipse] Plugin Omondo	8% [9]
[Microsoft] Visio	8% [9]
[Sybase] PowerAMC/PowerDesigner	7% [8]
[Objecteering Software] Objecteering/UML	5% [6]
ArgoUML	5% [6]
[I-Logix] Rhapsody Modeler	0% [1]
[Télélogic] TAU UML / TAU GII	0% [1]
Umbrello UML Modeller	1% [2]
ClassBuilder	0% [1]

Total : 105 Votants

Quel outil utilisez- vous pour concevoir vos bases de données ?

[Sybase] PowerAMC/PowerDesigner	32% [115]
Aucun pour l'instant	18% [66]
Autres ?	12% [44]
[fabforce.net] DBDesigner	12% [42]
[Cecima] Win'Design	11% [39]
[Microsoft] Visio	6% [23]
[GNU] Dia	3% [12]
[Charonware] CaseStudio	2% [7]
[Embarcadero] ERStudio	0% [1]
[Devaki] NextObjects	0% [0]

Total : 349 votants

Date du Sondage : début 2004.

THESE-MACAO	Juin 2008	180 / 266
	Nicolas FERRY	

Chapitre II : Critiques et suggestions

II.1.1 - Outils informatiques actuels

A Capgemini Sud, les Techniciens sont actuellement équipés de plates-formes de développement Eclipse Java ou environnement Microsoft. Ces environnements sont parfois complétés par des ateliers d'architecture comme Rational Rose.

Nous souhaitons inscrire dans l'environnement actuel des développeurs, un atelier de génie logiciel complet intégrant des modules de conception d'IHM avancée. Cet environnement doit devenir un support commun pour l'équipe en charge des développements.

II.1.2 - Critiques

Dans la réalisation d'applications informatiques, bon nombre d'outils existent sur le marché mais peu associent dans un environnement intégré la conception d'IHM jusqu'à la génération de code pour une équipe industrielle.

L'atelier de génie logiciel MACAO doit permettre la mise en application de la démarche dans un environnement stable et efficace pour l'équipe projet. Du fait que MACAO utilise des concepts novateurs, il faut les intégrer de manière transparente pour les équipes.

Par exemple, dans le cadre de génération des prototypes : la règle de non-régression, il est important d'assurer que les sources de prototypes précédents validés n'ont pas été modifiées par les développeurs afin de garantir une intégrité des données.

Il faut aussi que les intervenants puissent déclencher facilement l'envoi des documents spéciaux comme les DMP (Demandes de Modification Prototype) ou FAP (Fiches d'Anomalies Prototype). Ceci nécessite d'intégrer directement ces commandes et mécanismes dans l'atelier.

Dans les pages suivantes, nous présentons l'analyse de l'atelier de génie logiciel MACAO qui intègre le module de génération IHM qui a été détaillée dans cette thèse ainsi que l'étude du processus de la démarche MACAO en vue de son informatisation. L'analyse globale concerne les acteurs et les tâches typiques pour une équipe projet.

THESE-MACAO	Juin 2008	181 / 266
	Nicolas FERRY	

Chapitre III : Spécification du module d'IHM

Le module MACAO pour l'IHM propose de créer des modèles d'IHM dans un projet donné. Le module gère un niveau conceptuel, un niveau logique, et un niveau physique.

Pour le niveau conceptuel, le modèle du SNI est utilisé.

Pour le niveau logique, le modèle du SEF ou SEP ou SEM est utilisé.

Pour le niveau physique, le modèle MAQ est utilisé.

Chaque modèle est conçu lui-même sur un métamodèle. Comme les diagrammes d'IHM de MACAO sont hérités d'un diagramme d'activité (par conséquent d'états-transitions) : ils utilisent sous Eclipse le métamodèle d'UML2.0 (basé sur le Ecore d'EMF). Il est possible de passer du Ecore (Eclipse) à MOF (OMG) et vice-versa.

Le métamodèle du SNI hérite de UML2.0.

Les métamodèles du SEF/SEP/SEM héritent du SNI

Le métamodèle de la maquette MAQ hérite de ceux du niveau logique.

Ainsi tous les métamodèles sont interdépendants avec une relation de généralisation / spécialisation suivant le niveau d'abstraction.

Tout modèle construit dans un projet Eclipse aura un fichier lié à ce modèle.

Par exemple dans un **paquetage achat** :

Le modèle du SNI sera stocké dans ihm-achat.sni

Le modèle du SEF sera stocké dans ihm-achat.sef

Le modèle du MAQ sera stocké dans ihm-achat.maq

Enfin les modules de codes seront créés dans un sous paquetage.

Les modèles sont aussi portables sous la norme XMI 2.0 (pour UML 2.0) par la fonction d'import / export du projet Java.

Les différents modèles sont implémentés dans le plug-in MACAO comme des profils UML, ce format est standardisé et les fichiers XMI peuvent être réutilisés avec d'autres outils qui supportent XMI et les profils (du standard UML 2.0).

THESE-MACAO	Juin 2008	182 / 266
	Nicolas FERRY	

Chapitre IV : Etude de la cible

E.IV.1 - Etude des domaines existants

Cette étude des différents domaines, faite à la demande de Capgemini, sera plutôt orientée industrielle que recherche, si bien que beaucoup de travaux ou standard de recherche ne seront pas mentionnés ici. Il est important aussi de mentionner que cette étude a été réalisée en début de thèse et que les outils cités ont évolué considérablement durant ces trois ans. Enfin, cette étude malgré la rigueur déployée ne peut prétendre être exhaustive du fait du nombre considérable de solutions existantes.

Pour analyser l'industrialisation du processus de création d'applications informatiques, des macro domaines sont définis afin de pouvoir répertorier l'ensemble des outils. Chaque domaine contient un ensemble de technologies utilisées dans l'industrie pour la création d'applications impactant sur l'IHM.

Liste des macro domaines de la classification :

- Les méthodologies
- Les plates-formes exécutives
- Les environnements de développement (IDE)
- Les outils de génie logiciel pour modéliser (AGL)
- Les technologies des clients riches

Les forces en présence du point de vue technique :

Les forces ne sont probablement pas exhaustives.

Les méthodologies :

- Merise [83]
- Unified Process [84]
- XP [85]
- TTUP [86]
- Autres [www.businessprocesspartner.com/methodes/index]

- WebML [87]
- SWITCH-technology [is.ifmo.ru/English]
- UMLi [www.cs.man.ac.uk/img/umli/]
- MACAO [www.iut-blagnac.fr/MACAO/PageMACAO.html]

THESE-MACAO	Juin 2008	183 / 266
	Nicolas FERRY	

Plates-formes et langage support d'exécution de programmes :

Pour les langages compilés :

- Microsoft Win32 API [www.iseran.com/Win32/FAQ/faq.htm][126]
- Les systèmes Unix [89]

Pour les langages managés :

- La plate-forme Microsoft .Net [90]
- La plate-forme Java [91]

Ici, nous ne prenons pas en compte les langages pour des plates-formes de programmation très spécifiques, mais bien sûr, toute API qui interprète ou exécute un langage compilé est une plate-forme d'exécution. Nous passons donc sous silence beaucoup de plates-formes tiers.

Les environnements de développements :

- Microsoft Visual Studio [92]
- IBM WebSphere / Eclipse [93]
- BEA Weblogic [94]
- Borland Suite [95]
- JetBrains IntelliJ [96]
- SUN Java Studio [97]
- Exadel STRUTS Studio [98]
- PcSoft WinDev / WebDev [99]
- etc...

En général, si la technologie est bien avancée, elle doit posséder un environnement de développement dédié. C'est pour cette raison que la liste restera volontairement limitée.

Les AGL pour modéliser :

- Borland Together [100]
- IBM Rational Atlantic sur Eclipse [101]
- SoftTeam Objecteering [www.objecteering.com]
- Entreprise Architect [102]
- Omondo Eclipse UML Studio [103]
- Gentleware Poseidon for UML [104]
- ArgoUML [105]
- I-Logix Rhapsody Modeler [106]

THESE-MACAO	Juin 2008	184 / 266
	Nicolas FERRY	

- Envision Case France [107]
- Sybase PowerAMC [108]
- Telelogic TAU [109]

De nombreux autres AGL existent mais sont peu ou pas assez représentatifs.

Les technologies pour le client léger :

- HTML
- JAVA : JSP, JSF
- AJAX (Web 2.0) [78]
- PHP
- Ruby On Rails [110]
- etc...

Les technologies pour le client riche (RC) :

Ce sont les nouvelles techniques permettant de faire de la présentation Web plus évoluée.

- Microsoft .Net “One Click Deployment”
- Java Web Start [112]
- Adobe Acrobat Designer [113]
- Adobe Macromedia Flex [78][114]
- W3C XForm [115]
- SWT [116]
- XUL [117][118]
- XAML [119]
- Laszlo (LZX) [120][121]
- Eclipse RCP [122][73]

Une vision de l'évolution du client riche est décrite dans un article de ZDNet [123].

Librairie MSDN : [125]

Eclipse SWT :

www.eclipse.org/articles/Article-SWT-Design-1/SWT-Design-1.html

Nous allons examiner dans un premier temps, les plates-formes exécutives qui permettent soit de s'abstraire du système d'exploitation soit de la dépendance liée aux

THESE-MACAO	Juin 2008	185 / 266
	Nicolas FERRY	

langages de programmation. Les deux grosses plates-formes sont inévitablement Microsoft .Net avec le Common Langage Runtime (CLR) et Java de Sun Microsystem avec la Java Virtual Machine (JVM).

Java a été créée dans une volonté de portabilité, on se souvient du slogan : « écrire une fois, exécuter partout ». La JVM est conçue pour être utilisée indépendamment de la machine et du système d'exploitation. De l'autre côté le framework .Net de Microsoft permet d'unifier les différentes castes de développeurs qui ont des compétences de programmation hétéroclites.

Actuellement, ces deux plates-formes concentrent une part importante des développements globaux sur micro ordinateurs mais il ne faut pas oublier les développements ayant trait aux ERP, et aux services annexes.

Examinons maintenant plus en détail ce qu'offrent ces deux plates-formes en les comparant d'un point de vue technologique bien sûr et en terme de leur perspective d'évolution.

E.IV.2 - La plate-forme Microsoft .Net

IV.2.1 - Présentation du Framework

Microsoft .Net fut présenté par Bill Gates en Juillet 2000 lors du Congrès des développeurs professionnels d'Orlando, en Floride. La plupart de ceux qui ont eu un avant goût de Microsoft .NET ont senti qu'il s'agissait d'un tournant dans le monde des logiciels pour les plates-formes Microsoft. La première version 1.0 du framework .NET a été rendue publique le 15 janvier 2002.

Voici comment Microsoft définit .NET :

" .NET est la plate-forme Microsoft pour la nouvelle génération de logiciels distribués et coopérants : les services Web XML. Elle vise à simplifier la vie du développeur en lui fournissant des services intégrés, centrés sur lui, accessibles depuis tous ses périphériques, à tout moment et en tout lieu. S'il ne fallait retenir qu'un seul mot de .NET, c'est donc l'intégration.

Fondée sur des standards de l'industrie (http, XML, SOAP, WSDL), la plate-forme .NET est un moyen simple de normaliser la coopération des services logiciels entre eux (services Web XML), quelle que soit leur localisation, leur implémentation technique, qu'ils soient internes ou externes. "

THESE-MACAO	Juin 2008	186 / 266
	Nicolas FERRY	

Le Framework .NET permet le développement d'applications dans l'environnement fortement distribué d'Internet. Le Framework .NET est conçu pour fournir un environnement cohérent, sécurisé, performant, supportant les applis standards ou Web.

Le Framework .NET contient deux composants principaux :

- La CLS : Common Language Specification
- la CLR : Common Language Runtime
- la bibliothèque de classes du Framework .NET.

Le Common Language Runtime est la base du Framework .NET. Le runtime peut être considéré comme un agent qui manage le code au moment de l'exécution, fournit des services essentiels comme la gestion de la mémoire, la gestion des threads, le « just in time » et l'accès distant. Il applique également une stricte sécurité des types et d'autres formes d'exactitude du code qui garantissent un code sécurisé et robuste. En fait, le concept de gestion de code est un principe fondamental du runtime. Le code qui cible le runtime porte le nom de code managé (par opposition au code non managé).

La bibliothèque de classes, l'autre composant principal du Framework .NET, est une collection complète orientée objet, de types réutilisables qui peuvent être utilisés pour développer des applications allant des traditionnelles applications à ligne de commande ou à interface graphique utilisateur (GUI, Graphical User Interface) jusqu'à des applications qui exploitent les dernières innovations fournies pour le net comme ASP.NET, les services Web XML et les Web Forms.

IV.2.2 - La vision de Microsoft

D'après une interview de Marc Gardette : responsable de la division architecture de Microsoft
Lien interview [<http://dotnetguru.org/articles/interview/marcgardette/interview.htm>]

« La plate-forme .Net témoigne d'une bonne santé, avec plus de 70 millions de systèmes déployés avec le framework .NET dont 50% de cette couverture touchent les 'Fortunes 100'. Pré-installé sur 60% des nouveaux systèmes et plus de 20 Millions de téléchargements avec Windows Update, plus de 450 livres disponibles, plus de 250 sociétés offrent des formations, et plus de 2,5 Millions de développeurs équipés avec Visual Studio .NET ».

On remarque la stratégie classique de déploiement de Microsoft qui est de fournir la plate-forme .NET gratuitement et pré-installée avec son système d'exploitation Windows. La

THESE-MACAO	Juin 2008	187 / 266
	Nicolas FERRY	

diffusion de la plate-forme .Net est donc acquise et sera largement répandue. Il faut noter aussi que la technologie .NET va ‘faire vivre’ de nombreuses sociétés tierces.

« [...] En France, selon une enquête qui a été menée en février dernier par un organisme indépendant, l’adoption de la plate-forme .NET progresse significativement et la répartition sur les 6 derniers mois donne le résultat suivant :
 .Net avec 22% et J2EE avec 19%.

La part des développements J2EE dans les grandes entreprises de plus de 1000 employés restant légèrement au-dessus de celle de .NET, mais la tendance s’inverse avec les PME. Selon notre dernière grande enquête Dev-Tracker, la part des développeurs .NET serait de 37% pour .NET contre 30% pour Java sur le continent Nord Américains. »

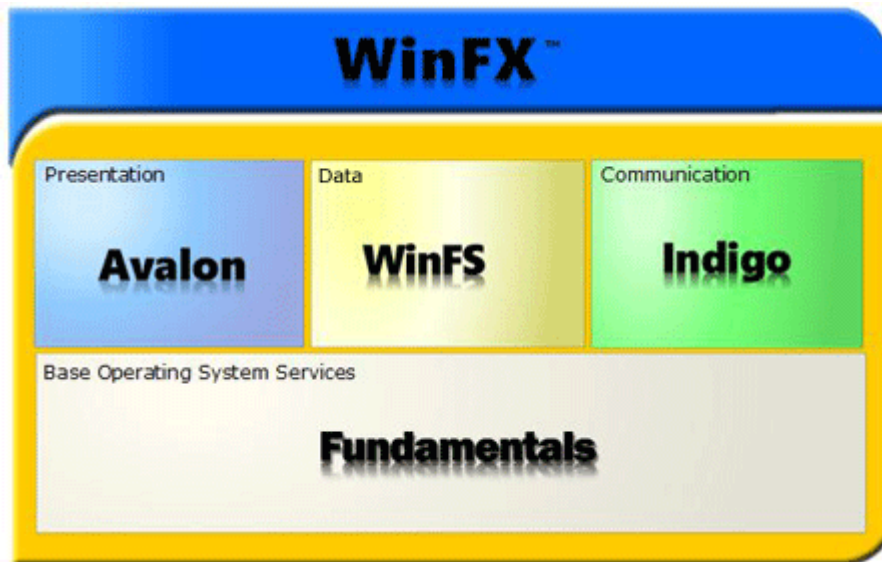
Cette interview de Marc Gardette de chez Microsoft prend bien sûr position dans les chiffres. Cependant, il me semble intéressant de retenir que : .Net et Java sont à l’heure actuelle sensiblement proches en terme de choix d’architecture du système d’information par les entreprises. Je pense qu’il faut aussi noter la volonté de Microsoft de vouloir imposer sa plate-forme face à JAVA...

IV.2.3 - Evolution de Windows

L’évolution du côté du système d’exploitation de Microsoft sera portée par Longhorn : le prochain Windows qui sortira en 2007. Celui-ci apportera son lot de nouveautés avec notamment la plate-forme .NET intégrée et son l’API-managed WinFX.

D’une façon plus synthétique, il est intéressant de noter que la structure de WinFX est mappée directement sur l’architecture standard des clients/serveurs :

THESE-MACAO	Juin 2008	188 / 266
	Nicolas FERRY	



L'interface de programmation fournit :

- une couche commune **Fundamentals** pour les opérations de bases du système
- une couche pour la présentation : **Avalon**
- une couche pour les communications : **Indigo** (SOA architecture)
- une couche pour la gestion donnée : **WinFS**.

Pour plus d'informations sur Longhorn : [<http://aumha.org/win5/a/longhorn.php>]

Remarque : un descriptif du mappage des fonctions .Net et Win32 est disponible ici : [<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/win32map.asp>]

L'interface **Avalon** confèrera aux utilisateurs un environnement 3D de leur bureau et de leurs applications [156]. Cette philosophie va changer beaucoup de choses en ce qui concerne l'interface graphique utilisateur. Je pense qu'il serait utile de prévoir dès à présent le support pour ce type de conception. Du point de vue du SNI, celui-ci reste valable en l'état, la description des écrans pourra elle être amenée à changer avec les niveaux logiques.

Pour utiliser Avalon, on peut écrire des applications WinForms classiques avec des langages supportés par .NET comme par le passé. Cependant, Avalon introduit également un

THESE-MACAO	Juin 2008	189 / 266
	Nicolas FERRY	

nouveau langage utilisable sous Longhorn : le **Xaml** – **EX**tensible **A**pplication **M**arkup **L**anguage.

Le XAML (prononcé Zammel) ressemble beaucoup à l'HTML. Il est très facile de créer des zones de textes, des boutons ou une image d'une façon très similaire. Ce qui est intéressant, c'est que le même code Xaml peut être exécuté dans Internet Explorer et affiché par le navigateur mais peut également être directement compilé pour tourner sous forme d'application Windows. Il est aussi possible de faire communiquer une application Windows avec une application XAML séparée. Ceci a pour but de pouvoir séparer le côté visuel de l'application du code en lui-même.

L'idée est de concevoir entièrement une application en XAML puis de rajouter le code en C#. Cela permet d'envisager la possibilité de porter très rapidement une application Windows en application Web et vice versa, le visuel étant indépendant du moteur.

Cette architecture n'est pas sans rappeler celle de l'ASP.Net avec son 'code behind'.

IV.2.4 - Evolution de Visual Studio 2005

Toujours selon l'interview de Marc Gardette de Microsoft.

Lien de l'interview :

[<http://dotnetguru.org/articles/interview/marcgardette/interview.htm>]

Vers le 1^{er} semestre 2005 devrait sortir la succession de sa plate-forme de développement : Visual Studio 2005.

Point important, Microsoft semble s'écarter du standard UML et notamment de la démarche MDA (Model Driven Architecture). Microsoft recommande l'utilisation Visio pour faire des schémas UML ou renvoie à des fournisseurs tiers comme Borland ou Rational.

En ce qui concerne Visual Studio 2005 : celui-ci possèdera une couche dédiée à la modélisation nommée WhiteHorse. Cette couche fera usage de modèles spécifiques à certains domaines (appelés aussi perspectives).

On retrouve trois outils intéressants dans Visual Studio 2005 puisque applicables dans la théorie des modèles :

- Le class designer

THESE-MACAO	Juin 2008	190 / 266
	Nicolas FERRY	

- Le WebService designer
- Logical Datacenter Designer (Deployment tool)

IV.2.5 - Vision du futur par Microsoft

Pour Microsoft le futur s'oriente vers un langage ou une description de plus haut niveau : le business process.

Les modèles devront s'appliquer au réel avec un lien technologie – modèle spécifique. Le summum ultime consisterait à ce que l'utilisateur puisse générer ses propres modèles qui répondent le mieux à ses besoins.

E.IV.3 - La plate-forme JAVA

IV.3.1 - Présentation

La plate-forme JAVA s'appuie avant tout sur le langage objet Java qui a été mis au point en 1991 par la société Sun Microsystems. James Gosling est considéré désormais comme le père de Java. Il décida de créer un langage orienté objet reprenant les caractéristiques principales du C++ mais en le simplifiant. Java se veut comme la plate-forme idéale de développement pour les réseaux informatiques. Indépendant et inter opérable entre plates-formes, il convient aux développements de services sur le Net. La plate-forme JAVA bénéficie d'une communauté massive de développeurs. Son maintien donc l'évolution de la plate-forme, est assuré par la Java Community Process (JCP) qui standardise et valide les évolutions.

IV.3.2 - Description Technique de JAVA

La spécification J2EE quatre types de composants d'application :

- Les **applications** clientes qui sont programmées en langage **Java** sont généralement des applications avec IHM qui sont prévues pour s'exécuter sur le poste client. C'est le type adapté pour faire des applications qui ont accès au 'middle tiers'.
- Les **Applets** sont des composants graphiques qui sont typiquement exécutés dans un navigateur Internet. Les Applets peuvent être utilisés pour faire des interfaces utilisateurs pour d'autres applications J2EE.

THESE-MACAO	Juin 2008	191 / 266
	Nicolas FERRY	

- Les **Servlets**, les pages **JSP**, les filtres et les événements Web sont exécutés dans un conteneur WEB et peuvent répondre à des requêtes HTTP de client Web. Ils permettent de générer dynamiquement des pages HTML pour faire de la présentation. L'échange de données peut se faire en XML entre différents composants. Un type spécial de composant gère le support des **WebServices**.
- Les **EJB** sont des composants exécutés dans un environnement managé qui supporte les transactions. Les Entreprises **Beans** contiennent la logique métier des applications J2SE dans une architecture n-tiers. Les EJB peuvent fournir directement des services Web.

L'évolution de Java me semble portée par la plate-forme Websphere et plus particulièrement Eclipse qui connaît un véritable engouement de la part de la communauté.

E.IV.4 - La plate-forme WebSphere d'IBM

IV.4.1 - Présentation d'Eclipse

La plate-forme Eclipse a été créée en 2001 par IBM [43] avec un financement de 40 Millions de dollars, Eclipse a constitué un des plus importants projets Open Source. IBM a laissé totale autonomie à l'Eclipse Fondation qui gère désormais le projet avec Mike Milinkovitch l'ancien président d'Oracle.

Eclipse bénéficie du soutien d'une cinquantaine d'acteurs dont HP, Intel, Ericsson, MontaVista Software, QNX, SAP, ...

Avec plus de 39 millions de téléchargements, Eclipse est certainement un des projets open source les plus actifs.

Eclipse est une plate-forme ouverte à l'intégration d'outils construits par une communauté de fournisseurs de plugin. Sous licence open source, avec une licence publique commune qui préconise de fournir le code source gratuitement, la plate-forme Eclipse fournit aux développeurs des outils pour qu'ils gardent une grande souplesse et un contrôle pointu de leur technologie.

Les outils Eclipse permettent aux développeurs de choisir entre un environnement multi langages, multi plates-formes, et multi vendeurs. Eclipse est composée d'un noyau et fournit une interface commune pour utiliser des plugins qui facilitent la création, l'intégration, et l'utilisation de programmes économisant du temps et de l'argent.

THESE-MACAO	Juin 2008	192 / 266
	Nicolas FERRY	

IV.4.2 - Evolution d'Eclipse

Lien internet : [127]

IV.4.3 - Plugins

Avec le rachat de Rational par IBM en début d'année (2004), IBM va proposer une offre permettant de couvrir la totalité du cycle de vie. Les outils de Rational vont être intégrés sur la plate-forme Eclipse sous le nom de code 'Atlantic'. Je pense que c'est une très bonne chose car IBM a beaucoup investi dans Eclipse, ainsi on devrait obtenir le support d'un outil de conception/développement tout intégré performant. Atlantic va couvrir la totalité du cycle de développement et sera proposé par modules payants adaptés à chaque phase. Techniquement, Atlantic pour la modélisation sera basé sur standard UML 2.0

E.IV.5 - Les autres plates-formes disponibles

IV.5.1 - Borland

Borland prépare beaucoup de changements dans sa gamme de produits. En effet, avec l'essor de la plate-forme .NET et du code managé, Borland migre son environnement IDE phare : Delphi pour supporter la plate-forme .NET. C'est un changement majeur puisque tous les compilateurs spécifiques de fournisseurs tiers sont amenés à être remplacés par .NET.

Borland fournit des outils qui couvrent tout le cycle de développement. On retrouve d'ailleurs l'outil Together pour la phase de conception. Celui-ci est aussi disponible sous Eclipse en tant que plugin. Il est intéressant de noter qu'une version 'community edition' existe pour Eclipse.

IV.5.2 - BEA Weblogic

BEA Weblogic est une plate-forme intégrée axée sur le développement d'architecture orienté services (SOA). Le côté spécifique de l'outil BEA en fait une plate-forme performante. Cependant François DEZA de Weblogic France, nous a précisé que BEA ne supporte pas pour l'instant d'aspect modélisation/conception au sens d'UML. D'après lui les développeurs du continent Américain ne font pas ou peu d'analyse pour la création de services sur le Web.

Weblogic possède cependant des vues différentes pour la conception :

- Le 'PageFlow view' qui permet de modéliser les enchaînements de pages Web
- Le 'Action view' représente graphiquement les entrées/sorties des contrôles
- Le 'Source view' qui est la vue sur le code des pages Web.

THESE-MACAO	Juin 2008	193 / 266
	Nicolas FERRY	

Les modifications portées sur une vue sont répercutées dans tout le système ce qui est une synchronisation du modèle avec le code.

IV.5.3 - Autres fournisseurs

D'autres outils existent pour répondre à des besoins spécifiques. On distingue plusieurs catégories :

- Les IDE pour l'aide au développement.
- Les serveurs d'applications intégrées

C'est le cas pour le développement JAVA des IDE comme :

- Forte promu initialement par Sun
- IntelliJ qui est très apprécié par notre équipe sur le projet CMH. (CMH est un projet interne à Capgemini)
- Etc.

En matière de serveurs d'applications sont souvent utilisés :

- BEA Weblogic, Tomcat, IPlanet, Web methods
- Oracle application server, Etc.

Outils existants GUI

E.IV.6 - Modélisation

A l'heure actuelle, il existe une spécification pour les interfaces Web qui s'appellent WebML [128].

Elle fournit des graphiques, des spécifications pour exprimer la structure d'une application Web. Elle est décrite comme :

“WebML provides graphical, yet formal, specifications, embodied in a complete design process, which can be assisted by visual design tools. The main objectives of the WebML design process are:

- (a) expressing the structure of a Web application with a high-level description, which can be used for querying, evolution, and maintenance;*
- (b) providing multiple views of the same content;*
- (c) separating the information content from its composition into pages, navigation, and presentation, which can be defined and evolved independently;*
- (d) storing the meta-information collected during the design process within a repository, which can be used during the lifetime of the application for dynamically generating Web pages*

THESE-MACAO	Juin 2008	194 / 266
	Nicolas FERRY	

- (e) *modelling users and communities explicitly in the repository, to permit the specification of personalization policies and one-to-one applications*
- (f) *enabling the specification of data manipulation operations for updating the site content or interacting with arbitrary external services.*”

On peut noter que WebML possède un outil d’aide à la création de ces graphiques.

E.IV.7 - AGL

Durant mes recherches en cours, je n’ai pas référencé de logiciel permettant la création d’interface homme-machine à partir d’un modèle conceptuel jusqu’à son implantation physique. Un outil comme un AGL MACAO serait alors une voie assez novatrice.

D’après une interview menée par mail avec Philippe DESFRAY de SoftTeam France [81], voici ce qu’il a répondu à la question :

- Que pensez-vous de la modélisation conceptuelle des IHM ?

« [Desfray] c’est sûrement une bonne idée. Il ne faut pas oublier que les IHM se font beaucoup par design graphique, maquettage, et que cette idée a du mal à passer en pratique. »

THESE-MACAO	Juin 2008	195 / 266
	Nicolas FERRY	

E.IV.8 - Tableau récapitulatif des outils de modélisation du marché

Il existe plus de 250 produits liés à UML. Voici la liste des outils 'sérieux' qui sont disponibles pour réaliser une modélisation avec UML. Ils sont répertoriés par nom de la société, coût du logiciel, version d'UML supportée, existence d'un plugin pour la plate-forme Eclipse et langages générés.

	Société	Nom du produit	Coût	UML	Eclipse	Gen. Code
1	IBM	Rational Rose Atlantic	Imminent	UML2,0		?
A	IBM	Rational Rose (version standard)	Payant	UML1,1 ?	E3,0	Java, C,...
X	IBM	Rational Developer Application	Payant			?
X	IBM	Rational Rose XDE	Payant	UML1,4 ?		Java, C,...
1	SparxSystem	Enterprise Architect	Payant	UML2.0	E /.NET	Java, C#, ...
1	Borland	Together Control Center	Payant	UML2,0		Java, C#,...
		Together Community edition for Eclipse	<u>Gratuit</u>	UML2,0	E3,0	Java
1	SoftTeam	Objetcteeering Software	Payant	UML2,0	E2,0	Java
1	Gentleware	Poseidon CE for UML	<u>Gratuit</u>	UML1,4(9/9)		Java
1	Gentleware	Poseidon for UML	Payant	UML1,4(9/9)		Java, C#, ...
2	Omondo	EclipseUML Free	<u>Gratuit</u>	UML1,4	E3,0	Java
2	Omondo	EclipseUML Studio	Payant	UML2,0	E3,0	Java
1	Sybase	PowerAMC 10	Payant	UML, Merise, ...		?
X	Open-source	ArgoUML	<u>Gratuit</u>	UML1,1(8/9)		?
X		PyUT	<u>Gratuit</u>	UML1,3		
3	I-Logix	Rhapsody Modeler	Payant	UML2,0		C,C++,ADA
3	Case France	Envision				C++, Java (2)
3	Telelogic	TAU	Payant	UML2,0		

Légende :

A – Utilisé actuellement à Capgemini Sud -Ouest

X – Plus de suivi du fournisseur

THESE-MACAO	Juin 2008	196 / 266
	Nicolas FERRY	

- 1 – Outil très intéressant
- 2 – Outil à surveiller
- 3 – Outil faiblement intéressant

Commentaires sur les produits :

IBM va continuer de maintenir les anciens outils de Rational mais nous l'avons vu : il va se focaliser sur le prochain **Atlantic** pour sa plate-forme **WebSphere/Eclipse**. Les autres outils seront maintenus mais peu supportés. Atlantic n'est pas encore disponible au moment de la rédaction de ce document. Il couvrira tout le cycle de développement. Les tarifs prévisionnels des licences sont élevés.

Du côté de chez **Borland**, je note le très bon produit **Together** dans sa communauté Edition qui présente l'avantage de faire une synchronisation en direct du modèle et du code. C'est subjectivement le meilleur moteur de synchronisation 'live' que j'ai testé pour l'instant. Le reverse engineering de l'application CMH a été réalisé à partir des fichiers java. Together a été relativement efficace. Beaucoup de classes ont été synchronisées à partir de l'existant cependant l'outil n'a pas su retrouver certains liens entre paquetages. Le résultat a été tout de même meilleur qu'avec Atlantic .

Dans sa version payante, Together supporte plusieurs plates-formes et langages pour la génération de code. Son principal inconvénient est lié au tarif des produits. Bizarrement, le tarif de Together pour Visual Studio est relativement faible, ce qui est dû à un accord entre les deux sociétés.

Avec **Entreprise Architect**, l'éditeur **SparxSystems** livre un outil riche en fonctionnalités. Il supporte la modélisation de tous les diagrammes UML et gère les profils de UML2.0. Celui-ci offre (de base) une gestion multi utilisateurs et la génération de code multi plates-formes notamment Java, C#, Delphi,... Il existe des modules qui permettent de faire du Round Trip Engineering avec Eclipse et Visual Studio .Net. Très intéressant surtout vu le tarif de ses licences. Il semble représenter le meilleur rapport fonctionnalités/prix.

SoftTeam avec **Objecteering** offre un très bon produit de modélisation de toutes les phases du cycle de développement. Celui-ci supporte notamment les dernières nouveautés en terme de conception objet : UML2.0, MDA et les « profils ». Son seul point faible : c'est d'être un outil payant.

Gentleware avec **Poseidon** offre aussi un très bon produit. L'éditeur est simple et convivial et l'on sent le souci des petits détails qui simplifient la vie. C'est un produit très complet car il gère l'intégralité des diagrammes UML. Nouveauté intéressante, dans sa nouvelle version 3.0, il peut s'intégrer à Eclipse directement.

THESE-MACAO	Juin 2008	197 / 266
	Nicolas FERRY	

Le plugin **EclipseUML** de l'éditeur **Omondo**, est basé sur son ancêtre : argoUML. Celui-ci dans sa version Free est complètement intégrable à Eclipse. Pratique pour faire de la conception, cet outil a présenté certains bugs à l'exécution, notamment il provoque la fermeture d'Eclipse de temps à autre. Le reverse de l'application CMH avec ce plugin a été laborieux. Il reste un exemple d'utilisation de la brique UML2.0 livrée avec Eclipse.

PowerAMC de **Sysbase** est l'outil de modélisation qui couvre le plus de méthodes (Merise, UML, base de données) et permet la génération dans de nombreux domaines. Le prix des licences est fortement élevé.

L'ensemble de cette étude a permis de définir la plate-forme cible de développement et en correspondance avec les besoins de Capgemini de choisir la **plate-forme Eclipse** comme socle de la plate-forme de réalisation de l'AGL MACAO. Eclipse convient bien au rôle de plate-forme commune pour une équipe de développement [72], car il peut être paramétré selon les besoins [145].

THESE-MACAO	Juin 2008	198 / 266
	Nicolas FERRY	

Chapitre V : Analyse de l'AGL MACAO

E.V.1 - Description

Sur un projet, chaque intervenant veut tirer le projet à son compte. C'est pourquoi il est important de maintenir un équilibre entre tous les intervenants.

Dans le pôle nouvelles technologies de Capgemini (Advanced Development Center), plusieurs intervenants collaborent à différents niveaux. La transmission des informations se fait par la communication et l'échange entre les individus. Chaque pont de communication est généralement rédigé sous forme d'un document. Le suivi des documents et de la méthodologie est défini, assuré par le système qualité. Capgemini utilise la méthode Deliver.

Voici maintenant une présentation du processus MACAO et des ses artefacts.

E.V.2 - Les acteurs :

V.2.1 - Maîtrise d'ouvrage : MOA

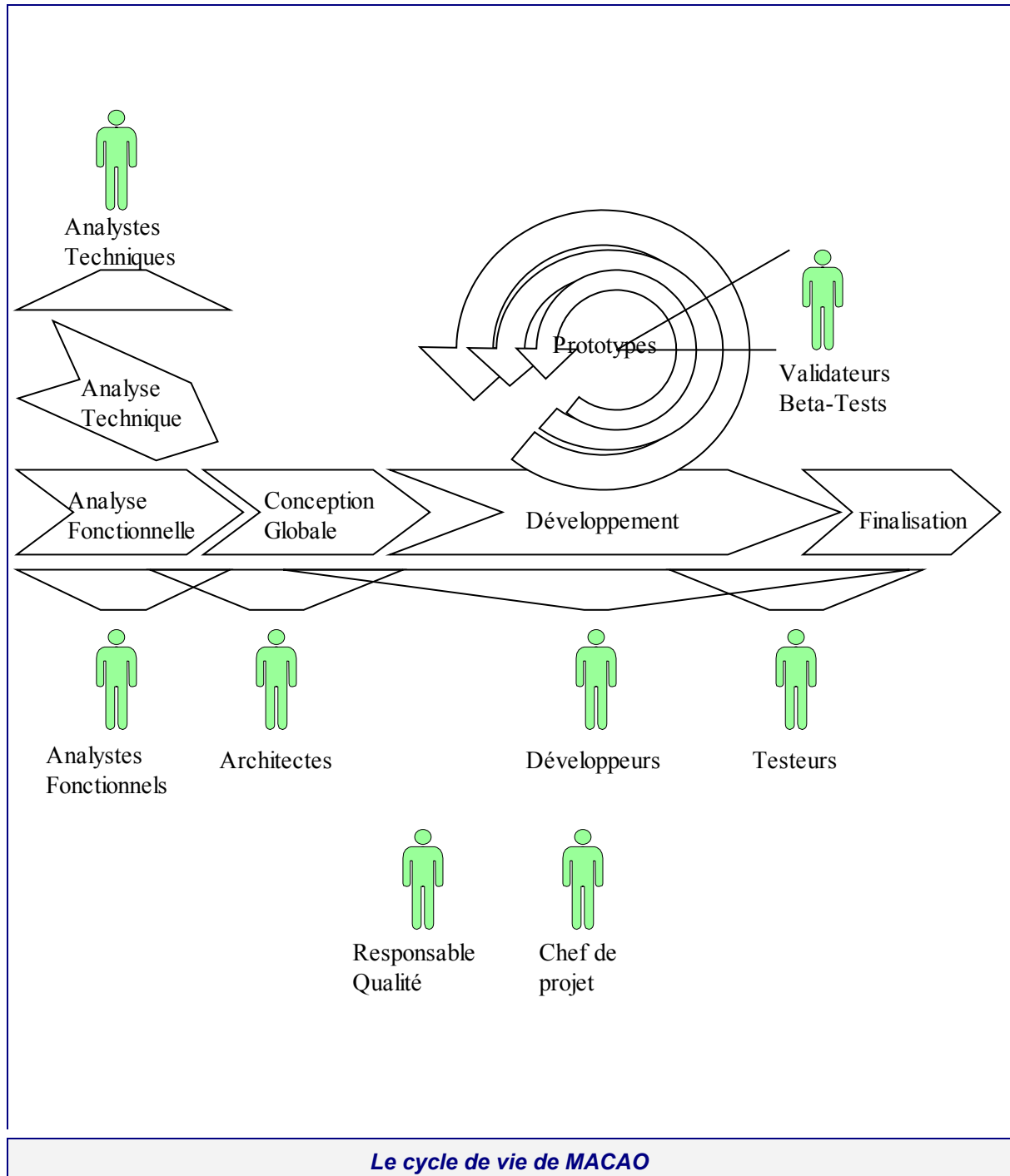
CI : Commission Informatique
DTCI : Délégué de la commission
RCI : Responsable de la CI
VAL : Validateur

V.2.2 - Maîtrise d'oeuvre : MOE

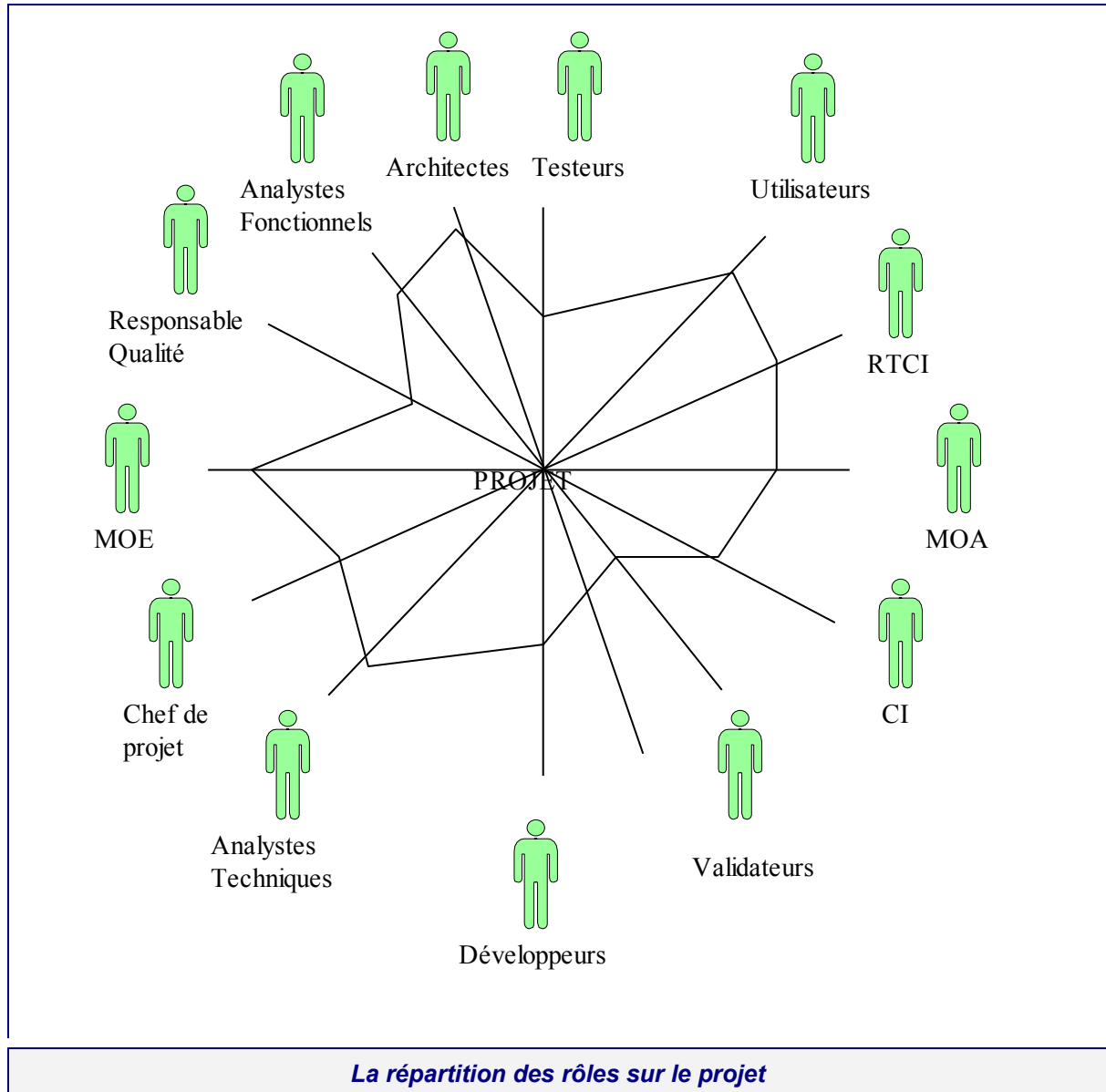
CP : Chef de projet
FONC : Fonctionnel / Analyste
ARCHI : Architecte expert technique
DEV : Développeur
TEST : Testeur
QUAL : Responsable qualité
(ERG : Ergonome)

E.V.3 - Le processus et les rôles du système

THESE-MACAO	Juin 2008	199 / 266
	Nicolas FERRY	



THESE-MACAO	Juin 2008	200 / 266
	Nicolas FERRY	



Le but d'un projet est d'amener suivant les termes d'un accord commun, les intervenants à un degré maximum de satisfaction. MACAO est orienté « Usability » ce qui veut dire qu'il recherche la satisfaction des participants.

THESE-MACAO	Juin 2008	201 / 266
	Nicolas FERRY	

E.V.4 - Description des postes de travail :

Tout au long du processus de développement de multiples intervenants participent au projet et y apportent une expertise particulière. Nous définissons un rôle ou un acteur comme un type d'intervenant particulier.

Dans MACAO, nous distinguons deux grandes catégories d'intervenants, à savoir la maîtrise d'ouvrage qui exprime des besoins et la maîtrise d'œuvre qui réalise la réponse aux besoins.

V.4.1 - Maîtrise d'ouvrage :

La maîtrise d'ouvrage regroupe la commission qui gère le projet côté client ainsi que les utilisateurs finaux expérimentés à même de valider le produit. Voici la liste des rôles pris en charge dans un projet MACAO :

- CI : Commission Informatique

La commission informatique est un regroupement de personnes qui assume la responsabilité du projet qui rassemble les principaux responsables aptes à faire connaître les besoins ainsi que des spécialistes dans des domaines particuliers.

La commission informatique ne doit pas dépasser une dizaine de membres afin d'éviter les discussions interminables.

Elle joue le rôle essentiel car elle représente le maître d'ouvrage.

- DTCl : Délégué de la commission

Le(s) délégué(s) technique(s) de la commission informatique sont des experts qui conseillent la commission pour des choix techniques.

- MOE : Responsable de la CI

Le responsable de la commission informatique est en charge de la commission informatique et représente les décisions qui y sont prises. Il a un rôle de contrôle et de vérification des fonctionnalités des prototypes.

- BTEST : Bêta-Testeurs

Ce sont des utilisateurs expérimentés choisis particulièrement pour effectuer les bêta-tests de l'application. Ils interviennent dans la phase de Bêta-tests avant la recette des prototypes.

THESE-MACAO	Juin 2008	202 / 266
	Nicolas FERRY	

V.4.2 - Maîtrise d'oeuvre :

- **CP : Chef de projet**

Le chef de projet a la responsabilité des hommes et du déroulement du projet. Il doit pouvoir notamment gérer les ressources humaines, les affectations des tâches de chaque personne. Il doit garder une vision du périmètre et des objectifs tout en manoeuvrant à partir des indicateurs du projet. Enfin, il a en charge l'organisation du projet par des revues effectuées avec la maîtrise d'ouvrage.

La gestion de projet :

« La gestion de projet consiste en l'application de connaissances, de savoir-faire, d'outils et de techniques aux activités du projet dans le but de satisfaire et de dépasser les besoins et les attentes des intervenants. » (PMI2000)

- **FONC : Fonctionnel / Analyste**

Le fonctionnel (ou analyste) doit pouvoir comprendre les besoins des utilisateurs, et des autres intervenants, savoir les formuler et les documenter sous forme d'exigences. Enfin, il doit savoir négocier certaines exigences dans son rôle avec le client.

Il développe d'une certaine façon la vision du problème énoncé par le client.

- **ARCHI : Architecte expert technique**

L'architecte a un rôle d'expert technique. Il doit avoir un point de vue plutôt technique des exigences globales, et par ailleurs un point de vue sur l'ensemble de l'architecture du système, les échanges et communication entre objets, la plate-forme de configuration, et la plate-forme physique de déploiement.

Le rôle de l'architecte évolue au cours des différentes phases.

Notamment, il doit concevoir l'architecture générale et s'assurer de la faisabilité de la solution. Tout en ayant un rôle d'intermédiaire avec les équipes et le chef de projet, il doit veiller à l'implémentation de la solution.

- **DEV : Développeur**

Le développeur doit à partir des exigences et des contraintes de conception concevoir la réalisation des composants et la rédaction du code informatique.

- **TEST : Testeur / Valideur**

Le testeur réalise les fiches de tests, les tests de scénarios suivant un plan de test.

THESE-MACAO	Juin 2008	203 / 266
	Nicolas FERRY	

- **QUAL : Responsable qualité**

Le responsable qualité décrit les règles à appliquer sur le projet. Il rédige notamment le plan qualité. Sur de gros projets, il peut alléger le chef de projet sur la mise en place du processus et des règles à mettre en œuvre. (Non géré actuellement dans MACAO)

E.V.5 - Description détaillée des rôles :

Si les étapes de MACAO sont définies et sont recommandables, la définition du processus de chaque activité interne de MACAO est adaptable suivant les projets et les personnes.

Nous allons décrire les fiches descriptives de postes qui représentent pour chaque acteur du système : les objectifs de son rôle, sa participation dans le processus, mais aussi les documents entrants, les documents sortants, les tâches qu'il réalise, et les vues qu'il aura sur le système.

Les différents acteurs d'un projet doivent pouvoir intervenir de deux façons sur le système :

- en lecture, c'est la vision des informations. Dans ce cas, on appellera cette interaction une vue : une extraction des informations du système suivant des critères donnés.

- Il est aussi possible de réaliser un traitement particulier : c'est une commande sur le système. Les commandes seront déterminées à partir des tâches qu'il réalise.

THESE-MACAO	Juin 2008	204 / 266
	Nicolas FERRY	

V.5.1 - La Commission Informatique (CI)

Code : CI

Désignation : Commission Informatique de la maîtrise d'ouvrage

Définition :

La commission informatique a la responsabilité du projet. Elle rassemble les principaux responsables aptes à faire connaître les besoins ainsi qu'un ou plusieurs spécialistes des domaines concernés par le projet. Les spécialistes ont le rôle de Délégués Technique de la Commission Informatique (DTCI).

Buts :

Code	Description
B1	Fixer les objectifs généraux et les priorités d'informatisation
B2	Contrôler l'avancement du travail et procéder à des réorientations éventuelles de objectifs
B3	Attribuer les budgets nécessaires
B4	Assurer la liaison avec la direction
B5	Choisir les intervenants internes
B6	Réaliser les appels d'offres concernant les SSII ou les intervenants externes
B7	Aider à la définition des différents prototypes
B8	Autoriser ou interdire les modifications de fonctions ou de planning
B9	<i>Recetter</i> les prototypes et le logiciel livré
B10	Etablir tous les documents contractuels concernant la garantie, la maintenance, la formation,...

Participation :

Etape	Phase	Description
ANALYSE GLOBALE	Phase de Préparation	<ul style="list-style-type: none"> - Définir les grandes lignes du projet - Nommer les responsables et les principaux intervenants
TOUT AU LONG DU PROJET	Suivi du projet	<ul style="list-style-type: none"> - Phase de réunion - Suivi - Pilotage

THESE-MACAO	Juin 2008	205 / 266
	Nicolas FERRY	

Document en entrées :

Code	Description	Origine	Observations
CRP	Certificat Recette Prototype	CP	
TOUS	Tous les dossiers du projet	CP	

Documents en sorties :

Code	Description	Destination	Observations
FAP	Fiche Anomalie Prototype	CP	Normalement pas utile à la MOA
FMP	Fiche Modification Prototype	CP	
CRP	Certificat Recette Prototype	CP	Signé ou avec des remarques
TOUS	Tous les dossiers du projet	CP	

Tâches réalisées :

Code	Description	Observations
T0	Signature du contrat	Début Projet
T1	Relier Personnes aux rôles	
T2	Définir des instances / Groupes	
T3	Pilotage du projet	
T4	Organiser une réunion	
T5	Exprimer les besoins métiers	Emettre CCU
T6	Exprimer un nouveau besoin ou un changement	Emettre FMP
T7	Valider la spécification	Réception DSP
T8	Valider le dossier de la conception globale	Réception DCG
T9	Etudier les impacts d'une DMP	Réception DMP
T10	Validation Prototype – signature recette	Emettre CRP Signé

V.5.2 - Les Bêta-testeurs / Valideurs (BTEST)

Code : BTEST

Désignation : Les valideurs ou Bêta-Testeurs. (côté client)

Définition :

Les bêta-testeurs déroulent leurs tests sans contrainte afin qu'ils puissent explorer tous les cas qu'ils souhaitent.

Ils sont guidés dans cette tâche par le manuel livré avec le prototype (MUP).

Ils peuvent émettre deux sortes de retours :

- Des anomalies de fonctionnement :

Celles-ci sont inventoriées par les fiches d'anomalie Prototype (FAP).

- et l'expression de nouveaux besoins :

Concernant les nouvelles idées ou des changements de fonctionnalités. Elles sont inventoriées par les Fiches de Modification Prototype (FMP).

Buts :

Code	Description
B1	Assurer que le prototype livré est valide par rapport au besoin initial
B2	<i>Recetter</i> les prototypes et le logiciel livré

Participation :

Etape	Phase	Description
ANALYSE GLOBALE	Phase de Préparation	- Définir les grandes lignes du projet
DEVELOPPEMENT	Phase 5 : Tests utilisateurs	- Bêta-tests par les utilisateurs en conditions réelles et externes. (REUP)

Document en entrées :

Code	Description	Origine	Observations
PROTO	Le prototype du logiciel	CP (-DEV)	
MUP	Manuel utilisateur prototype	CP (-DEV)	
GIP	Guide d'installation prototype	CP (-DEV)	
MU	Manuel utilisateur	CP (-DEV)	
GI	Guide d'installation	CP (-DEV)	

THESE-MACAO	Juin 2008	207 / 266
	Nicolas FERRY	

Documents en sorties :

Code	Description	Origine	Observations
FAP	Fiche Anomalie Prototype	CP	Un ou plusieurs par prototype
FMP	Fiche Modification Prototype	CP	Un ou plusieurs par prototype
REUP	Rapport Utilisateur Essai Prototype	CP	Un par prototype

Tâches réalisées :

Code	Description	Observations
T1	Vérifier que le prototype livré est conforme et valide	Bêta-tests + Recette
T2	Vérifier que le logiciel livré est conforme et valide	Logiciel final + Recette

THESE-MACAO	Juin 2008	208 / 266
	Nicolas FERRY	

V.5.3 - Le chef de projet (CP)

Code : CP

Désignation : Chef de projet

Définition :

« La gestion de projet consiste en l'application de connaissances, de savoir-faire, d'outils et de techniques aux activités du projet dans le but de satisfaire et de dépasser les besoins et les attentes des intervenants. » [PMI2000]

Le rôle de chef de projet est complexe car il faut trouver un équilibre entre :

- Périmètre fonctionnel, temps, coût, qualité
- Intervenants (internes ou externes) avec différents besoins et attentes
- Exigences identifiées (besoins) et non identifiées (attentes)

Néanmoins, nous considérons que le chef de projet doit atteindre les buts suivants :

Buts :

Code	Description
B1	Obtenir la satisfaction des intervenants
B2	Organiser, gérer et assurer la réussite du projet

Participation : Toutes les phases du projet

Etape	Phase	Description
ANALYSE GLOBALE	Phase de Préparation	- Définir les grandes lignes du projet
ANALYSE GLOBALE	Phase d'Investigation	- Obtenir suffisamment d'informations sur le système - Rédaction du cahier des charges utilisateurs s'il y a lieu. (CCU)
ANALYSE GLOBALE	Phase de spécifications	- Procéder à un examen critique de l'existant. - Rechercher les cas d'utilisation permettant de traduire les résultats obtenus lors de la phase d'investigation en terme mieux adaptés à leur informatisation. (DSP) - Proposer plusieurs niveaux de solutions : Etude d'opportunités (ROP)
THESE-MACAO	Juin 2008	209 / 266
	Nicolas FERRY	

CONCEPTION GLOBALE	Prototype validant la plate-forme	- Etudier la faisabilité
CONCEPTION GLOBALE	Conception globale	<ul style="list-style-type: none"> - Construire l'architecture fonctionnelle - Conception de l'IHM - Définir les prototypes - Afin de suivre l'évolution du projet au travers de la réalisation de ses divers prototypes. (TDT) - Rédaction DCG et PDV
DEVELOPPEMENT	Phase 1 : Définition du prototype	Compléter le périmètre fonctionnel (DDP-DRP)
DEVELOPPEMENT	Phase 2 : Conception détaillée	Produire tous les éléments nécessaires au codage du prototype. (DRP) Plan d'essai du prototype (PEP)
DEVELOPPEMENT	Phase 3 : Codage	Produire les programmes exécutables.
DEVELOPPEMENT	Phase 4 : Intégration	Mise en place du prototype en conditions réelles simulées pour préparer sa livraison.
DEVELOPPEMENT	Phase 5 : Tests utilisateurs	Fournir aux Bêta-testeurs les éléments nécessaires et prendre notes des retours par les FAP et les FMP.
FINALISATION	Recette	Passage du dernier prototype recetté à la version client livrée à la MOA. Certificat de Recette Prototype. (CRP)

Document en entrées :

Code	Description	Origine	Observations
TOUS	Interface tous les documents	Variables	Possède un accès à tous les éléments
THESE-MACAO	Juin 2008	210 / 266	
	Nicolas FERRY		

CCU	Cahier des charges	MOA	Exigences et contraintes
-----	--------------------	-----	--------------------------

Documents en sorties :

Code	Description	Destination	Observations
TOUS	Interface tous les documents en transit entre le client et les autres intervenants	Variables	Possède un accès à tous les éléments
DSP	Dossier de SPécifications	MOA	Reformulation de la compréhension du CCU.
DCG	Dossier de Conception Gloable	MOA	Documentation technique.
ROP	Rapport d'opportunités	MOA	Propositions de solutions
PQL	Plan qualité	FONC ARCHI DEV	A la charge du responsable qualité.
PDV	Plan de développement	TOUS	Planning des prototypes
DRP	Dossier de Réalisation Prototype	TOUS	

Tâches réalisées :

Code	Description	Observations
T1	Définir un nouveau projet	
T1.1	Créer un nouveau projet dans MACAO	
T1.2	Signature du contrat	MOA et MOE (CP)
T2	Gérer les ressources humaines	Contact humain, Personnes
T2.1	Relier les personnes	
T2.2	Définir les groupes / instances	
T3	Organiser et gérer le processus MACAO de réalisation (Affinage dynamique du processus suivant les équipes)	Processus Affectation des tâches
T4	Planifier, organiser et gérer l'évolution du projet	Planning

THESE-MACAO	Juin 2008	211 / 266
	Nicolas FERRY	

		Indicateurs de contrôle : Trajectoire en cours, comptes-rendus de revues, liste des problèmes, évaluation de l'état en cours.
T4.1	Pilotage du projet	Planning, périmètre, contraintes, risques, objectifs
T4.1.1	Définition des objectifs et périmètres	périmètre, contraintes, risques, objectifs
T4.1.2	Vérifier que la demande de modification (FMP) soit dans le périmètre du projet	périmètre, contraintes, risques, objectifs
T4.2	Organiser des réunions	Planning, Rédaction de compte rendu
T4.3	Définir le plan de développement	Rédaction du PDV
T4.4	Etudier impact d'une demande de modification (DMP)	DMP de l'équipe de DEV
T4.5	Traitement d'une anomalie (FAP)	Gestion FAP
T5	Validation du prototype	Recette (CRP signé)
T5.1	Préparation de la livraison	
T5.2	Archivage des sources précédentes	
T5.3	Configuration du nouveau prototype	
T5.4	Déblocage de certaines classes	Suite à une DMP
T6	Communications & Approbation de documents	Gestion des documents
T7	Gestion de la traçabilité (Liens des besoins, de la conception jusqu'au code)	Archivage de tous les documents.

THESE-MACAO	Juin 2008	212 / 266
	Nicolas FERRY	

Le chef de projet doit avoir une vision sur plusieurs éléments du système. Il devra pouvoir consulter les vues suivantes :

- Plan de développement (PDV)
- Plan de test
- Plan de gestion de configuration
- Plan d'évaluation
- Gestion des risques
- Plan de documentation (Planche des documents de MACAO)
- Plan qualité (PQL)

Ses vues du projet :

Code	Description
V1	Initialisation d'un projet
V2	La vue des objectifs
V3	La vue du périmètre / contraintes / risques
V4	La vue du plan de développement
V5	La vue des contraintes (impératifs budgétaires, choix technologiques, système d'exploitation, exigences de cohabitation ou de compatibilité avec l'existant.
V6	La vue des intervenants (clients, utilisateurs, testeurs, chefs de projets, concepteurs, testeurs)
V7	La vue des fonctionnalités : Services fournis par le système à ses utilisateurs.
V8	La vue des besoins (CCU)
V9	La vue des exigences (DSP)
V10	La vue de la matrice de couverture des exigences/Besoins

Notes :

- Le chef de projet fait partie de l'équipe et travaille en collaboration avec les autres membres.
- Il est responsable de l'élaboration et de la modification du plan de développement logiciel (PDV).
- Le PDV se fonde sur un processus préalablement configuré et adapté au contexte du projet.

THESE-MACAO	Juin 2008	213 / 266
	Nicolas FERRY	

- Confronté à des compromis, le chef de projet doit faire les choix appropriés pour gérer le périmètre du projet à chaque itération.
- Il doit être focalisé sur les risques et les moyens de limiter leur impact si leur probabilité était avérée.
- Il doit se concentrer sur des résultats concrets.

THESE-MACAO	Juin 2008	214 / 266
	Nicolas FERRY	

V.5.4 - Le fonctionnel / analyste (FONC):

Code : FONC

Désignation : Fonctionnel ou Analyste

Buts :

Code	Description
B1	Prendre connaissance et décrire l'existant
B2	Définir le périmètre fonctionnel du domaine concerné
B3	Recueillir les besoins des utilisateurs et établir les spécifications

Document en entrées :

Code	Description	Origine	Observations
CCU	Cahier des charges fourni par la MOA	MOA	Exigences et contraintes
FMP	Fiche de modifications prototype	MOA/VAL	Nouvelles exigences

Documents en sorties :

Code	Description	Destinataire	Observations
CCU	Cahier des Charges Utilisateur	MOA	Si Aide à la rédaction du CCU
DSP	Dossier de Spécification	MOA, ARCHI	Reformulation des exigences
DDP	Dossier de Définition Prototype	ARCHI	Affinage du périmètre prototype
DRP	Dossier de Réalisation Prototype	ARCHI	Initialisation du dossier de réalisation

Dans la phase d'analyse globale, le fonctionnel doit avoir déterminé de 30 à 50% des besoins utilisateurs et de 10 à 20% des besoins techniques.

THESE-MACAO	Juin 2008	215 / 266
	Nicolas FERRY	

Tâches réalisées :

Code	Description	Observations
T1	Comprendre les besoins des utilisateurs et des autres intervenants : <ul style="list-style-type: none"> - Documenter, hiérarchiser et communiquer ces exigences - Négocier les exigences et faciliter l'acceptation de l'application par le client 	Rédiger DSP MOA
T2	Définition du prototype	Rédiger DDP et DRP

Participation :

Etape	Phase	Description
ANALYSE GLOBALE	Phase de Préparation	<ul style="list-style-type: none"> - Définir les grandes lignes du projet - Nommer les responsables et les principaux intervenants
ANALYSE GLOBALE	Phase d'Investigation	<ul style="list-style-type: none"> - Obtenir suffisamment d'informations sur le système - Rédaction du cahier des charges utilisateurs s'il y a lieu. (CCU)
ANALYSE GLOBALE	Phase de spécifications	<ul style="list-style-type: none"> - Procéder à un examen critique de l'existant. - Rechercher les cas d'utilisation permettant de traduire les résultats obtenus lors de la phase d'investigation en terme mieux adaptés à leur informatisation. (DSP)
DEVELOPPEMENT	Phase 1 : Définition du prototype	– Complète le périmètre fonctionnel (DDP-DRP)

Ses vues du projet :

Code	Description
V1	Les vues sur les modèles d'analyse
V2	La vue décrivant l'activité de l'entreprise

THESE-MACAO	Juin 2008	216 / 266
	Nicolas FERRY	

V3	La vue des intervenants (clients, utilisateurs, testeurs, chef de projet, concepteurs, testeurs)
V4	La vue des contraintes (impératifs budgétaires, choix technologiques, système d'exploitation, exigences de cohabitation ou de compatibilité avec l'existant.
V5	La vue de l'énoncé du problème : Description distincte du problème à résoudre
V6	La vue des fonctionnalités : Services fournis par le système à ses utilisateurs.
V7	La vue des besoins (CCU)
V8	La vue des exigences (DSP)
V9	La vue de la matrice de couverture des exigences/Besoins

V.5.5 - L'architecte (ARCHI)

Code : ARCHI

Désignation : Architecte système

Le travail de l'architecte est un compromis entre :

- Fonctionnalité
- Compétences
- Qualités
- Technologie
- Complexité
- Organisation et culture
- Aspect métier
- Processus
- Qualité

Buts :

Code	Description
B1	Définir la stratégie d'implémentation (architecture globale)
B2	Obtenir la faisabilité de la solution
B3	Assurer que l'implantation respecte les objectifs

Documents en entrées :

Code	Description	Origine	Observations
CCU	Cahier des charges	MOA	Besoins des utilisateurs
DSP	Dossier de spécification	FONC	Exigences et contraintes
DDP et DRP	Phase de conception prototype	FONC	Exigences et contraintes

Documents en sorties :

Code	Description	Destinataire	Observations
DCG	Dossier de Conception Globale	MOA, CP, DEV	Description de l'architecture

THESE-MACAO	Juin 2008	218 / 266
	Nicolas FERRY	

PROTO	Prototype architectural	MOA, DEV	Preuve architecturale (cycle en Y)
PEP	Plan d'Essai Prototype	DEV	Rédaction du plan de tests

Le dossier d'architecture DCG contribue et doit suivre la vision du projet.

Participation :

Etape	Phase	Description
CONCEPTION GLOBALE	Prototype validant la plate-forme	- Etudier la faisabilité
CONCEPTION GLOBALE	Conception globale	- Construire l'architecture fonctionnelle
CONCEPTION GLOBALE	Conception du SNI Phase 1 à 8	- Construire l'IHM
CONCEPTION GLOBALE	Recherche des Prototypes	- Rechercher quels sont les prototypes qui seront développés à l'étape suivante.
CONCEPTION GLOBALE	Traçabilité des fonctions (TDT)	- Afin de suivre l'évolution du projet au travers de la réalisation de la réalisation de ses divers prototypes.
CONCEPTION GLOBALE	Rédaction DCG	- Rédaction du dossier de Conception Globale (DCG)
CONCEPTION DETAILLEE	Conception détaillée	- Conception détaillée d'un prototype - Rédaction du plan d'essais prototypes (PEP)
DEVELOPPEMENT	Phase 2 : Conception détaillée Rédaction PEP	- Produire tous les éléments nécessaires au codage du prototype. (DRP) - Plan d'essai du prototype (PEP)

THESE-MACAO	Juin 2008	219 / 266
	Nicolas FERRY	

Tâches réalisées :

Code	Description	Observations
T1	Analyser les spécifications Pour concevoir l'architecture générale - Valider un prototype architectural	Rédaction du DCG Faisabilité
T2	Conception du prototype	Rédaction du PEP et MAJ du DRP
T3	Conseiller le chef de projet sur les choix et les coûts techniques	
T4	Assurer que l'implémentation respecte l'objectif Arbitrage de l'équipe	
T5	Gestion des briques de construction (Design pattern)	Base de connaissances
T6	Décisions critiques relatives aux principaux défauts	Phase Livraison

Ses vues du projet :

Code	Description
V1	La vue sur les exigences : <ul style="list-style-type: none"> - Les besoins utilisateurs - Les réponses aux exigences - Les cas d'utilisation - Les fonctionnalités du système - La matrice de traçabilité fonctionnelle (DTD)
V2	La vue logique du système décrit la structure : <ul style="list-style-type: none"> - Architecture du logiciel <ul style="list-style-type: none"> o Diagrammes UML + MACAO - Historique / palette de composants utilisés sur d'autres projets <ul style="list-style-type: none"> o Expérience / Réutilisation - Usage

THESE-MACAO	Juin 2008	220 / 266
	Nicolas FERRY	

	<ul style="list-style-type: none"> - Performances - Robustesse - Réutilisation - Compréhensibilité - Contraintes / Compromis économique et technique - Aspect esthétique
V3	<p>La vue des processus décrit la dynamique du système :</p> <ul style="list-style-type: none"> - Diagramme d'interactions, états-transitions - Diagramme de séquences - Communications - Synchronisation
V4	<p>La vue d'implémentation décrit les ressources informatiques physiques :</p> <ul style="list-style-type: none"> - Diagramme physique des BD, fichiers, modules
V5	<p>La vue de déploiement décrit l'implantation physique en production :</p> <ul style="list-style-type: none"> - diagramme de déploiement

V.5.6 - Le développeur (DEV) :

Code : DEV

Désignation : Développeur (Concepteur / Réalisateur / Intégrateur)

Le développeur consulte les exigences générales du système et les contraintes de conception. Il a accès à l'architecture générale décrite par l'architecte.

A partir de ces informations, il réalise le codage. Certaines de ces tâches peuvent être assimilées à de la conception, de l'implémentation, de tests du code et des données.

Buts :

Code	Description
B1	Réaliser la solution

Documents en entrées :

Code	Description	Origine	Observations
PDV	Plan de développement	CP	Planning des prototypes
DR	Dossier de réalisation	ARCHI	Source pour la réalisation

Documents en sorties :

Code	Description	Destinataire	Observations
CODE	Sources du système	ARCHI, CP	
DATA	Base de données	ARCHI, CP	
DRP	Dossier de Réalisation Prototype	ARCHI, CP	
REIP	Rapport d'Essais d'Intégration Prototype	FONC	Scénarios de tests
DMP	Demande de modification prototype	CP	Il peut y avoir un niveau MOA

THESE-MACAO	Juin 2008	222 / 266
	Nicolas FERRY	

Participation :

Etape	Phase	Description
DEVELOPPEMENT	Phase 3 : Codage	Produire les programmes exécutables.

Tâches réalisées :

Code	Description	Observations
T1	Réalisation du prototype	
T1.1	Analyser les exigences	retour sur l'architecture
T1.2	Concevoir l'architecture détaillée	
T1.3	Réaliser le codage des fonctions	
T1.4	Intégrer les fonctionnalités	
T1.5	Vérifier les fonctionnalités	Tests unitaires
T1.3b	Réaliser la base de données	
T1.4b	Intégrer les données au système	
T1.5b	Vérifier le stockage des données	Tests unitaires
T1.6	Gestion des composants	Base de connaissances
T2	Emettre une DMP	Vers le CP
T3	Modifier le prototype	Suite à une DMP validée

Ses vues du projet :

Code	Description
V1	La vue sur les exigences manifestes
V2	La vue sur l'architecture du système
V3	La vue sur les fonctions et le codage
V4	La vue sur les données et leur persistance (BD)
V5	La vue sur les éléments de tests unitaires
V6	La vue sur les composants

THESE-MACAO	Juin 2008	223 / 266
	Nicolas FERRY	

V.5.7 - Les testeurs (TEST)

Code : TEST

Désignation : Testeurs de logiciel (rôle d'intégrateur)

Le testeur doit pouvoir avoir accès :

- Exigences
- Code (en lecture seul à priori)
- Plan de test / Fiches de test / Résultats attendus
- Liste des idées de test (idée de test et approche de ce qu'il ne faut pas faire)

Buts :

Code	Description
B1	Assurer la convergence exigences / produit.

Documents en entrées :

Code	Description	Origine	Observations
PDV	Plan de développement	CP	Lecture seule
PEP	Plan d'Essais Prototype	FONC	Scénarios de tests
DRP	Dossier de Réalisation Prototype	DEV	

Documents en sorties :

Code	Description	Destinataire	Observations
REUP	Rapport d'Essais d'Utilisateur Prototype	VAL	Modèle
MUP	Manuel Utilisateur Prototype	VAL	
GIP	Guide d'installation Prototype	VAL	
DRP	Dossier de Réalisation Prototype	FONC	Passage au FONC pour prototype suivant
PROTO	Livraison Prototype	VAL, MOA	Livraison d'un prototype

Documents en sorties : (En Etape de Finalisation)

Code	Description	Destinataire	Observations
MU	Manuel Utilisateur Prototype	VAL, MOA	Complété à partir du MUP

THESE-MACAO	Juin 2008	224 / 266
	Nicolas FERRY	

GI	Guide d'installation Prototype	VAL, MOA	Complété à partir du GIP
DR	Dossier de Réalisation	VAL, MOA	Complété à partir du DRP
LOGICIEL	Livraison Logiciel	VAL, MOA	Livraison du Logiciel

Participation :

Etape	Phase	Description
DEVELOPPEMENT	Phase 4 : Intégration	Mise en place du prototype en conditions réelles simulées pour préparer sa livraison.
FINALISATION		Passage du dernier prototype recetté à la version client livrée à la MOA

Tâches réalisées :

Code	Description	Observations
T1	Intégrer le système	Intégrateur
T1.1	Intégrer les données au système	Intégrateur
T1.2	Relire le code / revoir les données	Relecture
T1.3	Vérifier les fonctionnalités et les données	Tests unitaires
T1.4	Tester la non régression de prototypes	Tests de scénarios
T1.5	Gestion de base de tests	Base de connaissances

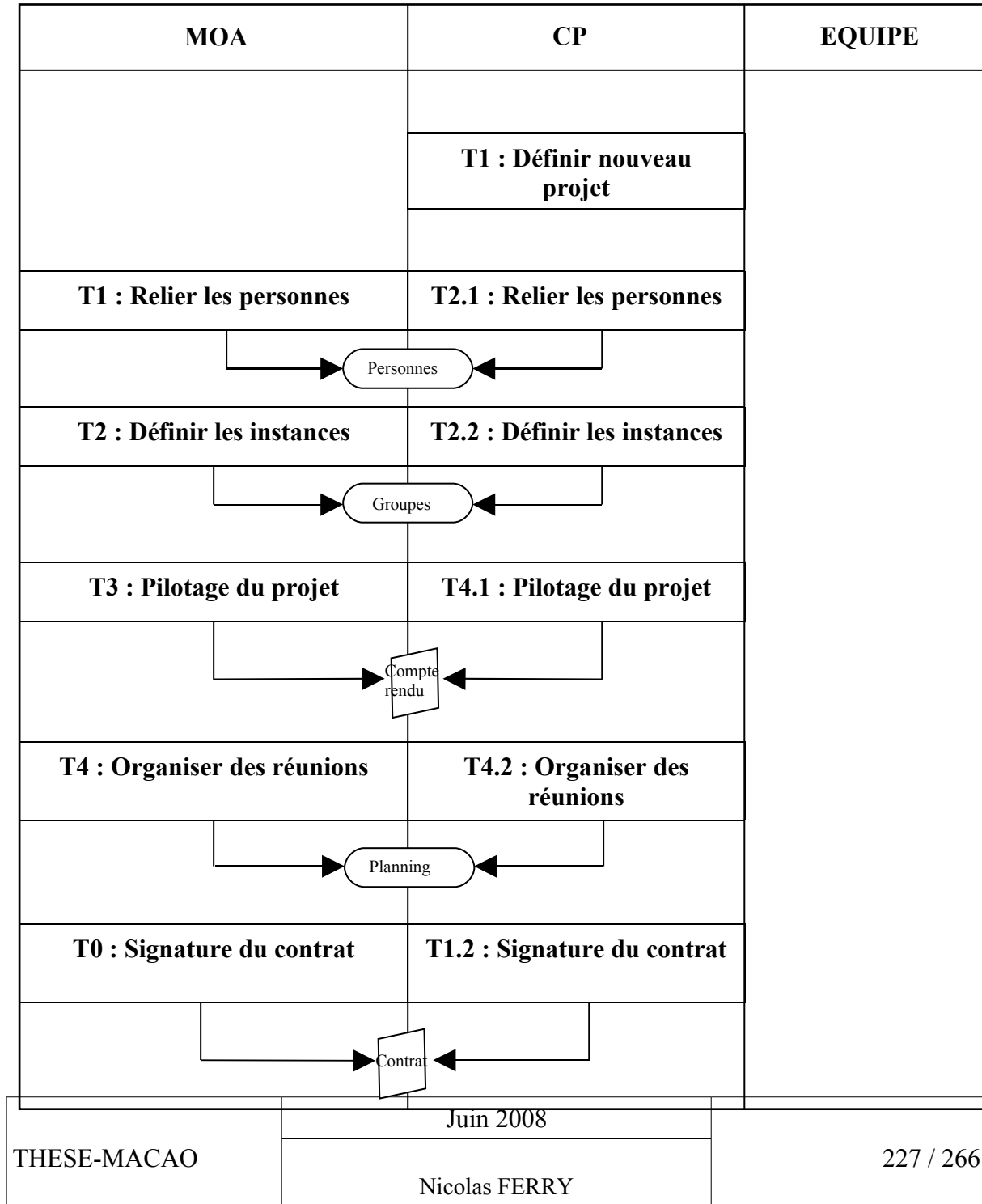
Ses vues du projet :

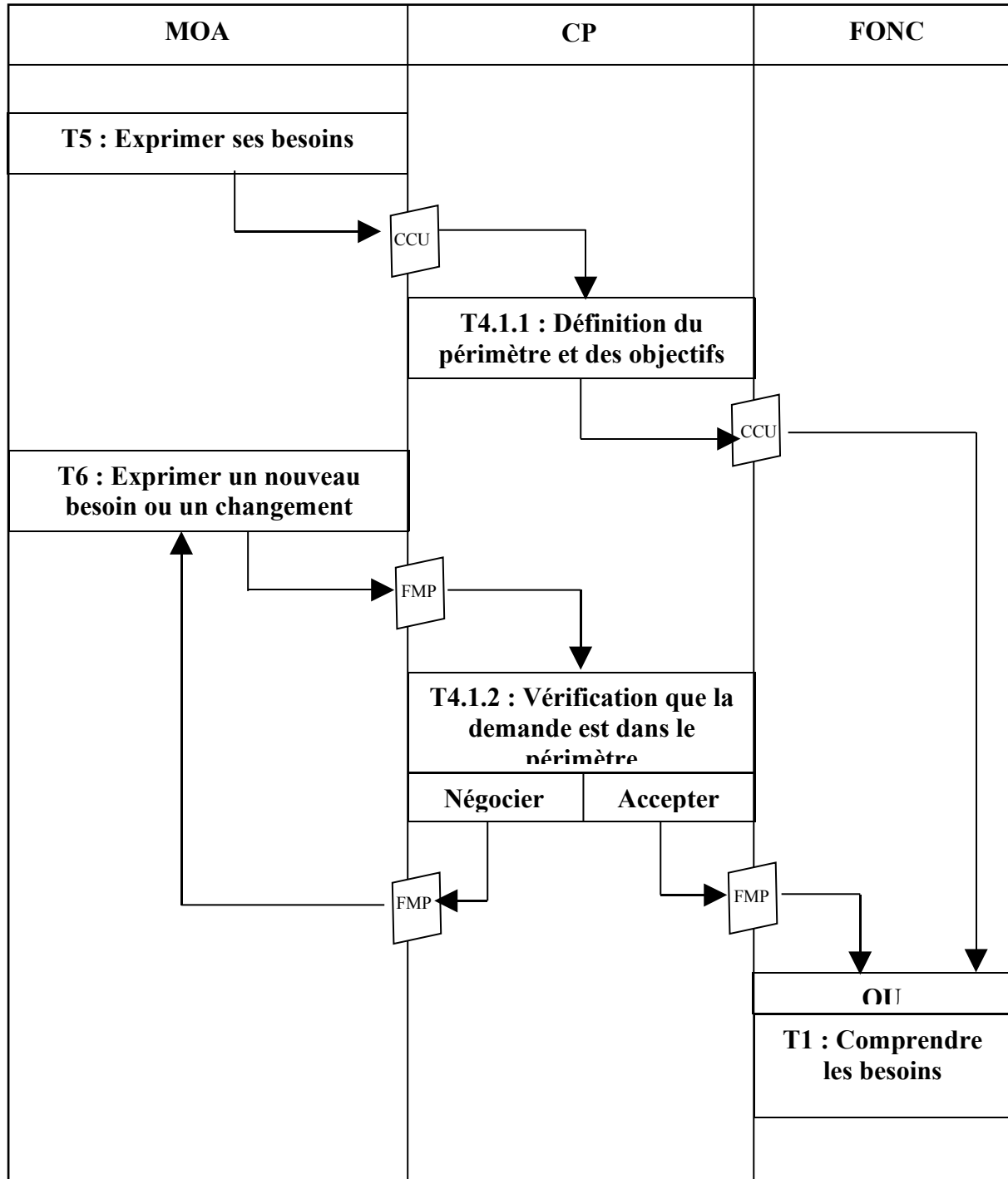
Code	Description
V1	La vue sur les exigences manifestes
V2	La vue sur l'architecture du système
V3	La vue sur les fonctions et le codage
V4	La vue sur les données et leur persistance (BD)
V5	La vue sur les éléments de tests unitaires
V6	La vue sur les composants

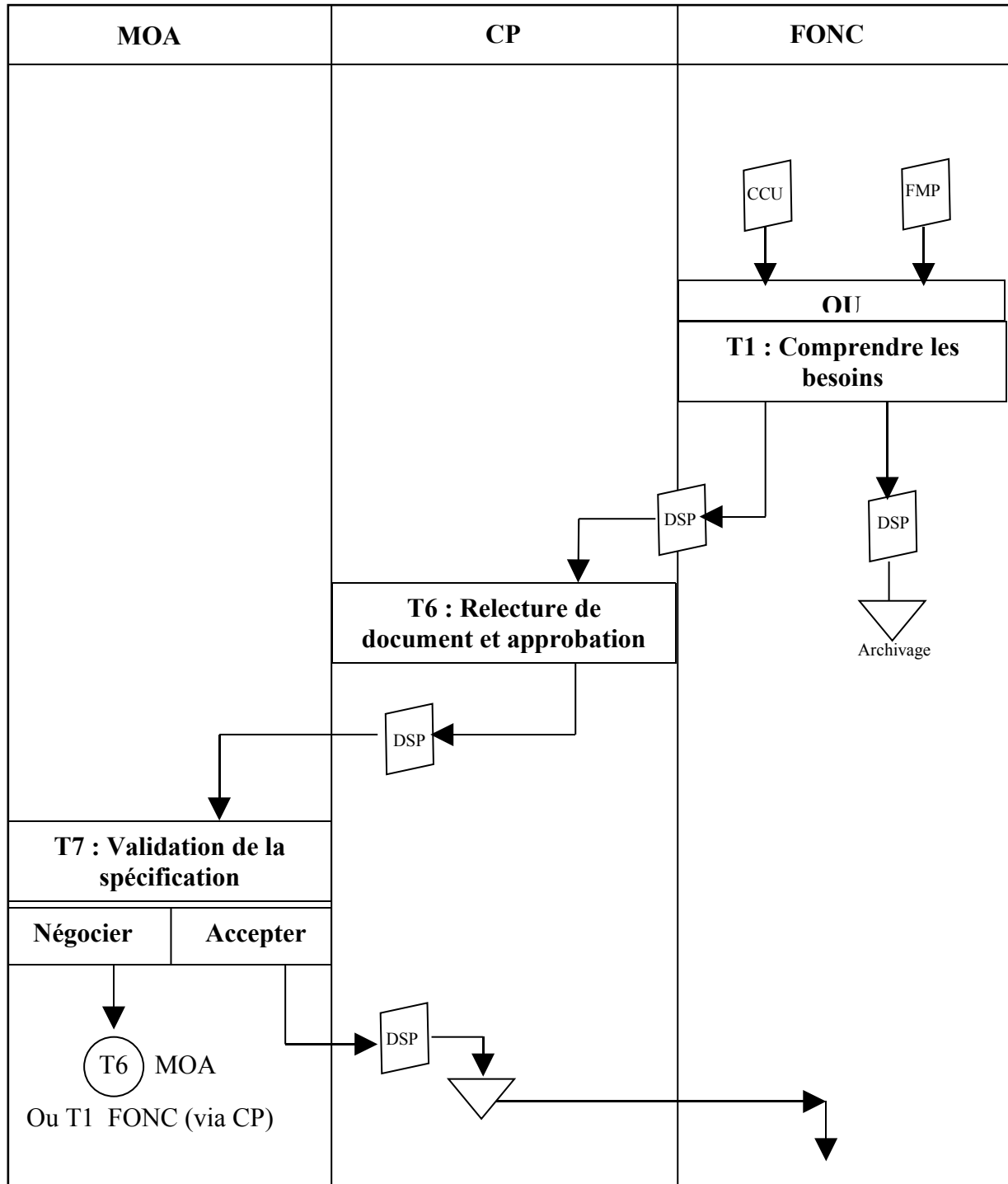
THESE-MACAO	Juin 2008	225 / 266
	Nicolas FERRY	

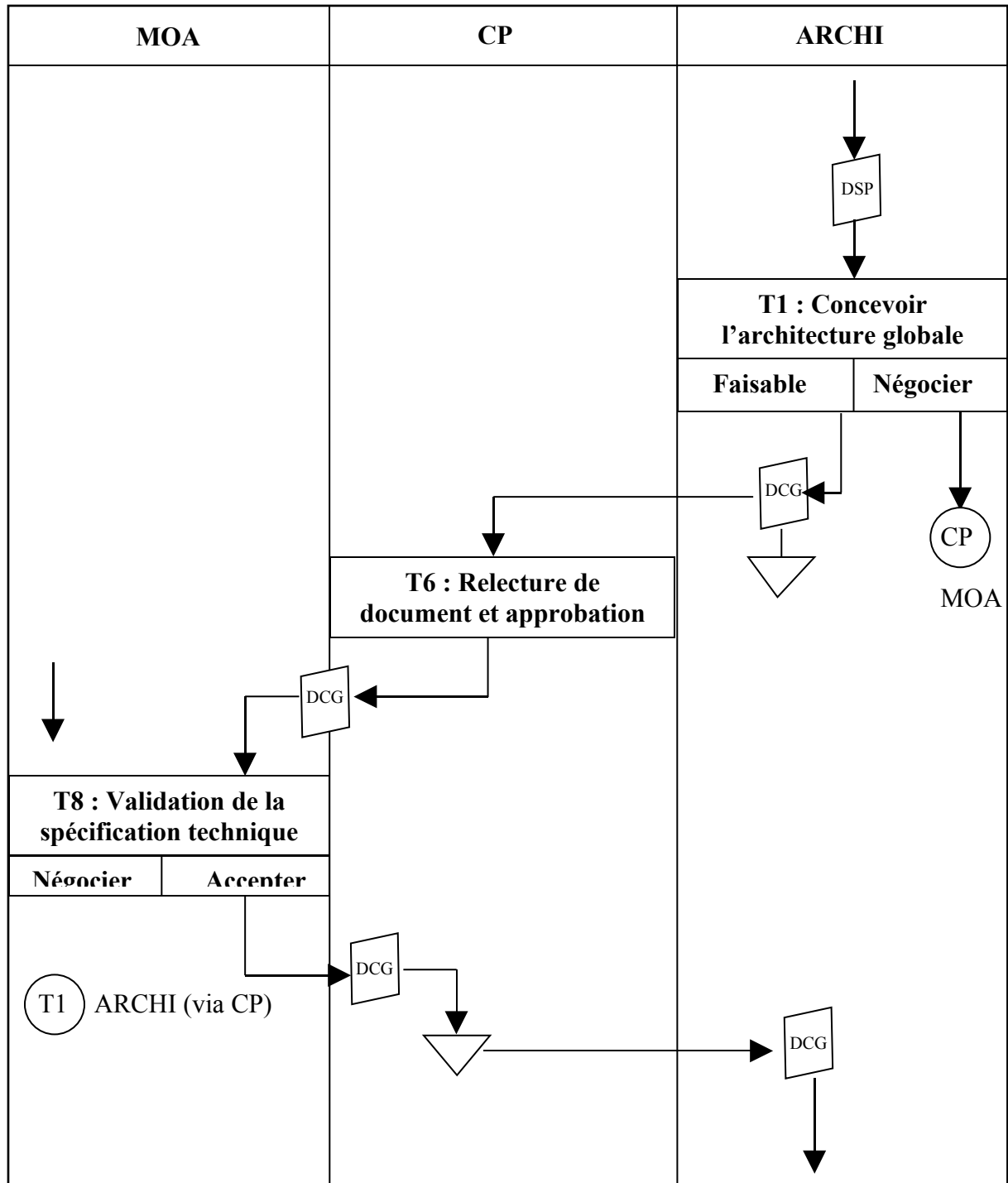
THESE-MACAO	Juin 2008	226 / 266
	Nicolas FERRY	

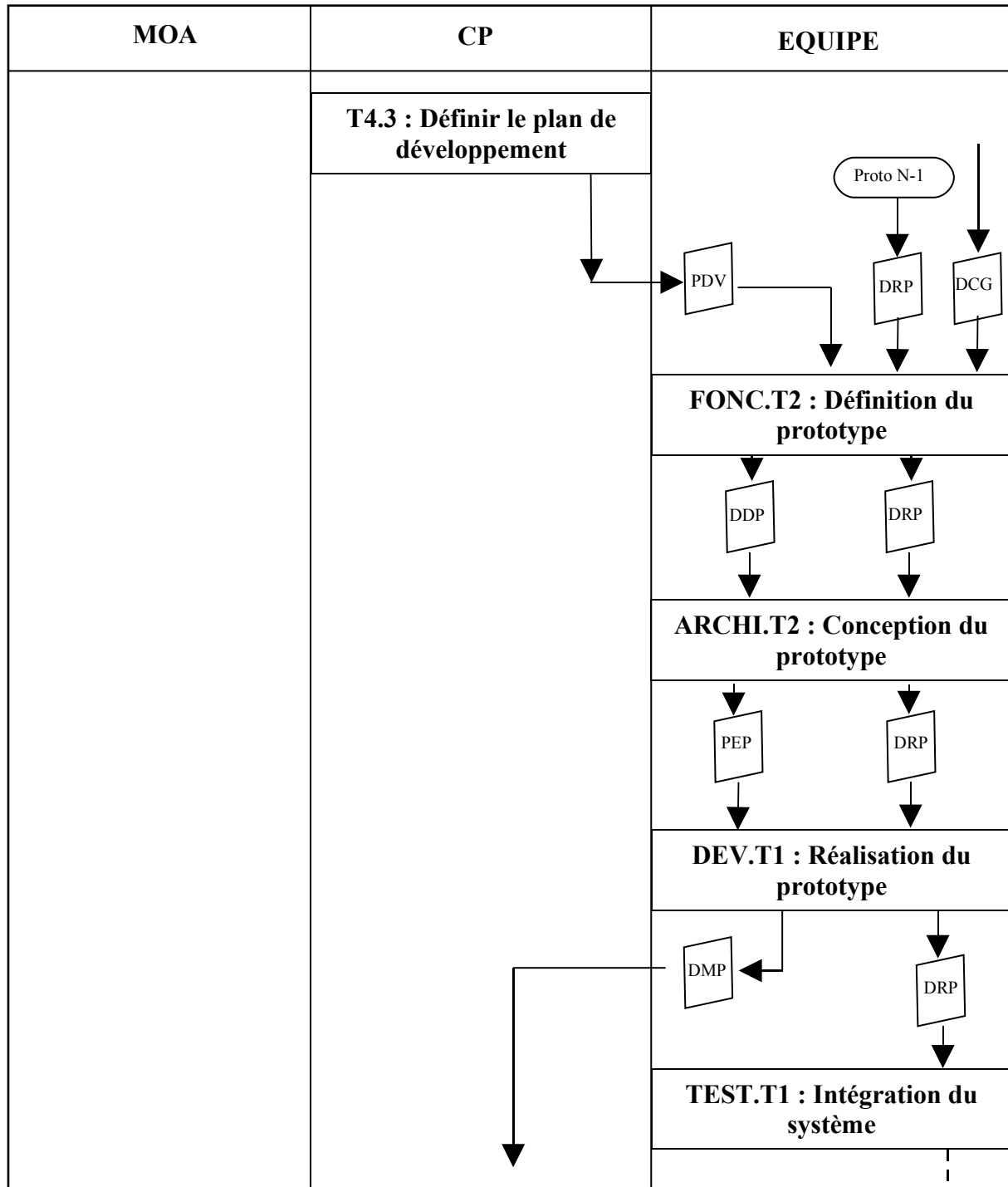
E.V.7 - Diagramme des circuits et des tâches

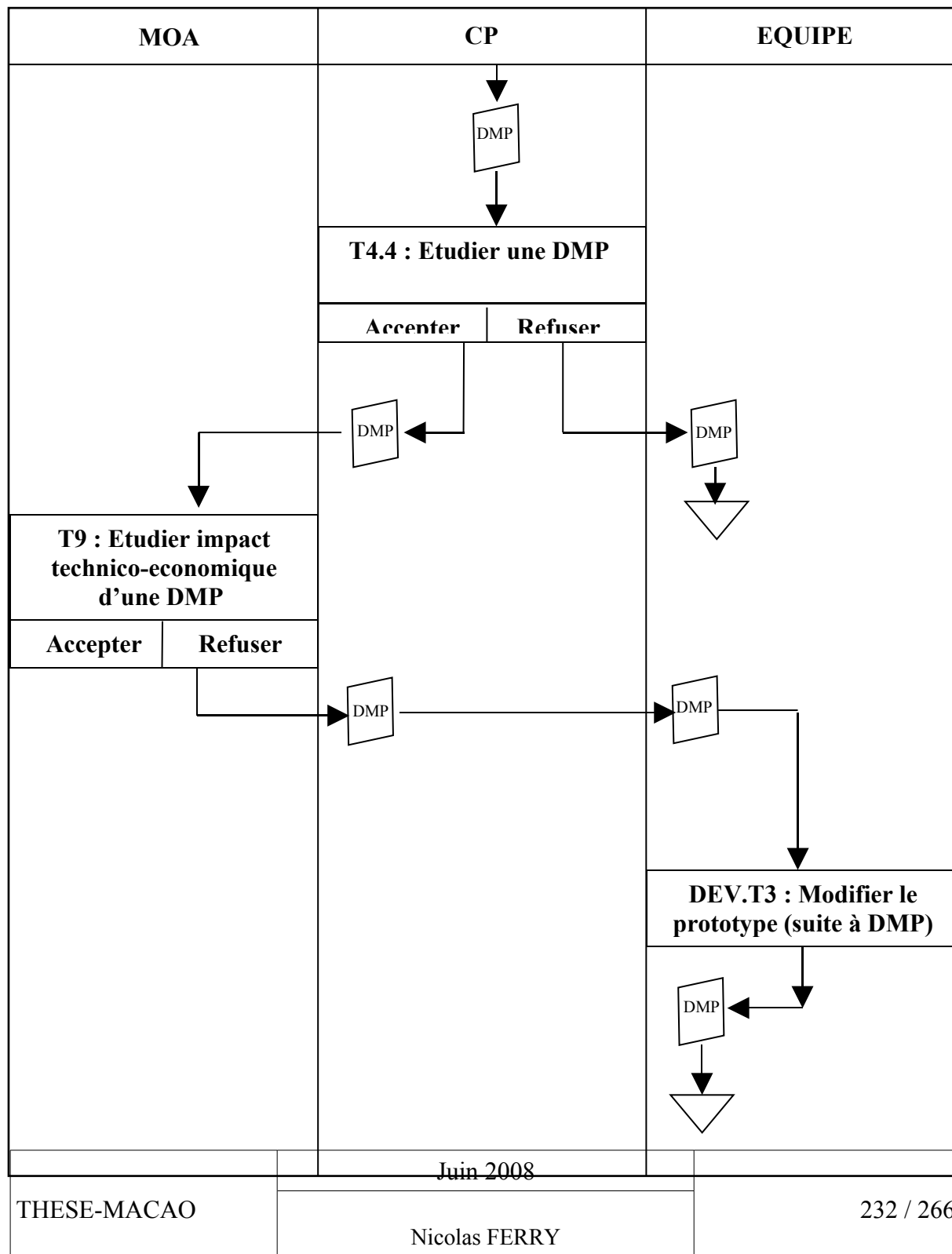


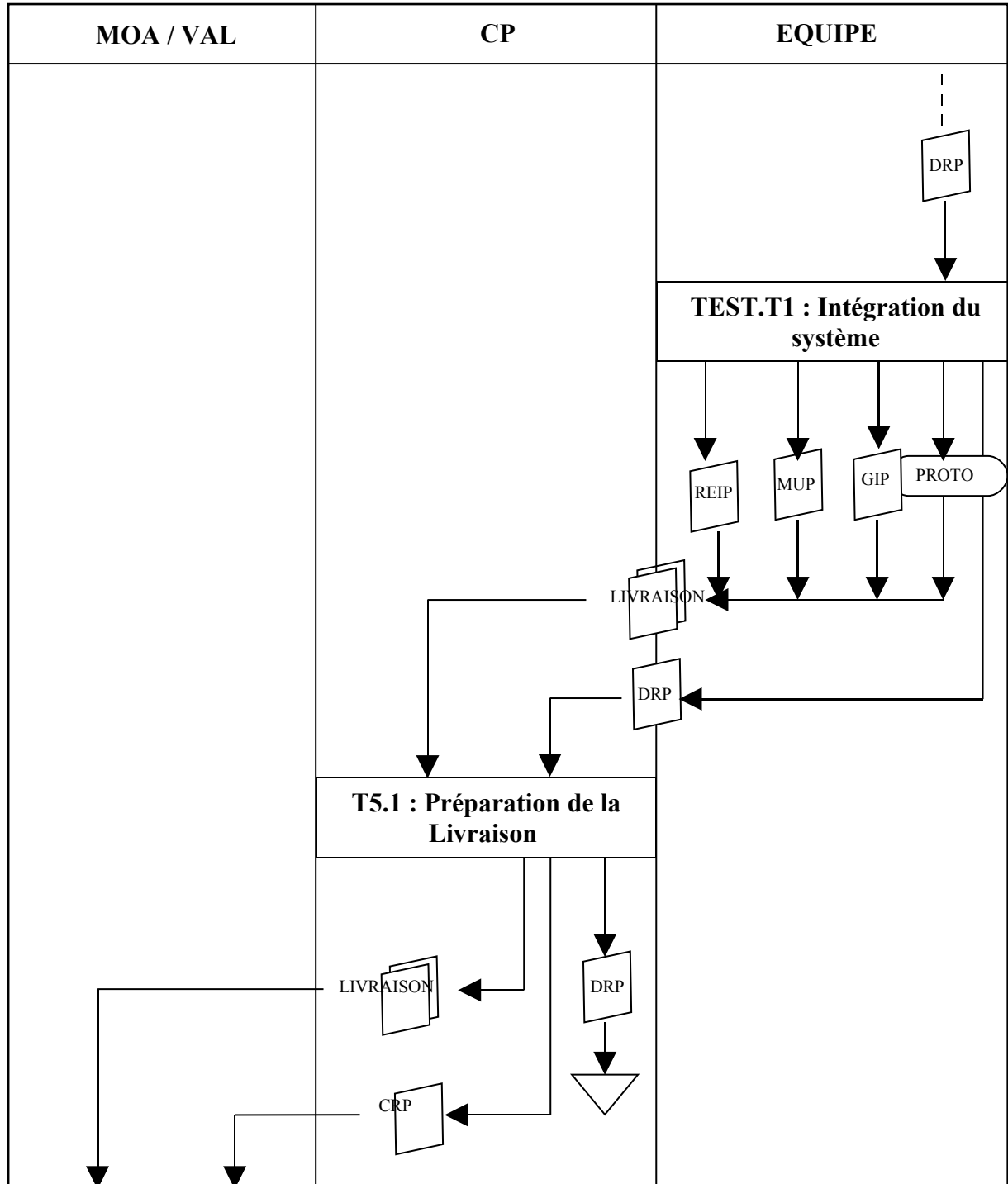


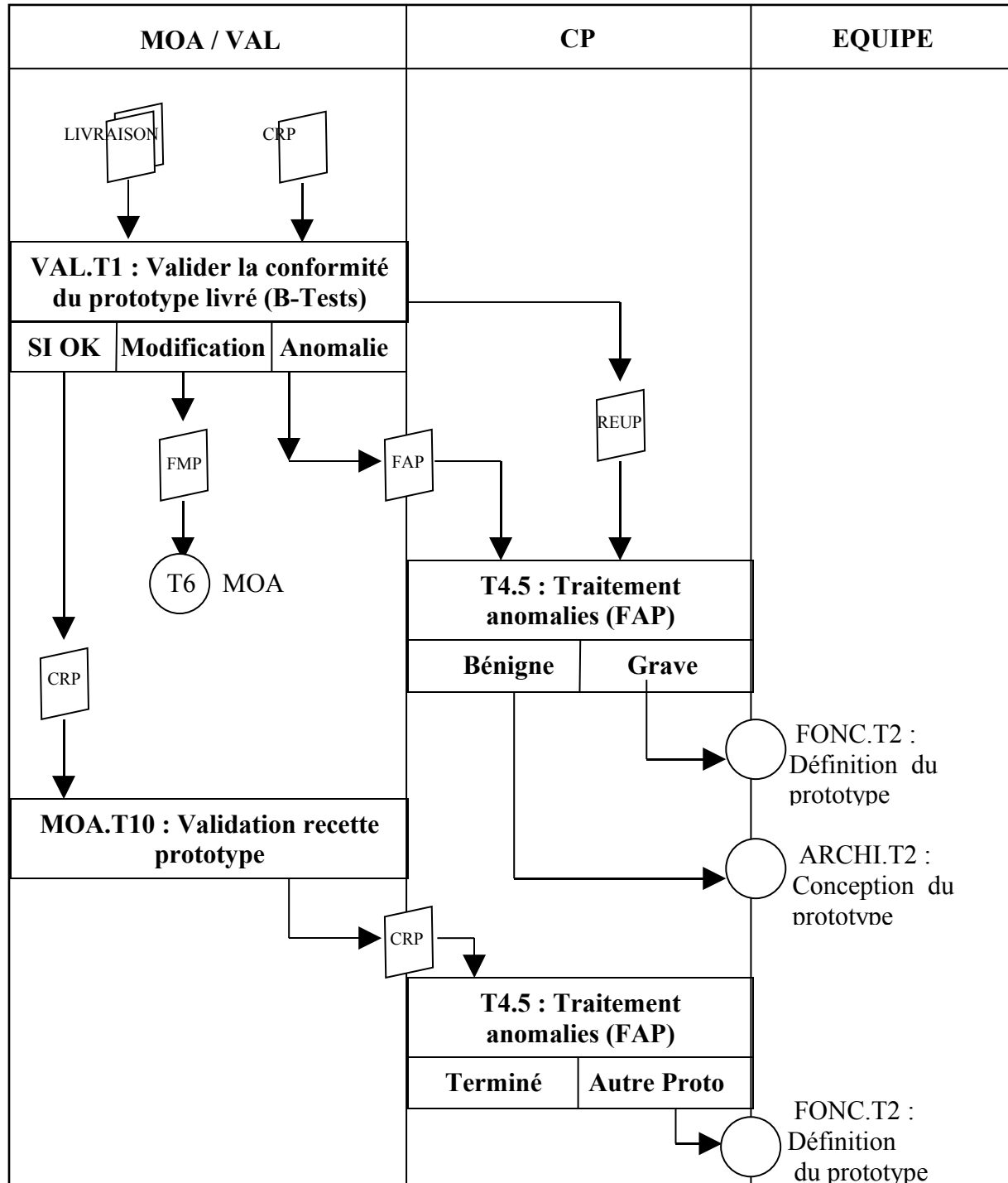












E.V.8 - Les Modèles utilisés par une équipe MACAO :

Modèles	Analyse globale	Conception globale	Développement				Finali- sation
			Définition	Conception	Codage	Intégration	
Use Cases	C	U	M	U			
DCT	C	M	U				
Activité	C	M	U				
Classes		C		M	U	U	M
Objets		C		M			
Composants				C	U	M	U
Déploiement				C		C	U
Séquence	C		C, M	C, M	U		
Collaboration	C		C, M	C, M	U		
Etats		C		C, M	U		
SET				C (B)			U
SNI		C (I)		U			U
MLI				C (I)	U		U

Chapitre VI : Etude de faisabilité

E.VI.1 - WMA CAO projet sur le Web

Le but est d'étudier la faisabilité d'une plate-forme Web d'assistance aux équipes de développement et aux intervenants. Cette plate-forme présentera une aide à l'analyse et conception de systèmes d'information utilisant la méthode MACAO.

Le projet devra prendre en compte la phase de création des prototypes et gérer les droits d'accès aux différents modules déjà programmés afin d'éviter toute régression.

Le développement se fera sur une plate-forme WAMP5 utilisant donc PHP5 (PHP objet), SQL Lite et le serveur HTTP Apache, le tout sous Windows.

E.VI.2 - EMACAO projet sous Eclipse

Pour EMACAO le but est le même mais le développement se fera sous Eclipse, logiciel libre. Il comprend les fonctionnalités suivantes :

- VE pour créer les fenêtres
- Diagramme UML
- Base de Données My SQL
- Environnement de développement Java (JDE)
- 2 bibliothèques graphiques :
 - awt/swing : pour les classes et les fenêtres Java
 - swt/jface : plus rapide mais non portable sur tous les systèmes d'exploitation

E.VI.3 - Conclusion sur le choix des deux technologies

Nous espérons étudier avec ces projets la faisabilité de MACAO sur deux plates-formes techniques différentes. ECLIPSE est largement utilisé dans le monde de la programmation sous JAVA et de nombreux éditeurs adoptent de plus en plus cette plate-forme, je pense à Borland, BEA, et IBM Rational par exemple.

La plate-forme WEB est par nature plus distribuée et peut permettre une collaboration de nombreux intervenants depuis des sites géographiquement éloignés. Des travaux ont déjà été mené dans ce domaine et il existe par ailleurs un langage pour décrire des collaborations entre rôles [151].

Au terme de cette étude, nous avons eu un excellent rendu graphique avec l'interface Web, mais paradoxalement la mise en oeuvre avec EMACAO a donné de meilleurs résultats.

THESE-MACAO	Juin 2008	236 / 266
	Nicolas FERRY	

Chapitre VII : Description des Cas d'Utilisations

Les cas d'utilisation donnent une vue des scénarios informatisés d'une équipe MACAO. Pour la nomenclature, chaque cas utilisation est référencé UC suivi des lettres optionnelles RWDX puis d'un nom détaillé. Une description succincte récapitule ce que doit faire le scénario.

- R signifie que le rôle en charge du scénario peut accéder aux données.
- W signifie que le rôle en charge du scénario peut modifier les données.
- D signifie que le rôle en charge du scénario peut supprimer des données.
- X signifie que le rôle en charge du scénario peut exécuter des traitements spéciaux.

L'indentation permet d'indiquer le niveau de détail d'un cas d'utilisation.

E.VII.1 - UC_CLIENT : Gestion de l'organisation du client

UC_RWD_CLIENT_Personnes : Gestion des personnes
 UC_RWD_CLIENT_Acteurs : Gestion des Acteurs (/Rôles)
 UC_RWD_CLIENT_Utilisateurs : Gestion des utilisateurs
 UC_RWD_CLIENT_Instances : Groupement de personnes pour gérer des instances / Commissions
 UC_RWDX_CLIENT_Décisions : Traçabilité des décisions
 UC_RWD_CLIENT_Réunions : Orchestration de réunions avec clients
 UC_R_CLIENT_Projet : Accès à certains indicateurs de projets
 UC_RWDX_CLIENT_Documents : Gestion des documents (partage, envoi, réception de documents)

E.VII.2 - UC_PROCESS-MACAO : Gestion du processus MACAO

VII.2.1 - UC_PROCESS_CP

UC_RWDX_CP_RessourcesHumaines : Gestion des intervenants internes et externes
 UC_RWDX_CP_Perimetre_Objectifs : Définition du périmètre et des objectifs du projet.
 UC_RWDX_CP_Processus : Définition et gestion du processus mis en œuvre pour atteindre les objectifs. (Organisation du projet)

UC_RWDX_CP_Projet : Gestion du projet
 UC_RWD_Glossaire : Consultation / Ecriture du glossaire métier et technique
 UC_RWDX_CP_Planning : Plannification du projet

THESE-MACAO	Juin 2008	237 / 266
	Nicolas FERRY	

UC_RWDX_CP_Risques : Gestion des risques
 UC_RWDX_CP_Changements : Gestion des changements
 InterfaceCLIENT : Changements Nouveaux Besoins, redéfinition de périmètre
 InterfaceINTERNE : Changements, Congés
 InterfaceEXTERNE : Changement sous-traitants

UC_R_CP_Trajectoire : Evaluation de l'état en cours / Avancement
 (Vision des indicateurs de contrôle du projet)
 UC_RWDX_CP_CR-Revues : Comptes rendus des revues (Comité de pilotage,
 avancement, changements)
 UC_RWDX_CP_Problèmes : Liste des problèmes
 UC_X_CP_DECISIONS : Prise de décision – Directives.

UC_R_CP_Avancement : Historique des Etats du projet à une date précise.
 UC_R_CP_JournalDeBord : Traçabilité des décisions (Journal de bord)
 UC_R_CP_CasRésolus : Historique des solutions mises en œuvre (Best-Practice)

VII.2.2 - UC_PROCESS_ANALYSE

UC_X_ANALYSE_InterviewsUtilisateurs
 UC_RWD_ANALYSE_RédactionCCU : Rédaction du Cahier des Charges Utilisateur si non
 réalisé par la MOA.

UC_RWD_ANALYSE_Système
 UC_RWD_ANALYSE_Besoins : Gestions des besoins (Fonctionnels, non-fonctionnels,
 techniques, Demande de modifications, retour d'anomalies)
 UC_X_ANALYSE_CritiqueExigence : Critique d'une exigence.
 UC_RWD_ANALYSE_Activités : Comprendre et spécifier l'activité de l'entreprise
 (Modélisation métier)
 UC_RWD_ANALYSE_ModélisationUCF : Modélisation des cas d'utilisation
 UC_RWD_ANALYSE_Techniques : Comprendre et spécifier les besoins techniques.
 (Modélis° tech)
 UC_RWD_ANALYSE_ModélisationUCT : Modélisation des cas d'utilisation
 UC_RWDX_ANALYSE_ModélisationIHM
 UC_RW_ANALYSE_Maquette : Réalisation d'une maquette écran (Conception d'IHM)

 UC_X_ANALYSE_NégocierExigences : Négocier les exigences et faciliter
 l'acceptation du client.
 UC_RWD_ANALYSE_Vision : Développement/synthèse d'une vision du projet

THESE-MACAO	Juin 2008	238 / 266
	Nicolas FERRY	

UC_X_ANALYSE_SuiviExigences : Vérifier que les exigences sont respectées et testées.

UC_X_ANALYSE_RespectExigences : Suivi du respect des exigences.

UC_RX_ANALYSE_AnalyseTests : Analyse de tests des exigences.

- UC_RWD_ANALYSE_RedactionDSP : Rédaction du dossier des spécifications.

VII.2.3 - UC_PROCESS_ARCHITECTE

UC_RX_ARCHI_Exigences : Voir la faisabilité des exigences du système.

UC_RWDX_ARCHI_Architecture : Construire l'architecture du système.

UC_X_ARCHI_AnalyseArchitecture : Analyse de l'architecture du système

UC_RWDX_ARCHI_LOGIQUE : Vue Structure du système

UC_RWDX_ARCHI_PROCESSUS : Vue Dynamique du système

UC_RWDX_ARCHI_IMPLEMENTATION : Vue Organisation des composants

UC_RWDX_ARCHI_DEPLOIEMENT : Vue physique de l'environnement

UC_RWDX_ARCHI_DesignPattern : Gestion des DesignPatterns et de la base de composants réutilisables.

UC_WDX_ARCHI_PrototypeFaisabilité : Construire un prototype pour vérifier la faisabilité.

UC_X_ARCHI_SuiviRealisation : Respect du suivi de l'architecture et des objectifs

UC_X_ARCHI_ArbitrageEquipe : Encadrement de l'équipement et arbitrages des décisions.

UC_X_ARCHI_ConseilsTechniquesCP : Conseillent techniquement le Chef de Projet.

UC_X_ARCHI_DecisionsTechniques : Prises de décisions techniques critiques dans les phases de livraison.

UC_RWDX_ARCHI_RédactionDCG : Réalisation du dossier d'architecture ou le Dossier de Conception Globale.

VII.2.4 - UC_PROCESS_DEVELOPPEUR

UC_RX_DEV_Exigences : Consulter les exigences à réaliser

UC_RX_DEV_Contraintes : Consulter les contraintes générales

UC_RWX_DEV_Architecture : Consulter l'architecture globale et pouvoir modifier l'architecture locale (MDA – PSM)

UC_RWX_DEV_Architecture : Demander / Modifier l'architecture (Code et données)

UC_X_DEV_GENERATION_Code : Récréer le code à partir des modèles de l'architecture.

UC_X_DEV_GENERATION_Reverse : Récréer des modèles à partir du code.

UC_RWDX_DEV_CODAGE : Générer le source applicatif.

UC_RWDX_DEV_DEBUGGAGE : Débugger le code source

THESE-MACAO	Juin 2008	239 / 266
	Nicolas FERRY	

UC_RWDX_DEV_INTEGRATION : Intégration de modules (Composants ou avant livraison)

UC_RWDX_DEV_TESTER : Vérifier et passer les apha-tests, génération des Tests unitaires et tests de performances.

VII.2.5 - UC_PROCESS_TESTEUR

UC_RX_TEST_Exigences : Consulter les exigences, les UC et flux à tester.

UC_RWX_TEST_PlanTest : Consultation / Modification et application du plan de test.

UC_RWD_TEST_IdéesTest : Création / Modification d'idées de test.

UC_RX_TEST_RELECTURECODE : Relecture du code source.

UC_X_TEST_DM_CODE : Demande de modification du code source.

UC_RX_TEST_VÉRIFICATIONDATA : Consultation de la base de données.

UC_X_TEST_DM_DATA : Demande de modification de la base de données.

UC_X_TEST_UNIT : Lancer les tests unitaires.

UC_X_TEST_SCRIPT : Lancer les tests fonctionnels

UC_X_TEST_CritiqueCode : Remonter des critiques / demandes aux DEV

UC_PROCESS_QUALITE

UC_R_QUAL_Intervenants : Listes des intervenants sur le projet

UC_RWD_QUAL_Processus : Définition du processus en accord avec le CP

UC_RWD_QUAL_Regles : Rédaction de règles qualités sur le projet

UC_X_QUAL_RedactionPQL : Rédaction du plan qualité

UC_RWX_QUAL_Revues : Revues de conformité au processus défini.

UC_RWX_QUAL_Historique : Historiquement de changement des règles.

VII.2.6 - UC_PROCESS_ERGONOME

UC_R_ERG_Besoins : Lecture des besoins graphiques.

UC_R_ERG_Contraintes : Lecture des contraintes graphiques.

UC_RWDX_ERG_Maquette : Définition de l'aspect graphique.

UC_RWDX_ERG_Ressources : Création et design des ressources graphiques.

UC_RWD_ERG_CharteGraphique : Rédaction de la charte graphique.

UC_RWDX_ERG_Contrôle : Vérification ergonomique du prototype.

THESE-MACAO	Juin 2008	240 / 266
	Nicolas FERRY	

Chapitre VIII : Cas d'utilisation impactés durant l'élaboration d'une IHM

E.VIII.1 - Architecture fonctionnelle

ANALYSTE		
- UC_RWD_ANALYSE_Système		
- UC_RWD_ANALYSE_Besoins : Gestions des besoins (Fonctionnels, non-fonctionnels, techniques, Demande de modifications, retour d'anomalies)		
- UC_RWD_ANALYSE_Activités : Comprendre et spécifier l'activité de l'entreprise (Modélisation métier)		
- UC_RWD_ANALYSE_ModélisationUCF : Modélisation des cas d'utilisation Fonctionnels		
- UC_RWD_ANALYSE_Techniques : Comprendre et spécifier les besoins techniques. (Modélisation technique)		
- UC_RWD_ANALYSE_ModélisationUCT : Modélisation des cas d'utilisation techniques		
- UC_RWD_ANALYSE_ModélisationIHM : Modélisation de l'IHM		
ARCHITECTE		
- UC_RWDX_ARCHI_Architecture : Construire l'architecture du système.		
- UC_RWDX_ARCHI_LOGIQUE : Vue Structure du système		
- UC_RWDX_ARCHI_PROCESSUS : Vue Dynamique du système		
- UC_RWDX_ARCHI_IMPLEMENTATION : Vue Organisation des composants		
- UC_RWDX_ARCHI_DEPLOIEMENT : Vue physique de l'environnement		
DEVELOPPEUR		
- UC_RWX_DEV_Architecture : Demander / Modifier l'architecture (Code et données)		
- UC_X_DEV_GENERATION_Code : Recréer le code à partir des modèles de l'architecture.		
- UC_X_DEV_GENERATION_Reverse : Recréer des modèles à partir du code.		
THESE-MACAO	Juin 2008	241 / 266
	Nicolas FERRY	

Chapitre IX : Réalisation de AGL en paquetages

E.IX.1 - Paquetage Projet

MACAO / Etape MACAO / Phase (en cours)
 Planning (Planning des tâches et événements du projet)
 Personnes (Personnes concernées par ce projet)
 Acteurs (Intervenants connectés à un rôle)
 Contacts (Liste des personnes liées au projet)
 Tâches (=Agent)
 Décisions (Traçabilité des décisions)
 Risques (Tableau des risques)
 Vision (Description du but du projet / Direction)
 Indicateurs (Contrôle de l'état du projet)
 Qualité (Vision du plan qualité)
 Glossaire (Glossaire du projet)
 Fichiers (Index des ressources fichiers)

E.IX.2 - Paquetage Client / Organisation

Personnes (Intervenants côté client)
 Acteurs (Rôle joué sur le projet)
 Commission (groupement hiérarchique de personnes décisionnelles)
 Décisions (Notification des décisions importantes prises sur le projet)
 Instances (instances compétentes ou externes)

E.IX.3 - Paquetage Architecture / Process

IX.3.1 - Analyse

Besoins (Requiements Emis et traduits)
 Acteurs (Acteurs Modéliser)
 Paquetages (Fonctionnels et Techniques)
 UC (Cas d'utilisation)

IX.3.2 - Conception

Prototypes
 Fonctions
 IHM

THESE-MACAO	Juin 2008	242 / 266
	Nicolas FERRY	

IX.3.3 - Réalisation

Développement
Composants
Classes
Attributs
Méthodes
Tests Fonctionnels (Tests non-régression)
Tests Unitaires
Version / Lot

IX.3.4 - Finalisation

Recette
Archivage des sources

E.IX.4 - Paquetage Documents - Traçabilité des documents

CCU (Cahier de charges de l'utilisateur)
DSP (Dossier de spécification)
DCG (Dossier de conception globale)
PDV (Plan de développement)
PQL (Plan Qualité)
DDP (Dossier de définition Prototype)
DRP (Dossier de réalisation Prototype)
PEP (Plan d'Essai Prototype)
DMP (Demandes de Modification Prototype)
REIP (Rapport d'Essai d'Intégration Prototype)
MUP (Manuel Utilisateur Prototype)
GIP (Guide d'Installation Prototype)
FAP (Fiche d'Anomalies Prototype)
FMP (Fiche de Modifications Prototype)
REUP (Rapport d'Essai Utilisateurs Prototypes)
CRP (Certificat de Recette Prototype)
MU (Manuel Utilisateur)
GI (Guide d'Installation)
FS (Fiche de Signalement)
DR (Dossier de Réalisation)

L'atelier de génie logiciel MACAO permettra de gérer des projets en appliquant les grandes lignes de la démarche MACAO. Sa réalisation constitue un défi valorisant pour l'avenir.

THESE-MACAO	Juin 2008	243 / 266
	Nicolas FERRY	

Bibliographie

- [1] Jean-Bernard CRAMPES. Méthode Orientée-objet intégrale MACAO. 2003.Ellipses. 2-7298-1428-8.
- [2] Levenez Evolution des langages <http://www.levenez.com/lang/history.html>
- [3] Grady Booch, James Rumbaugh, Ivar Jacobson. Le guide de l'utilisateur UML. ... 2212091036.
- [4] Franck barbier. UML2 et MDE, Ingénierie des modèles avec études de casISBN 9782100495269.
- [5] Craig Larman. UML2 et les Design Pattern. ...2744070904.
- [6] Martin Fowler et al.. UML 2. 2004..2744017132.
- [7] Pascal Roques. UML 2 - Modéliser une application Web2-212-12136-9.
- [8] Pascal Roques. UML 2 par la pratique. 2006..2-212-12014-1.
- [9] larry Constantine & Lucy Lockwood. Software for use. 1999.7th, 2006.0201924781.
- [10] AndroMDA 2004 <http://www.andromda.org>
- [11] Aditya AGRAWAL, Tihmer LEVENDOVSKY, Jon SPRINKLE, Feng Shi et Gabor KARSAI. Generative Programming via Graph Transformation in the Model-Driven Architecture. 2002.OOPSLA 2002 WORKSHOP on generative Techniques in the context of Model Driven Architecture
- [12] Aditya AGRAWAL, Tihmer LEVENDOVSKY, Jon SPRINKLE, Feng Shi et Gabor KARSAI. Generative Programming via Graph Transformation in the Model-Driven Architecture. 2002.OOPSLA 2002 WORKSHOP on generative Techniques in the context of Model Driven Architecture
- [13] Biju APPUKUTTAN, Tony CLARK, Sreedhar REDDY, Laurence TRATT et R. VENKATESH. A Model Driven Approach to Model Transformations.. 2003.Workshop on Model Driven Architecture : Foundation and Application (MDAFA2003)

THESE-MACAO	Juin 2008	244 / 266
	Nicolas FERRY	

- [14] Biju APPUKUTTAN, Tony CLARK, Sreedhar REDDY, Laurence TRATT et R. VENKATESH. A Pattern based Model-Driven Approach to Building Model Transformations.. 2003.Metamodelling for MDA 2003
- [15] Jean BEZIVIN, INRIA, LINA et Université de Nantes ATL Development Tools 2004 <http://www.sciences.univ-nantes.fr/lina/atl>
- [16] Jorn BETTIN, Ghica van EMDE Boas et E.D. WILLINK.. Generative Model Transformer: An Open Source MDA Tool Initiative.. 2003.Companion of the 18th annual ACM SIGPLAN conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2003)
- [17] Grady BOOCH, Alan BROWN, Sridhar IYENGAR, James RUMBAUGH et Bran SELIC.. An MDA manifest.. 2004.MDA Journal
- [18] Peter BRAUN et Franck MARSCHALL. BOTL The Bidirectionnal Object Oriented Transformation Language. 2003.Institut fur Informatik, Technische Universitat Muchen.
- [19] Franck BUDINSKY, David STEINBERG, Ed MARKS, Raymond ELLERSICK et Timothy J. GROOSE. Eclipse Modeling Framework: A Developer's Guide.. 2003.Addison-Wesley Pub Co..
- [20] Jean BEZIVIN et olivier GERBE. Towards a Precise Definition of the OMG/MDA Framework.. 2001.In Proceedings of the 16th international Conference of automated Software Engineering
- [21] jean BEZIVIN et Frédéric JOUAULT et Patrick VALDURIEZ. First Experiments with a ModelWeaver.. 2004.OOPSLA 2004 - Workshop of Best Pratices for Model Driven Software Development,
- [22] jean BEZIVIN et Frédéric JOUAULT et Patrick VALDURIEZ. On the need of Megamodels. 2004.
- [23] guy CAPLAT et Jean Louis SOURROUILLE. Model Mapping in MDA. 2002.Workshop in Software Model Engineering (WISME2002)
- [24] guy CAPLAT et Jean Louis SOURROUILLE. Considerations about Model Mapping. 2003.Workshop in Software Model engineering

THESE-MACAO	Juin 2008	245 / 266
	Nicolas FERRY	

- [25] CARL-OLAV.. MDA as Framework for Supporting Enterprise Standards.. 2004.Practitioner's Reports ECOOP 2004
- [26] CODAGEN Codagen Architect. 2004 <http://www.codagen.com/products/architect/>
- [27] COMPUWARE CORPORATION. 2004
<http://www.compuware.com/products/optimalj/>
- [28] Steve COOK. Domain Specific Modeling and Model Driven Architecture.. 2004.MDA journal
- [29] Lisa CRISPIN et Tip HOUSE. Testing Extreme Programming. 2002.Addison-Wesley..
- [30] DSTC, IBM et CBOP. MOF Query / View / Transformations. 2004.
- [31] ECLIPSE TOOLS PROJECT Eclipse Modeling Framework (EMF) version 2.2 2008
<http://www.eclipse.org/emf/>
- [32] ECLIPSE TOOLS PROJECT Graphical Editing Framework (GEF) version 3.2 2008
<http://www.eclipse.org/gef/>
- [33] E.D WILLINK. UMLX - A Graphical Transformatio language for MDA.. 2003.Workshop on Model Driven Architecture Foundations and Applications
- [34] Jean-Marie FABRE. Towards a Basic Theory to Model Driven Engineering.. 2004.UML2004 - Worshop in Software Model Engineering(WISME 2004)
- [35] Frédéric FONDEMENT et Raoul SILAGHI. Defining Model Driven Engineering Processes. 2004.WiSME@UML2004
- [36] David FRANKEL. Model Driven Architecture : Applying MDA to Enterprise Computing. 2003.Wiley and OMG Press..
- [37] Eric GAMMA et Kent BECK. Contributing to Eclipse : Principles, Patterns and Plug-ins. 2003.Addison-Wesley Pub Co..
- [38] Tracy GARDNER, Catherine GRIFFIN, Jana KOEHLER et Rainer HAUSER. A review of OMG MOF 2.0 Query / View / Transformations Submissions and Recommendations towards the final Standard. 2003.

THESE-MACAO	Juin 2008	246 / 266
	Nicolas FERRY	

- [39] Anatasius GAVRAS, Mariano BELAUNDE, Luis FERREIRA PIRES et João Paulo A. ALMEIDA. Towards an MDA-based development methodology. 2004.First European Workshop on software Architecture (EWSA 2004)
- [40] Anna GERBER, Michel LAWLEY, Kerry RAYMOND, Jim STEEL et Andrew WOOD. Transformation : the Missing Link of MDA. 2002.First International Conference on Graph Transformation (ICGT 2002)
- [41] Jan Hendrick HAUSMANN et Stuart KENT. Visualizing Model Mappings in UML. .
- [42] IBM Rational Rose <http://www-306.ibm.com/software/awdtools:developer/rose/>
- [43] IBM <http://www-306.ibm.com/software/awdtools:developer/rosexde/>
- [44] INRIA ATL Transformation Example : Class to Relational 2005
- [45] INTERACTIVE OBJECTS Software GMBH and Project Technology, INC 2nd Revised Submission to MOF Query / view / Transformation RFP 2004
<http://www.omg.org/docs/ad/04-01-14.pdf>
- [46] Jana KOEHLER, Jos WARMER et Wim BAST. MDA explained: The Model Driven Architecture: practice and Promise. 2003.Addison-Wesley..
- [47] Audris KALNINS, Janis BARZDINS et Edgars CELMS. Basics of Model Transformation Language MOLA. 2004.Workshop on Model Driven Development (WMDD2004)
- [48] Stuart KENT et Robert SMITH. The Bidirectional Mapping Problem. 2003.Electronic Notes in Theoretical Computer Sciences
- [49] Stephen J. MELLOR, Kendall SCOTT, Axel UHL et Dirk WEISE. MDA distilled: Principles of Model-Driven Architecture. 2004.Addison-Wesley..
- [50] MIA Model-In-Action 2008 <http://www.mia-software.com>
- [51] Pierre-Alain MULLER et Nathalie GAERTNER. Modélisation Objet avec UML. 2004.Eyrolles..
- [52] OMG Model Driven Architecture (MDA) 2001

THESE-MACAO	Juin 2008	247 / 266
	Nicolas FERRY	

- [53] OMG Ratinal Unified Process 2004
- [54] OMG Unief Modeling Language Specification, Version 1.4 2001
- [55] OMG Unified Modeling Language : Superstructure, Version 2.0 Final Adopted Specification 2003
- [56] OMG Meta Object Facility (MOF) Specification. 2002
- [57] OMG MOF 2.0 Query/View/transformation RFP
- [58] OMG XML Metadata Interchange (XMI) Specification Version 1.2 2002
- [59] OMG MDA Guides 2003
- [60] OMG Object Constraint Language (OCL) Specification 2003
- [61] OMG XML MetaData Interchange (XMI) Specification, Version 2.0 2003
- [62] OMONDO Omondo Eclipse UML 2004 <http://www.omondo.com>
- [63]: Mickael PELTIER, Techniques de transformation de modèles basées sur la méta-modélisation, 2003
- [64] QVT-Merge Group Revised submission for MOF 2.0 Query/view/transforms RFP 2004
- [65] Bill MOORE, David Dean, Ana GERBER, Gunnar WAGENKNECHT, Philippe VANDERHEYDEN Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework February 2004
<http://www.redbooks.ibm.com/abstracts/sg246302.html>
- [66] Koen Aers. A Gentle Introduction to GEF. 2006EclipseCON 2006
- [67] Eclipse GEF project Eclipse GEF API Specification 2006
- [68] IBM Create an Eclipse-based application using the Graphical Editing Framework 2006
<http://www-128.ibm.com/developerworks/opensource/librayry/os-gef>

THESE-MACAO	Juin 2008	248 / 266
	Nicolas FERRY	

- [69] Janak MULANI, Sibylle PETER. Implementation of ULC Visual Editor for Eclipse. 2006
- [70] Object Technology International, Inc. Eclipse Platform Technical Overview. February 2003
- [71] Catherine GRIFFIN (IBM). Using EMF. December 2002.
- [72] ZDot Eclipse : Standardizing plugins across your team 2004
<http://timshadel.com/blog/2004/08/28/1093715447000.html>
- [73] David ORME. Eclipse RCP vs RCPLite vs SWT. Juillet 2004
http://www.coconut-palm-software.com/the_visual_editor/2004/07/23#when_rcp
- [74]: Gamma, Helm, jhonson/vlissides, Design patterns, 1995
- [75] IBM alphaworks IBM Reflexive User Interface Builder 2006
<http://www.alphaworks.ibm.com/tech/rib>
- [76]: Laszlo Systems, Increasing Profitability by improving User Experinece (RIA), 2003
- [77] A. VIVIEN et P. BONNEMAIS. Guide de mise en oeuvre de MAVEN sur ADC. 2004
- [78] Adobe FLEX Adobe Flex FAQ (RIA)
<http://www.adobe.com/products/flex/productinfo/faq/>
- [79] Arnaud BUISINE, SYSDEO. Struts, vous avez dit Struts ?. 2002
- [80] Philippe DESFRAY. réussir mla modélisation UML des phases préliminaires.
<http://softeam.com>
- [81] philippe DESFRAY, SOFTEAM. MDA - When a major software industry trend meets our toolset, implemented since 1994. 2001.
- [82] SOFTEAM. Guarantee permanent Model/Code consistency : "Model Driven Engineering" (MDE) versus "Roundtrip engineering" (RTE). 2000White paper
- [83] Peter CHEN et Hubert TARDIEU Méthode Merise
sqlpro.developpez.com/cours/modelisation/merise/

THESE-MACAO	Juin 2008	249 / 266
	Nicolas FERRY	

[84] IBM, Rational Rational Unified Process www-306.ibm.com/software/awdtools/rup/

[85] Kent Beck, Ward Cunningham et Ron Jeffries. eXtreme Programming
<http://www.c2.com/cgi/wiki?ExtremeProgramming>

[86] Pascal Roques, Franck Vallee. UML en action. 2002.Eyrolles.2212112130.

[87] WebML www.webml.org/

[88] Paulo Pinheiro da Silva, Norman W. Paton UMLi <http://trust.utep.edu/umli/>

[89] Linux API www.unix.org/apis.html

[90] Microsoft ,Net Framework msdn.microsoft.com/netframework/

[91] Plateforme Java www.java.com/

[92] Microsoft Visual Studio msdn.microsoft.com/vstudio/

[93] IBM Websphere Eclipse www.eclipse.org

[94] BEA Webloogic www.bea.com/

[95] Borland Suite www.borland.fr/products/

[96] JetBrains IntelliJ www.intellij.com/idea/

[97] SUN Java Studio www.sun.com/software/sundev/familycomp

[98] Exadel STRUTS Studio <http://exadel.com/>

[99] PCSoft WinDev / WebDev www.pcsoft.fr/

[100] Bolrand Together <http://www.borland.com/fr/products/together/index.html>

[101] IBM Rational Atlantic sur Eclipse
www-306.ibm.com/software/rational/announce/oct-2004-fr/index.html

[102] Entreprise Architect www.sparxsystems.com

THESE-MACAO	Juin 2008	250 / 266
	Nicolas FERRY	

- [103] Omondo UML Eclipse Studio www.omondo.com
- [104] Gentleware Poseidon www.gentleware.com
- [105] Argo UML www.argouml.org
- [106] I-Logic Rhapsody Modeler www.ilogix.com
- [107] Envision Case France www.case-france.com
- [108] Sybase PowerAMC www.osinet.fr/brands/sybase/poweramc.htm
- [109] Telelogic TAU www.telelogic.com/products/tau/index.cfm
- [110] Ruby on Rails <http://www.rubyonrails.org/>
- [111] AJAX http://fr.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML
- [112] Java Web Start java.sun.com/products/javawebstart/
- [113] Adobe Acrobat Designer <http://www.adobe.com/products/lifecycle/designer/>
- [114] Macromedia FLEX www.macromedia.com/software/flex/
- [115] W3C XForm www.w3.org/MarkUp/Forms/
- [116] Eclipse SWT <http://www.eclipse.org/swt/>
- [117] XUL www.mozilla.org/projects/xul/
- [118] XULfr <http://xulfr.org/>
- [119] XAML <http://www.xaml.fr/>
- [120] Laszlo (LZX) www.laszlosystems.com/
- [121] Open Laszlo www.openlaszlo.org/
- [122] Eclipse RCP <http://www.eclipse.org/ercp/>

THESE-MACAO	Juin 2008	251 / 266
	Nicolas FERRY	

- [123] Article sur le Client Riche ZDNET
<http://www.zdnet.fr/actualites/informatique/0,39040745,39150184,00.htm>
- [124] MSDN Library <http://msdn.microsoft.com/library/default.asp>
- [125] Article SWT www.eclipse.org/articles/Article-SWT-Design-1/SWT-Design-1.html
- [126] Win32 API map to .Net API <http://msdn2.microsoft.com/en-us/library/aa302340.aspx>
- [127] Evolution de Rational sur Eclipse <http://www.itrmanager.com/article.php?oid=31406>
- [128] WebML : Web Modeling Language <http://www.webml.org>
- [129] Baron, M. Lucquiaud, v. Autard, D. Scapin, DL.. K-MADe : un environnement pour le noyau du modèle de description de l'activité. 2006.Proceedings of the 18th French-speaking conference on Human-Computer interaction, Montreal, Canada, April 18-21
- [130] Alexandre CORTIER, Thèse : Vers une méthodologie de Validation et de Vérification multi-formelle des Interfaces Utilisateurs en milieu industriel. ONERA-DTIM, Supaéro, Toulouse.
- [131] Philippe Palanque, Rémi Bastide. Spécifications formelles pour l'ingénierie des interfaces homme-machine. 1995.
- [132] Jean-Bernard CRAMPES, Laurent Nonne, Nicolas FERRY. Ingénierie d'utilisabilité, capture des exigences pour l'IHM. 2005.Actes AIM-2005
- [133] Thierry Duval, Jean-Claude Tarby. Améliorer la conception des applications interactives par l'utilisation conjointe du modèle PAC et des patrons de conception. 2006.Acte IHM-2006
- [134] Cédric Bach, Pascal Salembier, Emmanuel Dubois. Co-Conception d'expériences interactives augmentées dédiées aux situations muséales . 2006.Acte IHM-2006
- [135] Camille Grange, Henri Barki. Développement d'un outil de mesure de l'utilisabilité des intranets . 2006.Acte IHM-2006
- [136] Sami Baffoun, Jean-Marc Robert . État de l'art des techniques de présentation d'information sur écran d'assistant numérique personnel . 2006.Acte IHM-2006

THESE-MACAO	Juin 2008	252 / 266
	Nicolas FERRY	

- [137] Stéphane Huot, Pierre Dragicevic, Cédric Dumas. Flexibilité et modularité pour la conception d'interactions : Le modèle d'architecture logicielle des Graphes Combinés. 2006. Acte IHM-2006
- [138] Caroline Appert, Michel Beaudouin-Lafon. SMCanvas : augmenter la boîte à outils Java Swing pour prototyper des techniques d'interaction avancées . 2006. Acte IHM-2006
- [139] Yamine AIT-AMEUR, Idir AIT-SADOUNE, Mickael BARON, Jean-Marc MOTA. Validation et Verification Formelles de Systemes Interactifs Multi-Modaux Fondees sur la Preuve. 2006. Acte IHM-2006
- [140] Arnaud Lewandowski, Grégory Bourguin, Jean-Claude Tarby. Vers des Composants Logiciels Orientés T,ches. 2006. Acte IHM-2006
- [141] Tomas Dorta. Vers la maîtrise du virtuel à travers le réel : un nouvel usage de l'informatique en design. 2006. Acte IHM-2006
- [142] Cyril ROUSSEAU, Yacine Bellik, Frédéric Vernier. WWHT : un modèle conceptuel pour la présentation multimodale d'information. 2005. Acte IHM-2005
- [143] Vincent Lucquiaud. Proposition d'un noyau et d'une structure pour les modèles de tâches orientés utilisateurs. 2005. Acte IHM-2005
- [144] Renaud BLANCH, Michel Beaudouin-Lafon, Stéphane Conversy, Yannick Jestin, Thomas Baudel, Yun Peng Zhao. INDIGO : une architecture pour la conception d'applications graphiques interactives distribuées. 2005. Acte IHM-2005
- [145] Arnaud Lewandowski, Grégory Bourguin. Gestion de l'inter-activités pour le support au développement logiciel coopératif. 2005. Acte IHM-2005
- [146] Faouzi Moussa, Christophe Kolski, Meriem Riahi. De la modélisation des dysfonctionnements d'un système complexe à la déduction des besoins informationnels des utilisateurs : une transition difficile en IHM. 2005. Acte IHM-2005
- [147] Marco Winckler, Christelle Farenc, Eric Barboni, Florence Pontico. Modélisation orientée tâche de la navigation d'une application WEB : catalogue des thèses de l'AFIHM. 2005. Acte IHM-2005

THESE-MACAO	Juin 2008	253 / 266
	Nicolas FERRY	

- [148] Ludovic Le Bigot, Valérie Botherel, Eric Jamet. Asymétrie du transfert modal lors d'un dialogue personne-machine. 2005.Acte IHM-2005
- [149] Gilles Halin, Sylvain Kubicki. Architecture dirigée par les modèles pour une représentation multi-vues du contexte de coopération. 2005.Acte IHM-2005
- [150] Alan F. NEWELL, Anna DICKINSON, Mick J.SMITH, Peter GREGOR. Designing a Portal for Older Users : A Case Study of an Industrial/Academic Collaboration. 2006.ACM TransactionsonComputer-Human Interaction,Vol.13
- [151] Du Li, Richard R. Muntz. A Collaboration Specification Language. 2000.ACM Press
- [152] D.SCOTT, McCRICKARD, C.M.CHEWAR, JACOB P.SOMERVELL, ALI NDIWALANA. A Model for Notification Systems Evaluation - Assessing User Goals for Multitasking Activity. 2003.ACM Transactions on Computer-Human Interaction,Vol.10,No.4
- [153] Sitt SENCHOK, Kim MARRIOTT. Automatic Generation of Intelligent Diagram Editors. 2003.ACM Transactions on Computer-Human Interaction,Vol.10,No.3
- [154] Mary Czerwinski Microsoft Research. Bridging the Gap From Theory to Practice:The Path Toward Innovation in Human-Computer Interaction. 2004.UIST04, New Mexico, USA
- [155] CONSTANTINEE. STERIADIS, PHILIP CONSTANTINOU. Designing Human-Computer Interfaces for Quadriplegic People. 2002.
- [156] POURANG IRANI, COLIN WARE. Diagramming Information Structures Using 3D Perceptual Primitives. 2003.ACM Transactions on Computer-Human Interaction,Vol.10,No.1
- [157] Tim Sheard, Zine-el-abidine Benaissa, Emir PasMic. DSL Implementation Using Staging and Monads. 2000.
- [158] Jeffrey Nichols, Brandon Rothrock, Duen Horng Chau, Brad A. Myers. Huddle: Automatically Generating Interfaces for Systems of Multiple Connected Appliances. 2006.UIST06, Montreux, Switzerland
- [159] Kasper HORNBAEK, Benjamin B.BEDERSON, Catherine PLAISANT. Navigation Patterns and Usability of Zoomable User Interfaces with and without an Overview. 2002.ACM Transactions on Computer-Human Interaction, Vol.9,No.4

THESE-MACAO	Juin 2008	254 / 266
	Nicolas FERRY	

- [160] Scott E. Hudson, Jennifer Mankoff. Rapid Construction of Functioning Physical Interfaces from Cardboard, Thumbtacks, Tin Foil and Masking Tape . 2006.UIST06, Montreux, switzerland
- [161] Brian Willard (Lockheed Martin Aeronautics). UML for systems engineering. 2006.
- [162] Wolfgang Stuerzlinger, Olivier Chapuis, Dusty Phillips, Nicolas Roussel. User Interface Facades : Towards Fully Adaptable User Interfaces. 2006.UIST06, Montreux, switzerland
- [163] OpenArchitectureWare www.openarchitectureware.org
- [164] OpenArchitectureWare www.eclipse.org/gmt/oaw
- [165] Jean-Claude Tarby, Xavier Le Pallec, Raphael Marvie, Mirabelle Nebut. Processus de modélisation incrémental pour le développement d'applications interactives basées sur PAC. 2006.Publication interne Lift
- [166] Raphael Marvie, Mirabelle Nebut. Processus de Modélisation incrémentaux. 2006. IDM 2006
- [167] Joëlle Coutaz. Interface homme-ordinateur, conception et réalisation. 1999.Dunod Informatique.2040196358.
- [168] Ivar Jacobson, Grady Booch, James Rumbaugh. The Unified Software Development Process. 1999.Addison-Westley.2040196358.
- [169] Th. Reiter, E. Kapsammer, W. Retschitzegger, W. Schwinger. Model Integration through megaoperations. 2005.Workshop on model-driven Web Engineering (MDWE 2005)
- [170] Olivier Barais, Franck Fleurey, Pierre-Alain Muller, Didier Vojtisek, Jean-Marc Jézéquel. Nouvelles fonctionnalités de kermeta. 2007.Session Démonstrations des 3 ème Journées sur l'Ingénierie Dirigée par les Modèles, March 2007
- [171] Fabio Paterno. Model-Based Design and Evaluation of Interactive Applications. Springer-Verlag, London, UK, 1999.
- [172] Paternò, F. ConcurTaskTrees: An Engineered Notation for Task Models, Chapter 24, in Diaper, D., Stanton, N. (Eds.), The Handbook of Task Analysis for Human-Computer Interaction, pp. 483-503, Lawrence Erlbaum Associates, 2003.

THESE-MACAO	Juin 2008	255 / 266
	Nicolas FERRY	

[173] P. Girard Y. Aït-Ameur and F. Jambon. A uniform approach for the specification and design of interactive systems : the B method. In In Panos Markopoulos and Peter Johnson editors, editors, In proceedings DSV-IS'98, pages 333–352, Abingdon, UK, 1998.

[174] P. Girard Y. Aït-Ameur and F. Jambon. Using the B formal approach for incremental specification design of interactive systems. In Stéphane Chatty and Prasun Dewan, editors, Engineering for Human-Computer Interaction.

[175] F. Adreit et S-A. Mahé. La prise en compte du contexte dans le développement d'un système logiciel interactif : Apports et limites de la Systémique et nouvelles perspectives. *Dans actes 5th European Systems Sciences Congress* , Heraklion. 10 Octobre 2002.

[176] Limbourg Q., Vanderdonckt J. usiXML, User Interface eXtensible Markup Language. Site : www.usiXML.org

[177] Limbourg Q., Vanderdonckt J., Michotte B., Bouillon L., Lopez-Jaquero, V., "UsiXML: a Language Supporting Multi-Path Development of User Interfaces", Working Conference on Engineering for Human-Computer Interaction, 2004.

[178] Hubert Garavel, Thèse : Présentation du langage LOTOS1

[179] Joëlle Coutaz, Gaëlle Calvary, HCI and Software Engineering: Designing for User Interface Plasticity. In The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications. Pages 1107-1125. 2008. Second Edition, ISBN 9780805858709, Taylor & Francis CRC Press, Human Factor and Ergonomics series, A. Sears, J. Jacko Eds. <http://www.isrc.umbc.edu/HCIHandbook/>

[180] Jean-Bernard CRAMPES, Consultant Méthode MACAO. Site : <http://www.jbcc.fr>

THESE-MACAO	Juin 2008	256 / 266
	Nicolas FERRY	