N° Identification du projet : **99008325**

Date : 12/12/2001

PROCESSUS DE DEVELOPPEMENT BASE SUR UML

Livraison				
	Nom	**Partenaire**	**Date**	**Signature**
Rédigé par	Antoine Jammes	CS SI	12/12/2001	
Approuvé par	Jean-Yves SOUILLARD	CS SI	12/12/2001	
Pour application				

SUIVI DES MODIFICATIONS

Version / Révision		**Références**		**Description de**	**Nom de**
Indice	*Date*	*N° page*	*N° §*	**La modification**	**l'auteur**
1.0	30/03/2001			Version initiale	CS SI
1.1	07/08/2001			Premières Modifications	CS SI
2.0	12/12/2001			Nouvelle version : restructuration des chapitres document complété.	CS SI
3.0	21/12/2001			Ajout de l'analyse de l'existant	CS.SI IRIT

SOMMAIRE

1 INTRODUCTION

Aujourd'hui, UML apparaît comme la technique de modélisation objet la plus répandue pour un vaste éventail d'applications logicielles. Toutefois, UML est un langage, non une méthode. La définition d'un processus est donc nécessaire pour optimiser son application sur les projets.

Ce guide a comme objectif de formaliser un processus de développement utilisant UML comme technique de modélisation pour l'analyse et la conception. Le processus décrit ici, appelé UML/MERCURE, se veut simple. Il a été élaboré sur la culture de l'entreprise CS SI dans l'emploi de processus clairement définis pour le développement logiciel, en collaboration avec l'IRIT.

Ce processus se décompose en trois phases qui elles-mêmes se décomposent en une quinzaine d'activités au total. D'autre part :

- Les flots d'information entre les activités sont identifiés.
- Les activités sont détaillées de manière à faciliter leur exécution sur les projets.
- Les rôles intervenant dans les différentes phases et activités sont aussi présentés ; cela permet d'allouer aisément les ressources humaines en fonction de l'état d'avancement des projets et de leur expertise.
- La production de documentation est aussi abordée, et une proposition de catalogue des documents à produire dans les projets est proposée par défaut.

Les bénéfices attendus de l'application de ce guide sur les projets sont multiples lors de l'exécution des projets, comme par exemple :

- Augmentation de la qualité (*la modélisation UML formalise l'analyse et la conception*),
- Augmentation de la productivité (*les modèles UML sont convertis directement en code*),
- Meilleure maîtrise dans l'évaluation de l'état d'avancement des développements (*on sait ce que l'on doit faire et donc ce qu'il reste à faire*),
- Meilleure communication entre les équipes et leurs clients (*les phases, activités et fournitures attendues sont connues des deux côtés dès le démarrage du projet*).

Ce guide est aussi une arme efficace pour gagner des contrats. Il offre un cadre rigoureux pour élaborer les propositions techniques, et il crédibilise les réponses techniques car il agit en tant que référence d'une Société démontrant aux clients l'expertise des équipes dans le développement logiciel.

Pour l'entreprise et ses collaborateurs, les gains sont réels en termes de capitalisation du savoir-faire et de formation des collaborateurs aux meilleures pratiques du développement logiciel et de la conduite de projet.

Le seul pré-requis pour une parfaite compréhension du processus se limite à la connaissance de la notation UML.

2 DOCUMENTS APPLICABLES ET DOCUMENTS DE REFERENCE

2.1 DOCUMENTS APPLICABLES

DA1	MER-GP-PQ	Plan qualité du projet MERCURE

2.2 DOCUMENTS DE REFERENCE

DR1 : 99008325 MEthode et moyens Reliés aveC UML pour la coopération pluridisciplinaire (MERCURE)

3 TERMINOLOGIE

3.1 GLOSSAIRE

AFNOR	Association Française de NORmalisation
HTML	HyperText Markup Language
IRIT	Institut de Recherche Informatique de Toulouse
MERCURE	MEthode et moyens Reliés aveC UML pour la coopération pluridisciplinaiRE
OCL	Object Constraints Language
OMG	Object Management Group
UML	Unified Modelling Language
UPS	Université Paul Sabatier (France/Toulouse)
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language

3.2 DEFINITIONS

Termes de la méthode UML/MERCURE

TERME / EXPRESSION	TRADUCTION ANGLAISE	DEFINITION
Acteur	Actor	Un acteur représente un rôle tenu par un utilisateur, un processus logiciel ou une machine (ordinateur, périphérique), qui interagit avec le système à modéliser, et en déclenchant des fonctions du système.
Entité passive	Passive entity	Une entité passive est un acteur (au sens UML) qui ne déclenche pas des fonctions du système (une imprimante, un processus produisant des données pour notre système mais sans communication directe - ex. : production et dépôt d'un fichier dans un espace X, associé au système, dont le nom est configurable).
Modèle de conception métier	Business pattern	Un modèle de conception (pattern) créé sur les bases des projets opérationnels réalisés au sein de l'entreprise.

Phases du processus

TERME / EXPRESSION	TRADUCTION ANGLAISE	DEFINITION
Analyse des besoins	Requirements analysis	Cette phase permet de formaliser les besoins des utilisateurs. Elle regroupe les activités A, B et C.
Analyse objet	Object analysis	Cette phase permet d'identifier les objets du domaine. Elle est couverte par l'activité D.
Conception de l'architecture	Architectural design	Cette phase a pour objectif d'identifier, de structurer et de décrire les composants logiciels du système. Elle regroupe les activités E, F, G et H.
Conception objet	Object design	Cette phase couvre la conception détaillée des paquetages et des classes du système, et prend en compte les spécificités du logiciel (orienté-données, orienté-IHM, etc.). Elle regroupe les activités I, J et K.
Conception physique	Physical design	Cette phase structure l'architecture du système et associe les résultats de la conception objet aux composants logiciels. Elle regroupe les activités α, β et γ.

Activités

TERME / EXPRESSION	TRADUCTION ANGLAISE	DEFINITION
Ajout de composants de conception	Design packages identification	Cette activité (G) consiste à intégrer les composants de conception (composants techniques, composants réutilisés) à ceux identifiés lors des phases E et F.
Allocation des classes	Classes allocation	Cette activité (γ) réalise l'affectation des classes aux composants logiciels et aux processus identifiés lors de l'activité β.
Analyse objet	Object analysis	Cette activité (D) produit une première identification des objets et de leur organisation en packages.
Application de modèles de conception	Design pattern application	Cette activité (H) consiste à identifier les modèles de conception (design patterns) standard ou métier qui peuvent s'appliquer aux classes et paquetages identifiés lors des phases E et F.
Conception des classes	Class design	Cette activité (I) permet de compléter la description des classes, en précisant leurs attributs et leurs méthodes, de finaliser les associations (rôles, cardinalités, etc.), et d'identifier les classes de conception. Cette activité doit reposer sur une justification des choix, choix qui doivent être validés avant de formaliser la solution d'implémentation choisie.
Conception des classes BD	DB class design	Cette activité (K), similaire à l'activité "Conception des classes", couvre les éléments de conception spécifiques aux bases de données.
Conception des classes IHM	MMI class design	Cette activité (J), similaire à l'activité "Conception des classes", couvre les éléments de conception spécifiques aux interfaces homme-machine.
Définition des acteurs	Actors definition	Cette activité (A) définit tous les acteurs du système à concevoir. Elle permet ainsi de délimiter les contours du système, et d'en fixer les limites contractuelles.
Définition du contexte	Context definition	Cette activité (B) décrit le contexte dans lequel évolue le système. Ce contexte implique les entités passives et les flux statiques et dynamiques de données.
Description de l'architecture physique	Physical architecture description	Cette activité (α) décrit et organise les ordinateurs, les périphériques, les couches de communication, etc., qui composent le système. Cette activité peut démarrer en parallèle avec l'activité A.
Description des composants logiciels	Software components description	Cette activité (F) décrit la structure interne statique des packages, en termes de classes et d'associations entre ces classes. Une première identification des attributs et méthodes est également possible.
Description du système	System description	Cette activité (C) décrit les interactions fonctionnelles des acteurs avec le système, et permet d'établir une analyse du domaine couvert par le système. Le résultat principal est le regroupement fonctionnel des scénarios d'utilisation du système en cas d'utilisation.
Identification des composants logiciels	Software components identification	Cette activité (E) décrit les interfaces externes du système et les collaborations entre les paquetages identifiés lors de l'analyse objet.
Identification des processus et composants	Processes identification	Cette activité (β) complète la description de l'architecture physique en précisant la répartition des processus. Cette activité peut démarrer en parallèle avec l'activité D.

Diagrammes UML

TERME / EXPRESSION	TRADUCTION ANGLAISE	DEFINITION
Diagramme d'activité	Activity diagram	Un diagramme d'activités est une variante des diagrammes d'états-transitions, organisé par rapport aux actions et principalement destiné à représenter le comportement interne d'une méthode ou d'un cas d'utilisation [1].
Diagramme de cas d'utilisation	Use Case diagram	Un diagramme de cas d'utilisation représente les fonctions du système du point de vue de l'utilisateur. Il montre les interactions entre les acteurs et les cas d'utilisation.
Diagramme de classes	Class diagram	Un diagramme de classes exprime la structure statique d'un système, en termes de classes et de relations entre ces classes [1].
Diagramme de collaboration	Collaboration diagram	Un diagramme de collaboration montre des interactions entre objets, en insistant plus particulièrement sur la structure spatiale statique qui permet la mise en collaboration d'un groupe d'objets. Un tel diagramme exprime à la fois le contexte d'un groupe d'objets et l'interaction entre ces objets [1].
Diagramme de composants	Component diagram	Un diagramme de composants décrit les éléments physiques et leurs relations dans l'environnement de réalisation. Un tel diagramme montre les choix de réalisation [1].
Diagramme de déploiement	Deployment diagram	Un diagramme de déploiement montre la disposition physique des différents matériels qui entrent dans la composition du système, et la répartition des programmes exécutables sur ces matériels [1].
Diagramme de séquence	Sequence diagram	Un diagramme de séquence montre des interactions entre objets selon un point de vue temporel. Le contexte des objets n'est pas représenté de manière explicite [1].
Diagramme d'états-transitions	Statechart diagram	Un diagramme d'états-transitions montre un automate à états finis, du point de vue de ses états et des transitions entre ses états [1].
Diagramme d'objets	Object diagram	Un diagramme d'objets montre la structure statique d'objets et de liens entre ces objets. Ce type de diagramme est proche du diagramme de classes.

Rôles

TERME / EXPRESSION	TRADUCTION ANGLAISE	DEFINITION
Analyste	Analyst	Ce rôle couvre les activités d'analyse de la méthode UML/MERCURE. L'analyste est spécialisé dans le recueil du besoin client et dans sa formalisation. Ce rôle répond à la question "Quoi".
Architecte système	System architect	Ce rôle couvre la conception de la partie physique du système. L'architecte décompose le système en composants physiques, et les relie aux composants logiciels produits par le concepteur. Ce rôle répond à la question "Où".
Concepteur	Designer	Ce rôle couvre les activités de conception de la méthode UML/MERCURE. Le concepteur modélise les composants logiciels (packages) du système, décrit les classes associées, décide de l'utilisation de patterns, et choisit les paquetages de conception à utiliser. Ce rôle répond à la question "Comment".

Termes UML

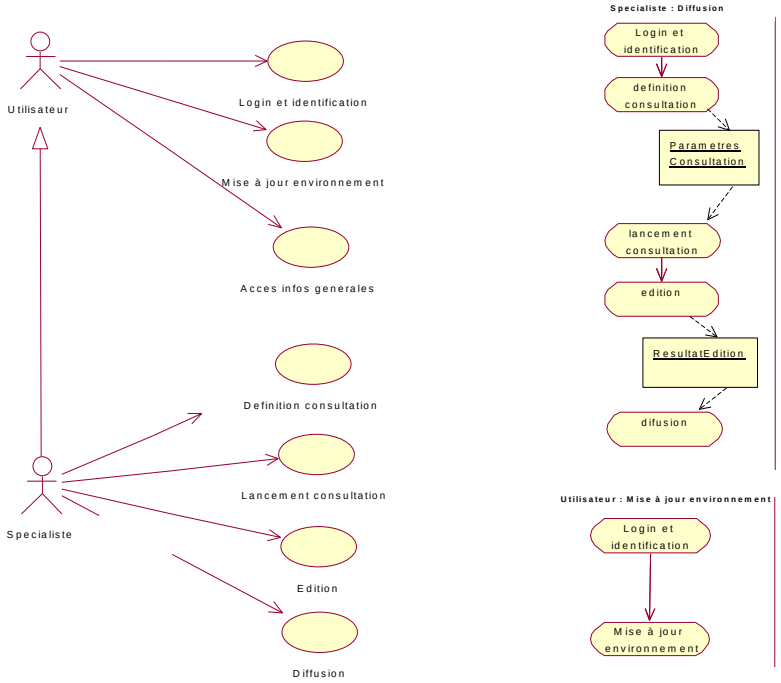
TERME / EXPRESSION	TRADUCTION ANGLAISE	DEFINITION
Analyse du domaine	Domain analysis	Partie de l'analyse qui se concentre sur l'environnement de l'application [1].
Cas d'utilisation	Use Case	Technique d'élaboration des besoins fonctionnels, selon le point de vue d'une catégorie d'utilisateurs [1].
Package (paquetage)	Package	Un paquetage est un regroupement d'éléments de modélisation (packages, classes, associations, etc.).

4 PREAMBULE

4.1 UTILISATION DES DIAGRAMMES D'ACTIVITES

4.1.1 Les DA pour modéliser un enchaînement de Use Case

Les Diagrammes de Use Case (UC), proposés par la notation UML **ne permettent pas** de modéliser la collaboration entre un ensemble d'acteurs et de UC. En effet ils ne montrent ni l'enchaînement, ni les différentes conditions d'exécution. Les diagrammes de séquence permettent d'exprimer l'ordre des opérations qui composent un UC mais ne sont pas prévus et ne conviennent pas pour exprimer l'enchaînement des UC. Il en est de même des diagrammes de collaboration. Le besoin de montrer ce comportement étant très fort sur nos projets, nous avons choisi (après différents essais) de compléter les diagrammes de UC qui en ont besoin, par un DA. La sémantique utilisée est la suivante:
une activité représente un UC, elle porte le même nom que ce dernier,
les échanges de données sont montrés à travers des objets, qui sont en fait des instances de classes (celles trouvées en analysant les UC),
le nom de l'enchaînement décrit, est donné par le nom du diagramme et/ou le nom de la "Swimlane" associée.

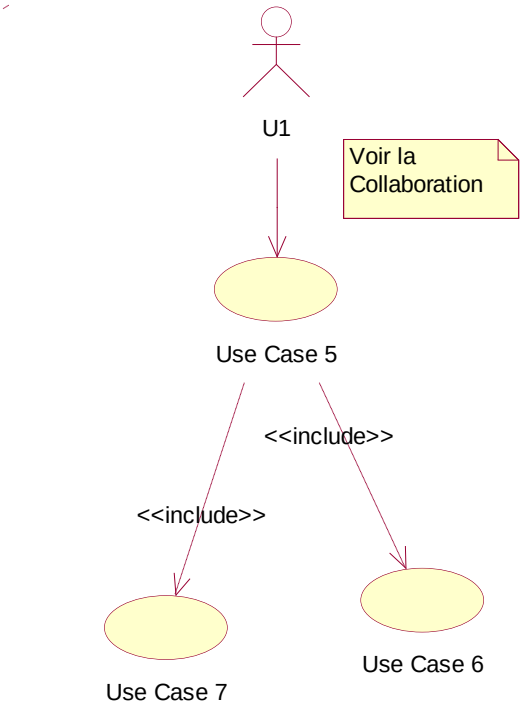


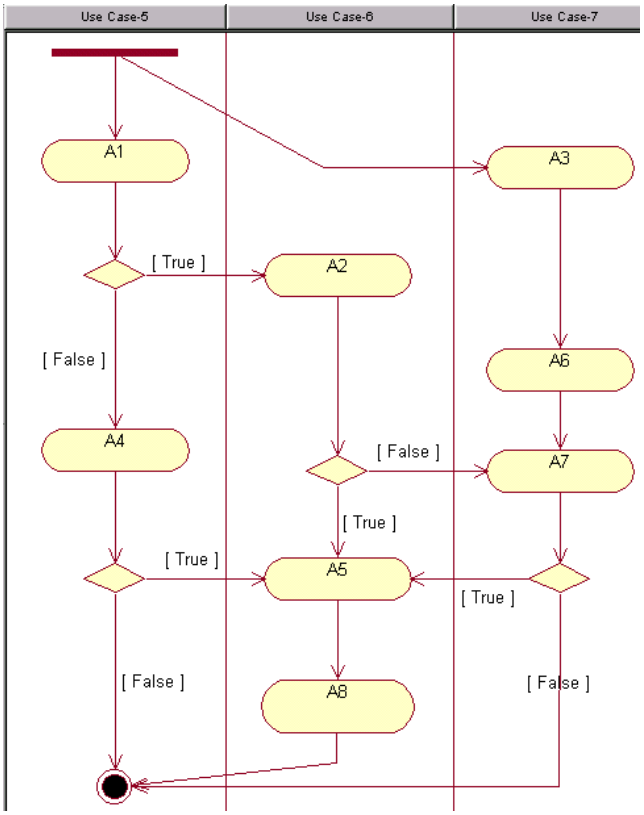
Dans l'exemple ci-dessus, nous pouvons voir :
un diagramme de Use Case (à gauche) représentant deux acteurs et les Use Cases qu'ils sont susceptibles de déclencher. Nous ne savons pas si ces Use Case sont liées entre eux, ni s'il y a un ordre dans leur déclenchement,
le premier diagramme d'activité (en haut à droite) permet de montrer l'enchaînement des Use Cases nécessaires au Spécialiste pour assurer la diffusion ainsi que les données échangées (ParametresConsultation et ResultatEdition),
le deuxième diagramme d'activité (en bas à droite) permet de montrer l'enchaînement des Use Case nécessaires à l'Utilisateur pour mettre à jour l'environnement.

Il s'agit ici d'un cas très simple avec uniquement une réalisation séquentielle. Dans les cas plus complexes, nous utilisons les synchronisations et toutes les possibilités des diagrammes d'activité proposées par la notation UML pour exprimer la sémantique précise de l'enchaînement modélisé.
Remarque: *Un Diagramme de UC peut éventuellement être accompagné par plusieurs DA, chacun montrant une facette différente du fonctionnement.*
L'**un des principaux avantages issus de ce principe est le suivant :** là où avant il fallait écrire beaucoup de texte dans la description des UC (pré-conditions, post-conditions, entrées …), et inclure des commentaires ou notes dans les Diagrammes de UC, **maintenant nous modélisons**. D'autre part, voir globalement le comportement du système à travers ses cas d'utilisation **n'est plus un problème**.

4.1.2 Les DA pour modéliser un enchaînement de Use Case à travers leurs activités internes

Nous avons vu ci-dessus, comment utiliser un diagramme d'activité pour montrer un enchaînement global de UC. Cependant ce principe n'est pas toujours suffisant lorsque l'analyste souhaite montrer la collaboration d'une manière plus détaillée. Dans ce cas nous allons utiliser le concept de "Swimlane" (SL) des DA pour détailler le comportement.
La sémantique utilisée est la suivante:
une SL représente un UC, elle porte le même nom que ce dernier,
les activités représentées à l'intérieur de la SL, sont des activités internes aux UC
les UC collaborent à travers les activités
A partir de là, comme pour l'exemple précédent, nous pouvons utiliser les barres de synchronisation, les conditions avec plusieurs branches, les boucles … proposées par la notation UML pour modéliser la collaboration.





Dans l'exemple ci-dessus, nous pouvons voir:
un diagramme de Use Case (à gauche) représentant un acteur U1 qui déclenche le "Use Case 5"; ce dernier déclenche obligatoirement les "Use Case 6 et 7".
Remarque: *Cette notion d'obligation est indiqué par le stéréotype <<include>>, mais sans notion d'ordre.*

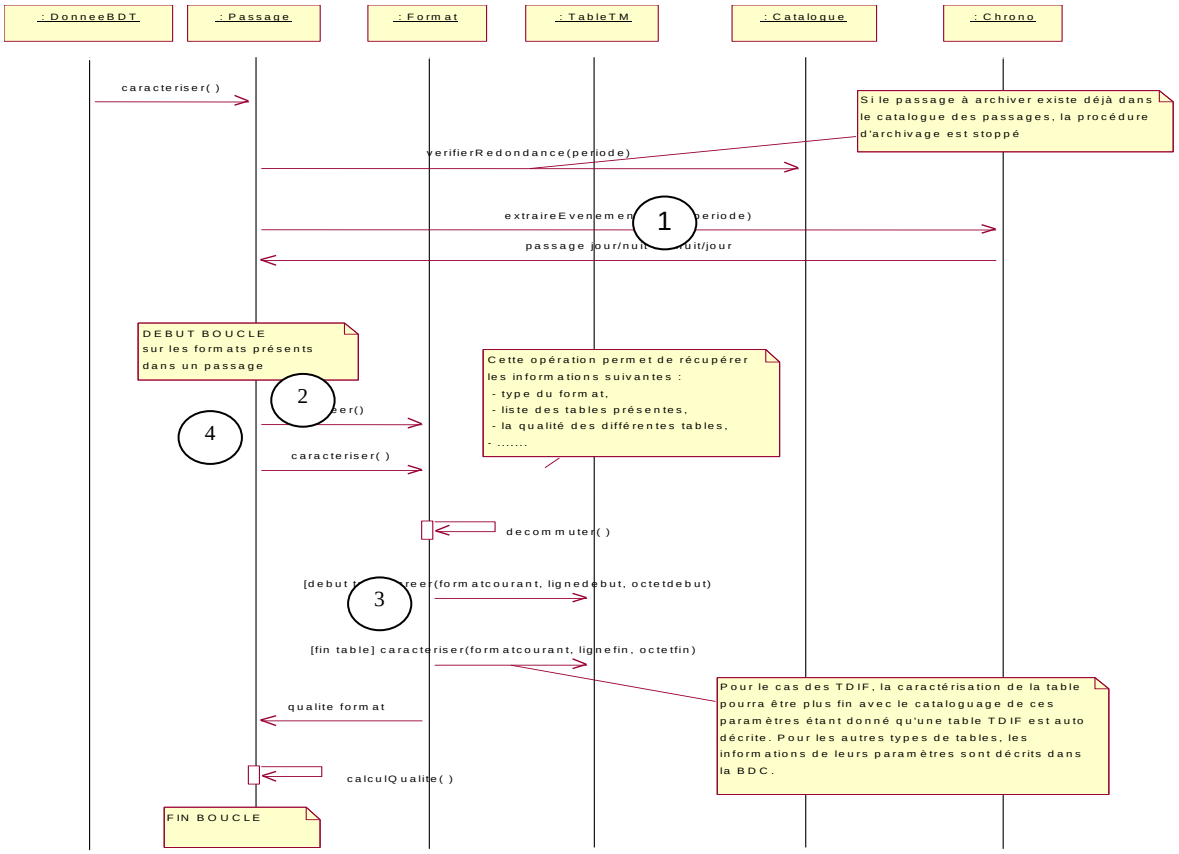
un diagramme d'activité (à droite) représentant trois "Swimlane", une par Use Case, avec pour chaque "Swimlane" les activités, les tests et les collaborations réalisées à l'intérieur de chaque Use Case.
Le diagramme d'activité nous permet de comprendre comment réagissent les trois UC après l'activation du "Use Case 5" par l'acteur U1. En effet sans le diagramme d'activité, la seule chose que nous pouvons déduire est que les Use Case 6 et 7 seront déclenchés lorsque l'acteur U1 active le Use Case 5.

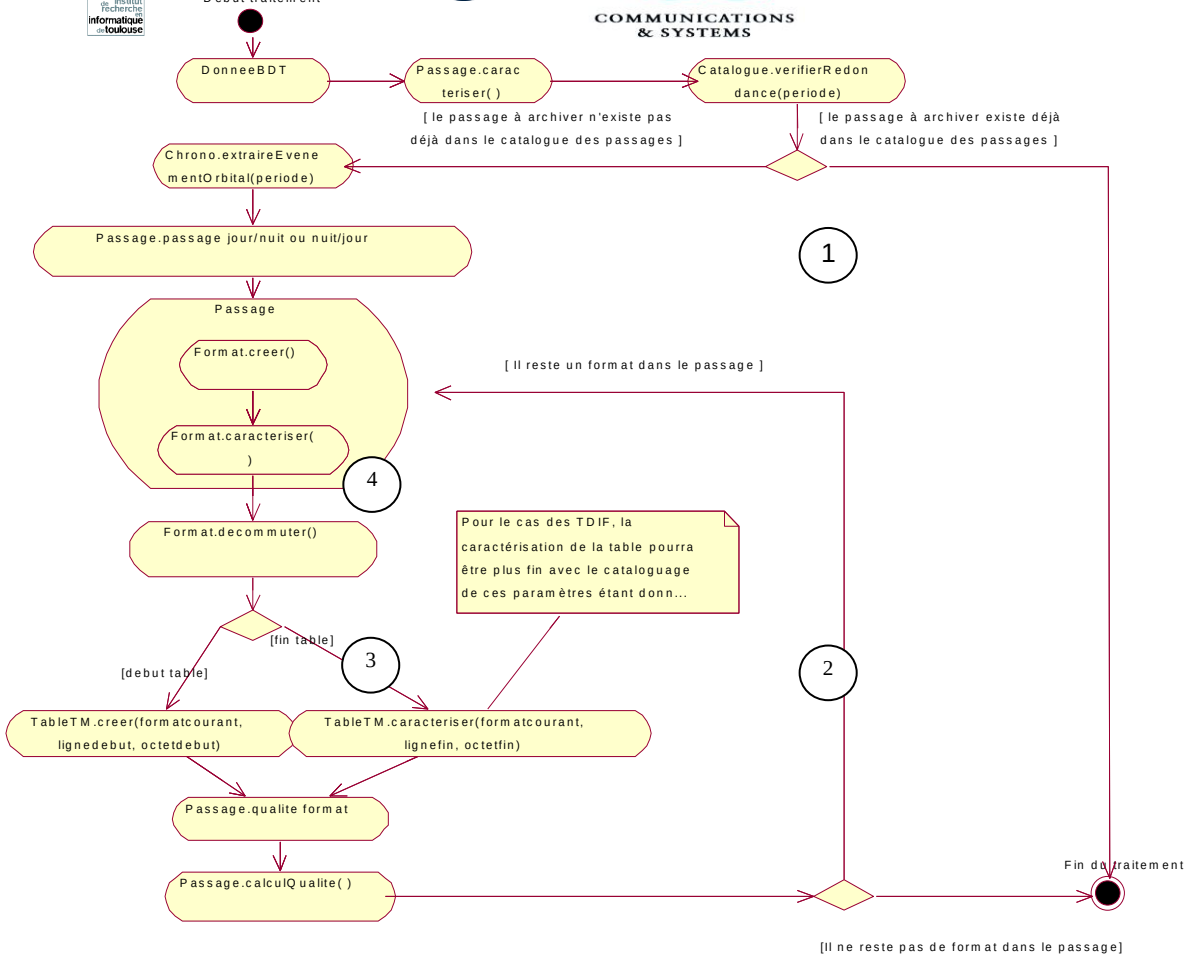
Le principal avantage issu de ce principe est évident, c'est une amélioration de la lisibilité du modèle d'analyse.
Remarque: *la note UML contenant le texte "Voir la collaboration" est en fait un "hyper lien" qui permet de visualiser le diagramme d'activité sur un simple "click". Ce principe de lier les diagrammes par "hyper lien" est très pratique, mais pour l'instant reste une facilité fournie par certains outils. A quand sa définition précise dans la norme ?*

4.1.3 Les DA pour remplacer avantageusement certains diagrammes de séquence

Les diagrammes d'activité peuvent remplacer les diagrammes de séquence dans les cas où ces derniers:
constituent un enchaînement d'étapes clefs,
présentent un grand nombre de branchements.
Dans ces cas, un diagramme d'activité permet une lecture plus simple et plus efficace que les traditionnels diagrammes de séquence, car ces derniers présentent des lacunes et des ambiguïtés sur certains aspects. En effet lorsque nous sommes confrontés à un test ou à une boucle il n'est pas aisé de l'exprimer.

La sémantique utilisée est la suivante:





une activité représente un objet, avec la convention de notation suivante : "NomClasse.Opération"(ex : Format.creer().)

Dans l'exemple présenté, nous pouvons voir un diagramme de séquence (à droite) et le même scénario représenté par un diagramme d'activité (en bas à gauche). Le diagramme d'activité lève les ambiguïtés des points **1**, **2**, **3** :
en précisant le point de sortie du test **1**,
en apportant une représentation plus juste de la boucle **2**,
en présentant le branchement en **3** sous la forme d'une décision.

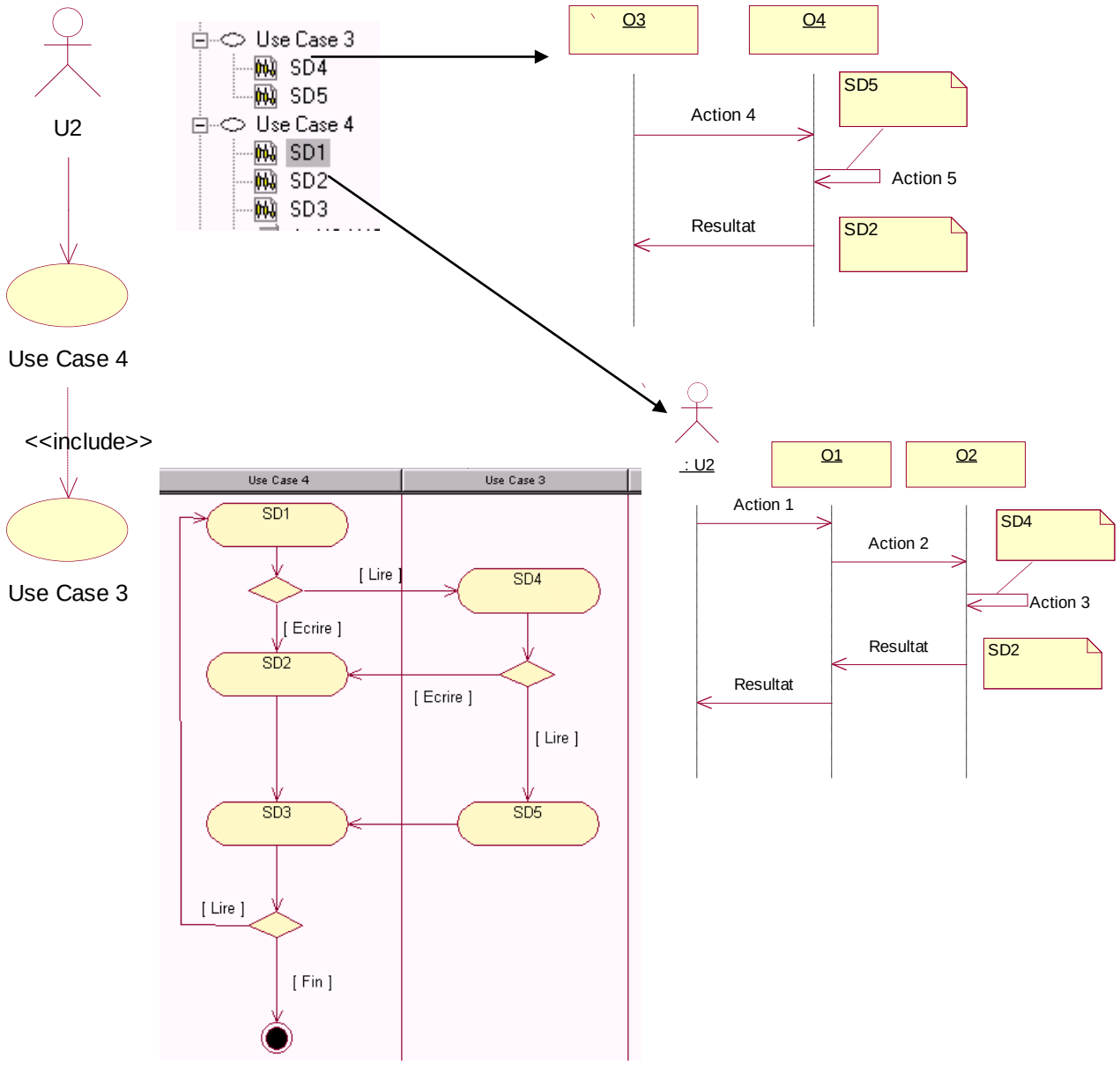
IMPORTANT: L'activité "passage (4)" n'est pas obligatoire, elle a été rajoutée pour conserver une information exprimée plus clairement dans un diagramme de séquence : le fait que "passage" est l'émetteur des deux messages suivants (créer et caractériser).

Les diagrammes d'activité permettent une représentation non ambiguë des enchaînements de messages dans les scénarii.

4.1.4 Les DA pour montrer l'enchaînement de n scénarios

Souvent il est nécessaire de montrer un enchaînement de scénarios afin de regrouper sur un même diagramme le comportement global d'un ensemble d'objets. Dans ce cas le diagramme d'activité est très pratique, en effet, là encore nous pouvons utiliser les barres de synchronisation, les conditions avec plusieurs branches, les boucles … proposées par la notation UML pour modéliser la collaboration, cette fois-ci entre des scénarios.
La sémantique utilisée est la suivante:

une activité représente un scénario, elle porte le même nom que ce dernier.



Dans l'exemple ci-dessus, nous pouvons voir:
un diagramme de Use Case (à gauche) représentant un acteur U2 qui déclenche le "Use Case 4"; ce dernier contenant trois scénarios SD1,SD2 et SD3. Le "Use Case 4" déclenche obligatoirement le "Use Case 3" (stéréotype <<include>>); ce dernier contenant deux scénarios SD4 et SD5.
un premier diagramme de séquence SD1 (en bas à droite) montrant la collaboration entre les objets O1,O2 pour réaliser l'action 1 demandée par l'acteur U2 à travers le "Use Case 4". A noter que l'action 3 est décrite soit par SD2 (pour écrire) soit par SD4 (pour lire) - voir les tests dans le diagramme d'activité -; les deux notes UML permettent d'aiguiller (grâce à un "hyper lien") vers le bon diagramme.
un deuxième diagramme de séquence SD4 (en haut à droite) montrant la collaboration entre les objets O3,O4 pour réaliser l'action 3 demandée par le "Use Case 4" à travers le scénario SD1. A noter que l'action 5 est décrite soit par SD2 (pour écrire) soit par SD5 (toujours pour lire).
un diagramme d'activité (au milieu) représentant deux "Swimlane", une par "Use Case", avec pour chaque "Swimlane" les activités (représentant les scénarios), les tests et les collaborations réalisées à l'intérieur de chaque "Use Case".

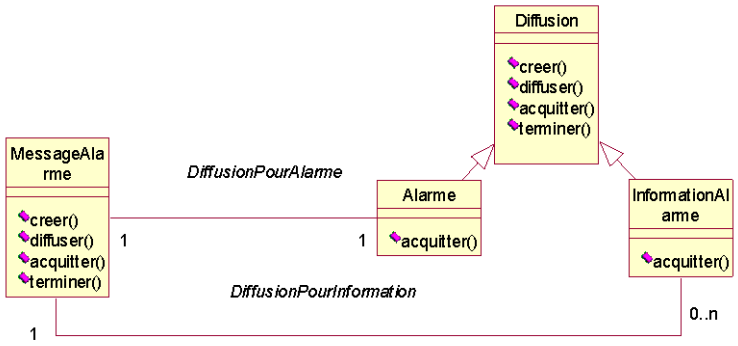
Remarques:
Les scénarios SD2,SD3 et SD5 ne sont pas représentés sur cet exemple.
Nous pouvons bien évidemment ne traiter qu'un seul Use Case à la fois, dans ce cas les "SwimLanes" ne sont plus nécessaires (le nom du diagramme suffisant à indiquer le Use Case décrit).

Le diagramme d'activité nous permet de comprendre comment s'enchaînent les n scénarios d'un UC, ou les n scénarios de n UCs.

Le principal avantage issu de ce principe est que nous pouvons avoir une vue d'ensemble du fonctionnement général de manière synthétique et claire.

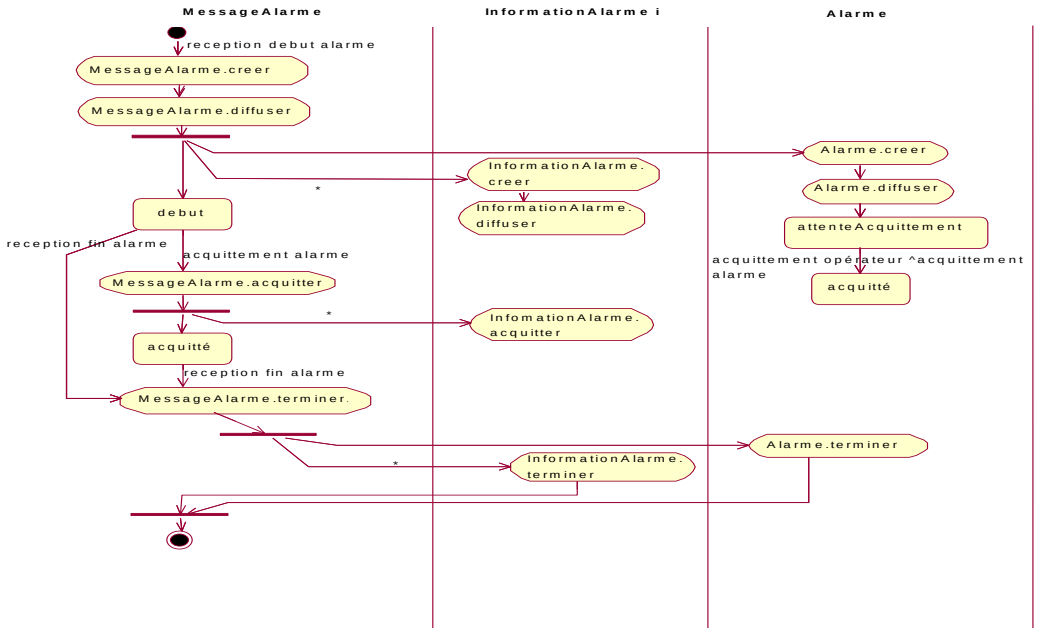
4.1.5 Les DA pour montrer l'enchaînement d'un ensemble d'opérations d'un groupe de classes

Il est fréquent de vouloir exprimer un traitement complexe faisant intervenir plusieurs méthodes de différentes classes :
au sein d'un même paquetage pour rendre un service global par exemple,
entre des classes d'interface de paquetages différents.
Le diagramme d'activité permet alors d'illustrer la manière dont les classes collaborent plus précisément que le diagramme de collaboration.
Dans ce type d'utilisation (pour représenter les interactions entre méthodes de classes différentes), les diagrammes d'activité complètent les diagrammes d'état de chacune des classes utilisées en montrant leur interaction. Les deux types de diagrammes permettent alors de spécifier complètement le traitement. La sémantique utilisée est la suivante :
une "Swimlane" représente une classe,
une activité correspond à une invocation de méthode.
L'exemple suivant présente un cas concret (logiciel existant) :



L'application traite des messages d'alarmes reçus d'un équipement et est chargée d'assurer la diffusion de ces messages vers des opérateurs de maintenance.
Chaque message d'alarme donne lieu à la diffusion d'une seule alarme (destinée à l'opérateur en charge de la résolution du problème) et à n diffusions d'information d'alarme (destinées aux autres opérateurs dont le réseau peut être impacté par le problème).

Lorsque l'opérateur de maintenance en charge d'une alarme déclenche les opérations de maintenance nécessaires à la résolution du problème (envoi d'une équipe sur le terrain), il acquitte l'alarme pour informer les autres opérateurs de la prise compte du problème.



Dans l'exemple ci-dessus, nous voyons la contribution des trois classes "MessageAlarme", "InformationAlarme" et "Alarme" à la réalisation du traitement suivant:
première phase, traitement de l'occurrence d'un problème : arrivée d'un message d'alarme et diffusion aux opérateurs,
deuxième phase, prise en compte du problème : acquittement de l'alarme par l'opérateur responsable,
troisième phase, disparition du problème : arrivée d'un message de fin d'alarme et disparition des diffusions aux opérateurs.
De plus, cette description montre :
que c'est la classe "MessageAlarme" qui est à l'origine de la mise à jour des diffusions d'"Alarme" et d'"InformationAlarme",
la dissimétrie dans le traitement du début/fin d'alarme et de l'aquittement ; en effet, lors de l'acquittement, l'événement parvient tout d'abord à la diffusion "Alarme" qui provoque la propagation de cet acquittement.

Le principal avantage de ce type d'utilisation est de :
montrer clairement l'enchainement des opérations des differentes classes,
compléter les diagrammes d'état des classes en montrant leurs interactions.

4.1.6 CONCLUSION

Utiliser les diagrammes d'activité avec une sémantique particulière tout en respectant la notation UML, nous permet d'améliorer de manière importante la lisibilité de nos analyses UML, notamment sur les projets mettant en œuvre des processus séquentiels (beaucoup d'algorithmique) qu'il est important de montrer en phase d'analyse.
De plus, nous obtenons une représentation plus précise et plus formelle qu'un simple texte ou des notes associées aux éléments.

Cette formalisation peut par exemple permettre l'utilisation d'un **CHECKER** pour vérifier la conformité du modèle (des aspects nouveaux peuvent alors être vérifiés).

Une voie en cours d'exploration :
Nous voulons exprimer des contraintes (du type pré-condition, post-condition, invariant) non pas sur une méthode d'une classe mais sur un traitement global faisant intervenir plusieurs classes.
Il est possible d'exprimer ces contraintes sur un diagramme de classe, mais dans ce cas, il est difficile de rattacher la contrainte à tous les éléments concernés.

Habituellement ces contraintes sont exprimées :
au moyen de notes, le diagramme étant surchargé de liens réduisant sa clarté et sa lisibilité,
au moyen de descriptions textuelles, ces description étant généralement intégrées à un document textuel annexe utilisé lors de la génération de documentation,
au moyen de contraintes UML de manière formelle en exprimant ces contraintes au moyen d'OCL.
Dans tous les cas, il est difficile d'exploiter cette information, car en général elle n'est pas proche des éléments concernés.

Le diagramme d'activité **devrait permettre** d'exprimer ceci de manière plus claire, et de montrer plus facilement les éléments impliqués dans la contrainte.

4.2 UTILISATION DES COMPOSANTS DANS LE PROCESSUS

4.2.1 Objet du chapitre

L'objectif de ce chapitre est de montrer comment peuvent être formalisées des solutions réutilisables ou composants.

4.2.2 Présentation et classification des solutions de conception

La notion de patterns a émergé dans le monde de la conception objet depuis une vingtaine d'années. Cette émergence s'est concrétisée par la proposition de patterns [8] [9] destinés à être repris par les concepteurs du monde industriel informatique. Ces patterns, regroupés en catalogues, sont toujours applicables car leurs descriptions (objet et implémentation) sont accompagnées de justification, de cas d'utilisation, et d'une argumentation critique de leurs intérêts. Les concepteurs peuvent ainsi tirer profit de ces patterns en terme de fiabilité et de productivité.

Compte tenu de la variété des solutions et des utilisateurs potentiels lors des activités du cycle de développement, il semble utile d'essayer de classer les différentes natures de solutions : tout d'abord, selon la distinction pattern ou composant, puis selon l'applicabilité (standard ou métier), et enfin selon la position dans le processus de développement.

4.2.3 Définitions

Une solution de conception réutilisable est une forme générale de conception répondant à un problème général.

Il existe deux types de solutions de conception réutilisables (représentation mentale et abstraite de l'objet à produire) : les « Patterns » et les « Composants ».

4.2.3.1 Pattern et Composant

Le **Pattern** est une solution à un problème donné dans un contexte précis, servant de modèle d'imitation pour une conception. Le mot « pattern » signifie « patron » comme ceux utilisés en couture. Ces derniers, répondent à un problème (ex : tailler un pantalon) en proposant une solution (ex : un modèle pour la réalisation d'un pantalon). Cependant il faut garder à l'esprit que l'utilisation du modèle doit être adaptée en fonction du contexte dans lequel il est utilisé (ex : taille de la personne...) ; c'est pourquoi, comme nous le verrons par la suite, la mise en œuvre d'un « Pattern » peut être différente en fonction du contexte.

Le **Composant**, contrairement à un Pattern, est déjà instancié. Il fournit une interface (avec aucune, une ou n implémentations).
Ces solutions peuvent être standard ou métier et porter sur les phases d'analyse, d'architecture ou de conception.

4.2.3.2 Solutions standards et solutions métier

Une solution standard peut s'appliquer sur tout type d'application, ses constituants sont génériques (ex : le **M**odel **V**iew **C**ontroler) ;

Une solution métier s'applique spécifiquement à un métier donné, ses constituants sont du domaine du métier (ex : Application Pilotée par les Données).

Remarque : *La classification entre standard et métier est délicate et probablement arbitraire. Elle est utile pour que le fournisseur de Patterns et/ou de Composants donne son appréciation sur le niveau de généralité de la solution proposée (ex : les solutions type IHM sont considérées standard plutôt que métier IHM).*

4.2.3.3 Solutions de niveau architecture et de niveau conception

Une solution de niveau architecture implique des modules de haut niveau comme les paquetage UML pouvant inclure eux-mêmes des éléments dits terminaux : classes, objets ….
Une solution de niveau conception implique uniquement des modules dits terminaux (classes, objets ...).

4.2.3.4 Intérêt de la notion de « Solution de conception »

Les « Solutions de conception » permettent d'une part la réutilisation de connaissance de manière formalisée et d'autre part, de constituer une base du savoir faire commune à un métier. Un projet, à autant d'intérêt à utiliser l'existant (gain de temps...) qu'à produire de nouvelles solutions (capitalisation sur le savoir faire métier).

4.2.3.4.1 Intérêt Pour le projet qui utilise des solutions de conception

Le principal avantage est que les solutions de conception préconisées ont été testées, utilisées et validées :

par un nombre important de projets pour les « Solutions de conception » standards [8],

par au moins un projet opérationnel pour les « Solutions de conception » métier.

Une « solution de conception » est donc une solution éprouvée qui :

garantit une certaine fiabilité,

facilite, du fait de sa formalisation, sa mise en œuvre et le dialogue entre les différents acteurs du développement (analystes, concepteur, développeurs, mainteniciens),

permet de capitaliser le retour d'expérience,

améliore en conséquence la productivité globale.

4.2.3.4.2 Intérêt pour le projet qui produit des solutions de conception

Les solutions de conception produites sont issues du métier ou d'une implémentation particulière d'une solution standard, elles permettent donc :

de capitaliser pour les projets à venir dans la même filière métier (ce qui permettra de réaliser des gains de productivité dans un proche avenir),

de formaliser la ou les solutions (ce qui a pour conséquence une certaine amélioration de la qualité globale).

4.2.3.4.3 Spécificités concernant la réutilisation des solutions de conception par rapport à la réutilisation de code

Réutiliser des solutions de conception offre un certain nombre d'avantages par rapport à la réutilisation de code :

les solutions de conception sont plus facilement adaptables au contexte spécifique du projet ;

la réutilisation de solutions de conception est plus aisée ;

- réutilisation des concepts sur la base d'un descriptif de haut niveau, plus accessible que le code ;
- meilleure distribution de la criticité des choix impliqués : choix stratégiques en conception, choix tactiques en codage ;

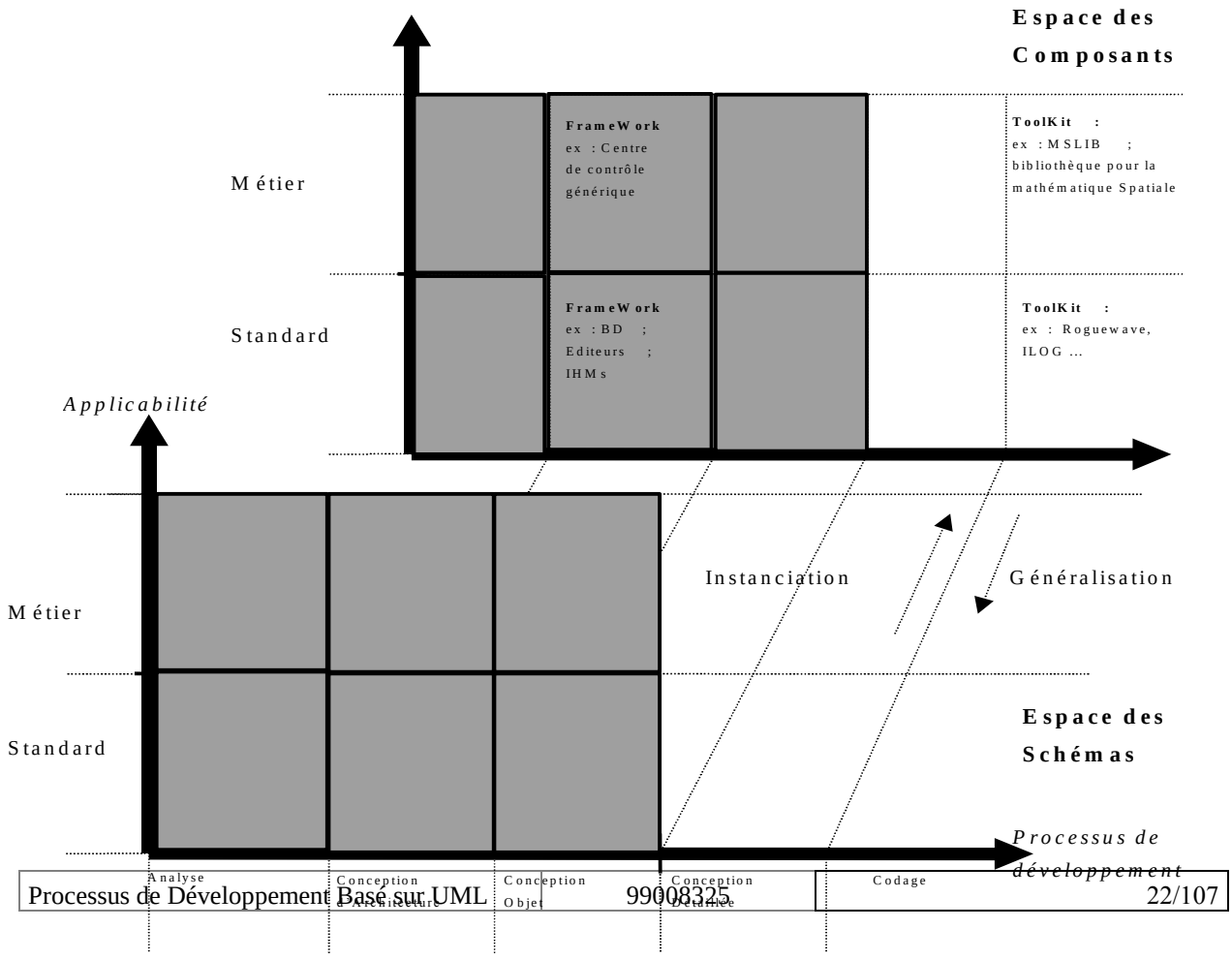
les solutions de conception sont utilisées avec moins de réticence (appropriation plus facile des concepts que du code) ;

les solutions de conception ont une portée plus large car indépendantes de la méthode (d'analyse/conception) et du langage utilisé pour le codage.

4.2.4 Classification des solutions de conception

La figure ci-après montre la couverture de notre étude (zones grisées) suivant les deux axes : Processus de développement (analyse, conception ...) et applicabilité (standard ou métier), à travers deux espaces parallèles : Patterns et Composants, liés par les deux relations suivantes :
une relation d'instanciation : un composant peut être (très rarement dans les faits) une instanciation d'un Pattern,
une relation inverse de généralisation : un Pattern peut être la généralisation d'un ensemble de composants.

Figure 1. Classification des solutions de conception



Critères de choix des solutions réutilisables

Ce paragraphe donne quelques critères de sélection de patterns et donne pour chacun d'eux une description et une justification.

4.2.4.1 Généralité de la solution

Description : Le modèle proposé doit s'appliquer dans un contexte général ; il est potentiellement réutilisable sur tout projet relevant du métier pour les Patterns métier et sur tout type de projet pour les Patterns standards ; il s'intègre dans la classification définie au paragraphe 4.2.4.
Justification : Le non respect de ce critère permet de supposer que le retour sur investissement d'un tel Pattern est proche de zéro et donc il est inutile de chercher à le formaliser.

4.2.4.2 Solution éprouvée

Description : Le modèle proposé doit être issu d'un projet opérationnel sur lequel il s'est appliqué avec des résultats satisfaisants.
Justification : Ce critère est discriminant, en effet il nous paraît impensable de formaliser un Pattern à partir d'une théorie (ou les gains théoriques ne seront pas forcément vérifiés dans la réalité). En effet, la production de notre guide doit se baser sur des projets opérationnels afin de proposer des Patterns validés et éprouvés.

4.2.4.3 Intérêt de la formalisation

Description : La formalisation du modèle doit apporter une aide importante aux futurs concepteurs, en termes de compréhension et d'utilisation.
Justification : Ceci est primordial, car un utilisateur veut : (1) comprendre facilement, (2) utiliser très rapidement, (3) ne pas se poser les éternelles questions (pourquoi ceci ou cela ...) ; il veut que sa marche bien et vite.

4.2.5 Guide pour produire et documenter de nouvelles solutions réutilisables

L'objectif de ce paragraphe est de faciliter les opérations de création d'un nouveau composant en précisant un ensemble de règles qui vont permettre de :

rechercher de nouveaux patterns sur votre projet et en documenter leur utilisation (ajout d'un nouveau pattern dans un catalogue de patterns ou de composants) ;

utiliser des patterns existants sur votre projet et en documenter leur utilisation (ajout d'un exemple d'utilisation dans le catalogue).

La documentation d'un pattern doit suivre le plan suivant :
Un plan standard garantit l'homogénéité, la cohérence et la lisibilité de la documentation.
Nom, Problème posé.
Structure et Description des classes.
Collaboration et Scénario(s).
Avantages et inconvénients.
Exemple(s) de mise en œuvre.

La documentation d'un exemple de mise en œuvre doit suivre le plan suivant :
Spécificités de mise en œuvre.
Structure et description des classes.
Collaborations et scénarios.

Le choix d'utilisation de patterns standards [8] doit être fait en début de la phase de conception.
Ce choix présente deux avantages : d'abord il favorise la réutilisation de l'existant, ensuite il permet de planifier l'effort et la préparation de l'exemple d'utilisation qui sera inclus en fin de projet dans le catalogue. La production de cet exemple permet entre autres de capitaliser l'expérience.

La définition d'un nouveau pattern doit se faire au début de la conception ou au plus tard avant la fin de la phase.
Après il est trop tard, les efforts de retro-conception ou d'extraction de solutions à partir du code sont très difficiles.

La description du pattern doit suivre la notation UML, elle doit présenter un bon niveau d'abstraction et mettre l'accent sur les scénarios.
L'utilisation d'une notation prédéfinie simple permet d'une part de garantir l'homogénéité et la cohérence du document et d'autre part de faciliter la compréhension du lecteur.
L'utilisation des scénarios ayant un bon niveau d'abstraction permet d'une part de compléter la partie textuelle sans s'appesantir sur les détails, et d'autre part d'illustrer le fonctionnement de manière claire et agréable, ce qui là aussi facilite la compréhension du lecteur.

En fin de projet la description du pattern doit être mise à jour afin de prendre en compte les évolutions dues à la conception détaillée et au codage.
La conception peut évoluer durant les phases de codage, ces modifications auront probablement un impact sur la description initiale du pattern ainsi que sur la description de son implémentation sur le projet. Ces modifications en fin de projet sont donc primordiales pour garantir la réalité de la solution mise en œuvre.

La description du pattern devrait être écrite par le concepteur l'ayant mis en œuvre
Le concepteur d'une solution de conception (pattern) est la personne la plus efficace pour documenter l'utilisation qui en est faite. Tout autre intervenant arrivera à un résultat, mais avec des efforts bien plus importants. Cependant un intervenant externe peut apporter des compléments allant dans le sens d'une meilleure généralisation du pattern.

5 APERÇU DU PROCESSUS

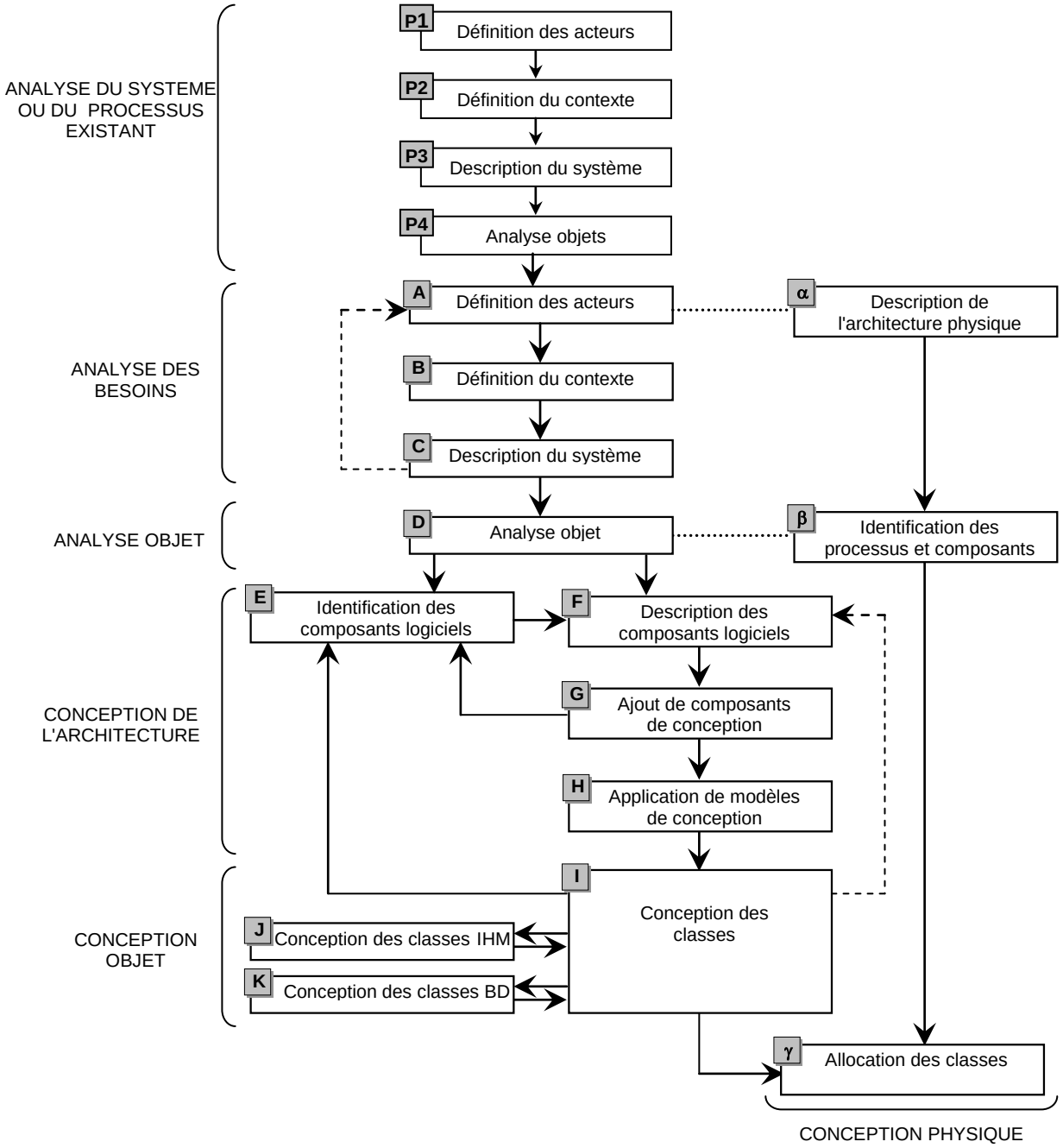
Ce paragraphe propose une vue macroscopique de notre processus, et présente son découpage en phases et activités. Il décrit également de manière succincte les différents rôles impliqués dans sa mise en œuvre.

La Figure 2 montre une vision globalement séquentielle du processus, mais il est important de préciser que ce processus est éminemment **itératif**. Vous pouvez remonter de n'importe quelle activité vers une activité en amont. De même, vous pouvez itérer sur une activité autant de fois que nécessaire.

Ce schéma précise les principales itérations inter-activités par des flèches en tirets.

Les traits pointillés indiquent que les activités peuvent démarrer simultanément.

Figure 2 - Phases et activités du processus



Six rôles, qui peuvent être tenus par une même personne, sont identifiés dans ce processus :

Le **chef de projet** met en place le plan de développement et organise le projet.
L'**analyste**, spécialisé dans le recueil et la formalisation du besoin des utilisateurs défriche le terrain pour répondre à la question "Quoi". Il couvre les phases d'analyse des besoins et de l'analyse objet.
L'**architecte système** décompose le système en composants physiques, identifie et structure les processus physiques et les associe finalement aux composants logiciels (packages) produits par le concepteur. Il couvre les activités de la phase de conception physique.
Le **concepteur** modélise les composants logiciels du système. En particulier, il décrit les classes, pour arriver à une définition précise et complète des attributs et méthodes. Il répond à la question "Comment".
Le **qualiticien** organise les revues qualité et vérifie la conformité des documents produits.
Le **Testeur** se charge des tests nécessaires à la validation du modèle.

Le processus est découpé en 5 phases, chacune d'elles étant découpée en activités (leur description détaillée est fournie dans les paragraphes suivants) :

L'**analyse des besoins**, dont l'objectif principal est de formaliser les besoins et exigences des utilisateurs. L'analyse des besoins comprend la définition des acteurs (utilisateurs, machines, périphériques, systèmes extérieurs, etc.) impliqués dans le système, la définition de son contexte et la description du système lui-même. Il est fortement conseillé de ne passer à la phase suivante qu'après la validation par les utilisateurs des résultats de cette première modélisation. Un des résultats de cette phase peut être le document de spécifications logicielles des besoins.
L'**analyse objet**, qui permet d'identifier et de structurer les principaux objets participants du système. L'analyste pose les bases de l'architecture des composants logiciels, réutilise éventuellement des paquetages standards ou métier, et décrit les interfaces entre les paquetages identifiés.
La **conception de l'architecture**, au cours de laquelle le concepteur structure les composants logiciels (packages) et décide des composants techniques à intégrer et de l'opportunité d'utiliser des modèles de conception (patterns). Le concepteur distingue les paquetages internes des paquetages externes, ces derniers étant en interaction directe avec les acteurs du système.
La **conception objet**, qui correspond à la conception détaillée des paquetages et des classes du système. Le concepteur prend en compte les spécificités du logiciel (orienté-données, orienté-IHM, etc.). Il décrit notamment les attributs et les méthodes des classes, les associations entre classes. Il décide de l'étendue de la conception, c'est-à-dire du niveau de détail à apporter à la conception.
La **conception physique** permet à l'architecte système de structurer le système en composants physiques, de définir les processus exécutés et de mettre en correspondance ces processus avec les composants logiciels. Cette dernière phase peut être réalisée (en fonction du type de projet) en parallèle des autres phases, dans la mesure où il est courant que l'architecture matérielle fasse partie des spécifications initiales du logiciel. Cette phase précède généralement le codage des classes.

STRUCTURE DU DOCUMENT

Les chapitres décrivant les différentes phases et activités de la méthode sont structurés de la façon suivante.

Un Chapitre par Phase

- Un sous chapitre par Activité
  - Objectifs
  - Intervenants
  - Entrées
  - Sorties
  - Description
    - Introduction
    - Nouveaux concepts (si nécessaire, si non standard UML)
    - Comment réaliser les sorties
    - Commentaires ou remarques (si nécessaire)
  - Règles de vérification
  - Exemples
- Fin de l'activité
- Modèles de documents
- Composants

Fin de la phase

6 ANALYSES D'UN SYSTEME OU D'UN PROCESSUS EXISTANT

6.1 NOUVEAUX STEREOTYPES

Dans UML, un élément de conception ne peut être stéréotypé qu'une et une seule fois. Dans un système d'information, chaque activité est caractérisée par trois critères :
Le type de l'activité : une activité peut être manuelle, automatique ou interactive. En effet, lors de l'analyse d'un processus, toutes les activités nécessaires à l'exécution de celle-ci peuvent être soit manuelles, soit interactives ou soit automatiques. Une activité manuelle est réalisée par un individu sans l'aide d'aucun dispositif automatique ou interactif. Une activité interactive est réalisée par un individu aidé d'un dispositif automatique ou interactif. Une activité automatique est réalisée par un dispositif automatique ne nécessitant pas l'implication d'un individu ;
La quantité de données traitée simultanément : unitaire ou par lot (batch). Le traitement des données peut se faire soit unitairement soit par lot ;
Le traitement peut être effectué soit immédiatement soit en différé. Dans le premier cas, l'activité est déclenchée par l'arrivée d'une nouvelle donnée. Dans le second cas, l'activité est soit déclenchée périodiquement, soit déclenchée lorsqu'un certain seuil de données est atteint.

Lorsque l'objectif est d'améliorer le système ou le processus, plusieurs optimisations sont possibles. Lorsqu'une activité est manuelle, l'amélioration consiste à la transformer pour qu'elle soit interactive ou, et c'est encore mieux, pour qu'elle soit automatique. De même, si une activité est interactive, il peut être intéressant de la transformer pour qu'elle devienne automatique. Une autre optimisation réside dans le de regroupement des traitements unitaires pour obtenir un ou plusieurs traitements par lots. La dernière optimisation possible est d'améliorer les délais pour augmenter le nombres de tâches pouvant être effectuées dans un temps donné. Une solution possible pour passer d'une tâche immédiate à une tâche différée est de regrouper les données à traiter par l'activité et d'effectuer le traitement soit lorsqu'une certaine quantité est atteinte soit périodiquement. Dans ce cas, le traitement devient un traitement par lots et il est différé.

UML ne permet pas de stéréotyper un élément déjà stéréotypé. Il est donc nécessaire si on veut prendre en compte les caractéristiques d'une activité de prévoir toutes les combinaisons possibles de ces trois caractéristiques. Toutefois, il est important de remarquer que certaines combinaisons peuvent ne pas être pertinentes
AUI - Automatique-Unitaire-Immediate : lorsque l'activité est réalisée automatiquement, donnée par donnée et immédiatement après la réception de la donnée ;
AUD - Automatique-Unitaire-Différée : lorsque l'activité est réalisée automatiquement, donnée par donnée et à une date donnée ;
ABI - Automatique-Batch-Immediate : lorsque l'activité est réalisée automatiquement, à partir d'un ensemble de données et immédiatement après la réception de l'ensemble des données ;
ABD - Automatique-Batch-Différée : lorsque l'activité est réalisée automatiquement, à partir d'un ensemble de données et qu'un certain volume de données ou une date donnée est atteint ;
IUI – Interactive-Unitaire-Immediate : lorsque l'activité est réalisée de manière interactive, donnée par donnée et immédiatement après la réception de la donnée ;
IUD – Interactive-Unitaire-Différée : lorsque l'activité est réalisée de manière interactive, donnée par donnée et à une date donnée ;
IBI – Interactive-Batch-Immediate : lorsque l'activité est réalisée de manière interactive, à partir d'un ensemble de données et immédiatement après la réception de l'ensemble des données ;
IBD – Interactive-Batch-Différée : lorsque l'activité est réalisée de manière interactive, à partir d'un ensemble de données et qu'un certain volume de données ou une date donnée est atteint ;
MUI – Manuelle-Unitaire-Immediate : lorsque l'activité est réalisée par un acteur actif du système sans l'aide d'un mécanisme interactif, par exemple un ordinateur, donnée par donnée et immédiatement après la réception de la donnée ;
MUD – Manuelle-Unitaire-Différée : lorsque l'activité est réalisée par un acteur actif du système sans l'aide d'un mécanisme interactif, par exemple un ordinateur, donnée par donnée et à une date donnée ;

MBI – Manuelle-Batch-Immediate : lorsque l'activité est réalisée par un acteur actif du système sans l'aide d'un mécanisme interactif, par exemple un ordinateur, à partir d'un ensemble de données et immédiatement après la réception de l'ensemble des données ;
MBD – Manuelle-Batch-Différée : lorsque l'activité est réalisée par un acteur actif du système sans l'aide d'un mécanisme interactif, à partir d'un ensemble de et qu'un certain volume de données ou une date donnée est atteint.

6.2 SYNTHESE DE L'ANALYSE

La figure 1 montre le la séquence des différentes étapes constituant l'analyse de l' existant ainsi que les passerelles entre cette analyse et l'analyse du nouveau système ou processus.

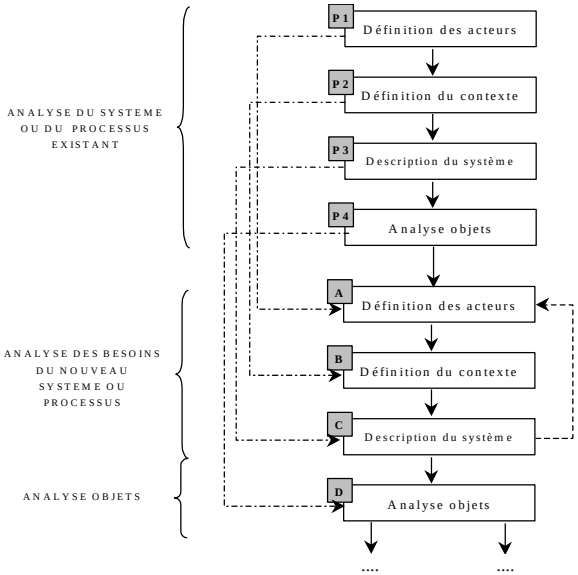


Figure 3: ***Phases et activités du processus UML/MERCURE incluant l'analyse de l' existant***

Les résultats issus des étapes P1, P2, P3 et P4 sont les entrées respectives des étapes A, B, C et D. Il est aussi essentiel de tenir compte du fait que les étapes P1, P2, P3 et P4 sont des vues macroscopiques, seuls les noms des différentes étapes sont similaires avec les étapes A,B, C et D ; en d'autres termes, il existe de nombreuses différences entre les étapes P1, P2, P3, P4 et respectivement les étapes A, B, C, D.

Il est important de remarquer que l'analyse du nouveau système ou du nouveau processus tient compte aussi bien des parties informatisées que des parties non-informatisées. Pour analyser et concevoir l'application ou les applications informatisées nécessaires à la réalisation des activités automatiques ou interactives, il est nécessaire d'isoler dans cette nouvelle analyse les parties pertinentes pour cette réalisation. Il est aussi important de remarquer que l'analyse de l'existant n'est pas à proprement parler itérative. Seule les étapes à partir de l'analyse de besoins utilisateurs peuvent être envisagées de manière itérative. L'analyse de l'existant se décompose en différentes phases elles-mêmes décomposées en sous-phases :

Le principal objectif de l'analyse de l'existant est de formaliser le système ou le processus existant. Cette phase inclut la définition des acteurs participant au processus ou associés au système, la définition du contexte, la description du processus ou du système lui-même et une analyse objet permettant l'identification et l'extraction d'objets représentatifs composant le système. Il est à noter que l'analyse objets est incluse dans cette première phase car l'analyste ne tient pas compte d'une éventuelle architecture logicielle ou d'un besoin de réutilisabilité. En effet, le seul objectif de cette extraction est d'avoir une première décomposition. Cette phase est principalement une décomposition fonctionnelle d'un système ou d'un processus existant. Il est essentiel de remarquer, que les informations nécessaires à la réalisation de l'analyse sont issues soit de l'audit du système, soit de l'organisation pour un processus ou/et soit de l'audition de personnes utilisant le système ou participant au processus. Dans ce dernier cas, les personnes auditionnées expliquent le système ou le processus en termes de fonctionnalités, d'activités, de tâches et de résultats. En aucun cas, elles ne perçoivent leur système ou leur processus en terme d'objets ou d'interactions.

Les phases permettant l'analyse des besoins pour le nouveau système ou le nouveau processus sont identiques à celles permettant l'analyse des besoins dans la méthode générale. La seule différence à noter est que chaque phase de cette analyse doit prendre en compte les sorties correspondantes issues de l'analyse de l'existant. Il est aussi important de remarquer que l'étude d'un système ou d'un processus peut se limiter uniquement à l'analyse de l'existant. Ce cas se produit lorsque l'objectif de l'étude se borne à connaître le fonctionnement d'un système ou d'un processus sans pour autant vouloir l'améliorer dans l'immédiat. Dans ce cas on obtient un instantané d'une situation.

6.3 ANALYSE DE L'EXISTANT

6.3.1 P1 – Définition des acteurs

6.3.1.1 Objectif

Déterminer tous les acteurs et les entités passifs impliqués dans le processus ou le système existant.

6.3.1.2 Participants

L'expert du domaine et l'équipe d'analyse identifient les acteurs. Ces acteurs sont directement extraits du processus ou du système.

6.3.1.3 Entrées

L'audit du système ou de l'organisation ;
Les comptes-rendus des entretiens des personnes en charge du système ou participant au processus ;
Une étude minutieuse du système ou de l'organisation.

6.3.1.4 Sorties

Le paquetage contenant les acteurs actifs ;
Le paquetage contenant les entités passives ;
Le diagramme général des acteurs ;
Le diagramme général des entités passives ;
Le diagramme présentant la hiérarchisation des acteurs (obligatoire) ;
La description textuelle des acteurs identifiés.

Donner le browser modèle montrant la hiérarchie des paquetages et les diagrammes d'acteurs

6.3.1.5 Règles

La notion d'acteurs et d'entités passives est la même que la notion décrite dans le guide méthodologique général. Les acteurs doivent être hiérarchisés en utilisant la notion d'héritage entre acteurs pour montrer les différentes responsabilités au sein de l'activité ou du processus. La définition d'un acteur racine, pour l'ensemble des acteurs internes au système, est souhaitable. Cette racine pourra servir pour matérialiser une responsabilité partagée par l'ensemble des acteurs à un processus ou à un système.

Chaque acteur interagit individuellement avec les cas d'utilisations, déclenche des interactions dans les diagrammes de séquences ou de collaboration de l'étape suivante. Les entités passives apparaissent uniquement dans les diagrammes définissant le contexte général du système ou du processus (activité P3-2).

6.3.2 P2 – Définition du contexte

6.3.2.1 Remarque

Nous ne ferons aucune différence entre les flots de données et les interactions. En effet, cette dichotomie est très liée à une approche informatique de l'analyse et n'est pas pertinente lors de l'analyse de processus ou de systèmes non informatiques.

6.3.2.2 Objectif

Décrire et restaurer le contexte général du système en utilisant un environnement plus global. Pour ce faire, on décrira les échanges de flots de données et les relations entre :
Les entités passives et les acteurs ;
Les acteurs actifs ;
Les acteurs et le système

6.3.2.3 Participants

L'expert du domaine et l'équipe en charge de l'analyse réalisent la définition du contexte. Ce contexte est extrait directement de l'étude du processus ou du système.

6.3.2.4 Entrées

L'audit du système ou de l'organisation ;
Les comptes-rendus des entretiens des personnes en charge du système ou participant au processus ;
Une étude minutieuse du système ou de l'organisation ;
Les résultats de l'activité P1.

6.3.2.5 Sorties

Un diagramme de collaboration représentant les flots de données

6.3.2.6 Règles

Les flots de données, représentés par un diagramme de collaboration ou de séquences, doivent montrer les interactions et les données échangées entre les acteurs et les entités. Le type de diagramme choisi n'est pas pertinent car il existe une bijection entre ces deux types de diagrammes. Certains outils offrent une option permettant de passer automatiquement de l'une des représentations à l'autre.

Le système ou le processus que l'on désire analyser est représenté par une classe spéciale qui peut être stéréotypée. Cette classe s'appelle « ProcessusAEtudier » ou « SystèmeAEtudier ». Durant le reste de ce document et pour simplifier, nous appellerons le processus ou le système que nous étudions « SystèmeAEtudier ». Lors de l'analyse d'un système, les données transférées sont représentées sur le diagramme par de petites flèches positionnées à l'extrémité du lien reliant l'entité et le système. Lors de l'analyse d'un processus, seul les acteurs apparaissent sur le diagramme. Il est important de noter que les flots de données doivent être ordonnés de manière à déterminer l'ordre chronologique dans lequel ils sont échangés.

6.3.2.6.1 Contraintes méthodologiques

Une classe et seulement une classe peut représenter le système ou le processus à étudier dans toute l'analyse.

6.3.3 P3 – Description du système

Cette phase essentielle est subdivisée en quatre sous-phases :
P3-1: Déterminer les activités constituant le processus ou réalisées par le système ;
P3-2: Concevoir et déterminer les flots de données échangés par les activités extraites dans la sous-phase P3-1 ;
P3-3: Déterminer les cas d'utilisation ;
P3-4: Faire l'analyse du domaine.

Les sous-phases P3-1 et P3-2 peuvent être réalisées en parallèle. Les autres sous-phases doivent être réalisées en séquence.

6.3.3.1 P3-1 – Déterminer les activités constituant le processus ou fournies par le système

6.3.3.1.1 Remarques

Lors de l'analyse d'un processus ou d'un système existant, il est plus simple de déterminer les différentes activités et leur enchaînement que directement les cas d'utilisation. En effet, les méthodes d'investigation pour analyser le système ou le processus sont basées soit sur des audits soit sur des entretiens. Les résultats obtenus sont plutôt une description du « comment » (activités) qu'une description du « quoi » (cas d'utilisation).

6.3.3.1.2 Objectif

Identifier les différentes activités et les documenter.

6.3.3.1.3 Participants

L'expert du domaine et l'équipe chargée de l'analyse extraient les activités. Cette extraction se fait directement à partir de la description du processus ou du système à étudier.

6.3.3.1.4 Entrées

L'audit du système ou de l'organisation ;
Les comptes-rendus des entretiens des personnes en charge du système ou participant au processus ;
Une étude minutieuse du système ou de l'organisation ;
Les résultats de l'activité P2.

6.3.3.1.5 Sorties

Des diagrammes d'activités et leur description
Un diagramme de séquence montrant l'enchaînement des différents événements déclenchant les activités

6.3.3.1.6 Règles

Une activité est une fonction qui génère en sortie des flots de données ou qui modifie le système. Une activité est réalisée par un acteur ou un groupe d'acteurs. Toutefois, lorsqu'une activité est réalisée par un groupe d'acteurs il est essentiel que ces groupes soit constitués soit uniquement d'acteurs internes soit uniquement d'acteurs externes. Lorsque cela est possible, on préférera associer une activité à un et un seul acteur. Attention, les acteurs pouvant faire partie d'une hiérarchie d'acteurs (cf. phase 1), il est possible de remplacer un groupe d'acteurs par un acteur plus général de la hiérarchie. Une activité peut être stéréotypée avec l'un des douze stéréotypes proposés dans la partie 3.1. Elle peut être déclenchée soit par la fin de l'activité précédente, soit par le déclenchement d'un événement ou soit par l'arrivée d'un flot de données. Pour les activités déclenchées par un événement, il sera possible de réaliser un diagramme de séquences montrant l'enchaînement des différents événements. Lors de la description de processus ou de systèmes complexes, il sera possible de réaliser plusieurs diagrammes d'activités.

Les services fournis par le processus ou le système seront représentés par des méthodes de la classe « SystèmeAEtudier » ou « ProcessusAEtudier ». Pour chaque service, on définira un ou plusieurs diagrammes de manière à représenter le comportement de ce service à partir des activités extraites. Si la description d'un service complexe est réalisée à l'aide de plusieurs diagrammes de d'activités, chaque diagramme devra avoir une activité en commun pour pouvoir les relier les uns avec les autres et ainsi reconstituer le comportement global du service. L'utilisation de couloirs d'activités (swimlanes) est recommandée pour mettre en exergue les acteurs ou groupe d'acteurs intervenant dans les activités. Dans la sous-phase P3-4, ces groupes d'acteurs nous aiderons à déterminer les interactions avec les cas d'utilisation.

La description textuelle des activités respecte le schéma suivant :

Résumé	Rapide présentation de l'activité
Contexte de l'activité	Condition d'utilisation par l'élément déclencheur (fréquence d'activation par l'élément déclencheur, déclenchement synchrone et asynchrone, etc.)
Elément déclencheur	Acteur, fin d'une activité ou événement
Pré-condition	Condition nécessaire devant être vérifiée avant la réalisation de l'activité
Flots de données en entrée	Flots de données consommés par l'activité
Stéréotype	Stéréotype associé
Description	Description détaillée de l'activité. Cette description inclut la description des interactions entre les acteurs et l'activité.
Post-condition	Condition devant être vérifiée à la fin de l'activité
Flots de données en sortie	Flots de données produits par l'activité
Exceptions	Erreurs pouvant survenir durant la réalisation de l'activité et que le système ou le processus ne peut résoudre

6.3.3.1.7 Contraintes Méthodologiques

Deux activités qui ont le même nom représentent la même activité ;
Les activités présentées dans le diagramme de séquences doivent être un sous-ensemble des activités décrites dans le ou les diagrammes d'activités ;
Les diagrammes d'activités doivent être liés à la classe représentant le système ou le processus à étudier ;
Rechercher les enchaînements d'activités identiques dans plusieurs diagrammes d'activités pour les factoriser ;
Une activité doit être associée à un et un seul couloir d'activités.

6.3.3.2 P3-2 – Concevoir et identifier les flots de données échangés par les activités

6.3.3.2.1 Objectifs

Déterminer les activités qui génèrent et consomment des flots de données ;
Déterminer les classes modélisant l'ensemble des flots de données échangés ;
Factoriser les classes représentant un même flot de données, généré par des activités différentes, et ayant un état différent ;
Pour les flots de données consommés, préciser les flots de données qui sont des informations, des données ou des documents qui disparaissent pendant l'évolution du système ou durant le processus. Dans ce cas, aucune archive et aucune trace de ceux-ci ne seront conservés.

6.3.3.2.2 Participants

L'équipe en charge de l'analyse et l'expert du domaine établissent ces flots de données

6.3.3.2.3 Entrées

Les résultats de l'activité P3-1.

6.3.3.2.4 Sorties

Un ensemble de classes et leur diagramme états-transitions associé. Ces diagrammes incluent les activités qui modifient les états

6.3.3.2.5 Règles

Lorsqu'un flot de données est consommé ou produit par le système ou le processus, une classe est créée. Pour chaque classe définie, un diagramme états-transitions est réalisé pour modéliser le comportement du flot de données. Ce diagramme états-transitions mélange les états et les activités pour visualiser l'impact des différentes activités sur les différents états.

6.3.3.3 P3-3 - Definition of cas d'utilisation

6.3.3.3.1 Objectifs

Grouper les différentes activités dans différents cas d'utilisation et les documenter.

6.3.3.3.2 Participants

L'ensemble de l'équipe d'analyse établit les cas d'utilisation. La documentation des différents cas d'utilisation est partagée entre les différents analystes

6.3.3.3.3 Entrées

Les résultats des activités P3-1 et P3-2

6.3.3.3.4 Sorties

Diagramme général des cas d'utilisation
Description textuelle des cas d'utilisation

6.3.3.3.5 Règles

Le diagramme général des cas d'utilisation regroupe toutes les activités identifiées précédemment. Tous les acteurs identifiés dans la phase P1 doivent interagir avec au moins un cas d'utilisation. La seule exception à cette règle concerne les acteurs ancêtres d'acteurs plus spécialisés.

Un cas d'utilisation est un ensemble de services que le système doit fournir. Il est activé par un des acteurs ou des groupes d'acteurs définis dans les phases P1 et P2. Ils sont regroupés en cas d'utilisation principaux et secondaires pour permettre :
L'utilisation d'un cas d'utilisation déjà défini par un nouveau cas d'utilisation ;
Le regroupement dans un cas d'utilisation d'activités communes à plusieurs cas d'utilisation ;
L'établissement des relations entre les différents cas d'utilisation ;
Le regroupement des cas d'utilisation dans des paquetages, permettant par exemple un regroupement thématique.

La description textuelle des activités respecte le schéma suivant :

Résumé	Courte présentation du cas d'utilisation
Contexte d'utilisation	Condition d'utilisation par l'élément déclencheur (fréquence d'activation par l'élément déclencheur, déclenchement synchrone et asynchrone, etc.)
Elément déclencheur	Acteur ou cas d'utilisation
Pré-conditions	Condition nécessaire devant être vérifiée avant la réalisation du cas d'utilisation
Données consommées	Les données nécessaires au fonctionnement du cas d'utilisation
Description	Description détaillée des interactions entre les éléments déclencheurs et le système
Post-conditions	Condition devant être vérifiée à la fin du cas d'utilisation
Données produites	Les données produites par le cas d'utilisation
Exceptions	Erreurs pouvant survenir durant la réalisation de l'activité et que le système ou le processus ne peut résoudre

A la fin de cette sous-phase, chaque activité extraite dans la sous-phase P3-1 doit être incluse dans un et seulement un cas d'utilisation. Les différentes activités sont incluses dans les cas d'utilisation en respectant les règles suivantes :
Les activités ayant un comportement similaire doivent être incluses dans un même cas d'utilisation ;
Les activités qui sont régulièrement utilisées par plusieurs autres activités doivent être regroupées au sein de cas d'utilisation secondaires. Ces cas d'utilisation devront être inclus par les cas d'utilisation regroupant les activités utilisant ces activités communes ;
Les activités concourant à un objectif commun sont clairement identifiées et doivent être incluses dans un même cas d'utilisation.

6.3.3.3.6 Contraintes méthodologiques

A la fin de cette sous-phase, chaque activité extraite dans la sous-phase P3-1 doit être incluse dans un et seulement un cas d'utilisation.

6.3.3.4 P3-4 – Analyse du domaine

6.3.3.4.1 Objectif

Déterminer les classes propres au domaine de l'application et qui émergent à la fin de l'analyse du système ou du processus.

6.3.3.4.2 Participants

L'équipe en charge de l'analyse identifie les classes du domaine avec l'aide de l'expert du domaine.

6.3.3.4.3 Entrées

L'audit du système ou de l'organisation ;
Les comptes-rendus des entretiens des personnes en charge du système ou participant au processus ;
Une étude minutieuse du système ou de l'organisation ;
Les résultats de l'activité P3-3.

6.3.3.4.4 Sorties

Les diagrammes de classes modélisant le domaine
Des diagrammes d'objets ou de collaboration (optionnel)

6.3.3.4.5 Règles

Une classe du domaine représente un élément descriptif qui doit être représenté dans la description du système ou du processus. La détermination des classes du domaine nécessite d'adopter un vocabulaire commun pour l'analyse et éventuellement la suite de l'analyse. Ce vocabulaire est adopté en établissant une et seulement une représentation de l'élément descriptif. Les classes du domaine sont décrites par un ou plusieurs diagrammes de classes, dans lesquels il est important de montrer le lien entre les classes. Des diagrammes d'objets ou de collaboration peuvent être réalisés pour fournir un exemple concret du domaine.

6.3.3.4.6 Contraintes méthodologiques

Les activités automatiques doivent être associées à uniquement une et une seule classe du domaine ;
Les activités interactives doivent être associées à la fois à une et une seule classe du domaine et à seulement un acteur ou groupe d'acteurs ;
Les activités manuelles doivent être associées à un et seulement un acteur ou groupe d'acteurs.

6.3.4 P4 – Analyse Objet

6.3.4.1 Remarques

Cette phase est uniquement obligatoire si l'objectif du projet est d'informatiser des parties du processus ou du système. Ces parties sont extraites du processus ou du système analysé. Si l'objectif de l'analyse est uniquement d'obtenir une description formalisée du processus ou du système, cette phase et les suivantes sont optionnelles ;
Cette phase permet de préparer la transition d'une analyse n'utilisant pas forcément des paradigmes objets vers une approche orientée objets. Cette transition prépare le passage de l'analyse de l'existant vers la méthodologie générale pour l'analyse et la conception des applications informatiques liées au nouveau processus ou au nouveau système ;
Lorsque l'objectif de cette analyse est uniquement de proposer des améliorations du système ou du processus existant, cette phase permet de réfléchir aux activités pouvant être rendues interactives ou automatisées par exemple.

6.3.4.2 Objectifs

Déterminer les classes de l'analyse, classes du domaine et classes factorisant les classes du domaine, et les regrouper dans des paquetages.

6.3.4.3 Participants

Tous les membres de l'équipe d'analyse effectuent le découpage en paquetages. Leur description est allouée aux différents membres de l'équipe.

6.3.4.4 Entrées

Classes du domaine ;
Classes présentes dans les diagrammes de séquence de la sous-phase P3-5.

6.3.4.5 Sorties

Un ou plusieurs diagrammes de paquetages ;
Un diagramme de classes pour chaque paquetage ;

Un ou plusieurs diagrammes d'objets ou de collaboration (optionnel)

6.3.4.6 Règles

Pour chaque cas d'utilisation il faut créer un diagramme des objets participants aux scénarios qui regroupe toutes les classes utilisées dans les diagrammes de séquences. Ces diagrammes de classes fournissent une aide à localiser les interactions internes des objets du système ou du processus.
Les classes groupées dans les diagrammes des objets participants aux scénarios et les classes du domaine participent à la création du diagramme de classes final de l'analyse. Dans ce diagramme, il est nécessaire de créer des paquetages regroupant les classes. Ces regroupements sont effectués sur les critères suivants :
Regrouper les classes ayant une forte cohérence entre elles ;
Limiter le couplage entre classes issues de paquetages différents ;
Isoler les classes qui apparaissent dans plusieurs diagrammes d'objets participants aux scénarios dans un seul paquetage ;
Isoler les classes qui seront réutilisées dans un seul paquetage ;
Exprimer les associations entre classes aussi simplement que possible pour identifier les futures dépendances entre paquetages ; Généralement, on ne découpera pas les catégories sur des associations de type :
Composition ;
Agrégation entre classes (les classes pour lesquelles des associations de ce type existant sont indissociables) ;
Association de type 0 – n.
De même, on limitera
Le nombre de classes dans un paquetage (périmètre du paquetage plus restreint donc plus facile à appréhender) ;
Les boucles qui induisent une double dépendance entre paquetages : si un paquetage s'appuie sur un autre et inversement, un ou plusieurs paquetages tiers sont nécessaires.

7 ANALYSE DES BESOINS

7.1 (A)-DEFINITION DES ACTEURS

A Définition des acteurs

7.1.1 Objectifs

Il s'agit de déterminer l'ensemble des acteurs impliqués dans le système à étudier. Notre processus introduit ici trois types d'acteurs : acteur actif, entité passive et autres systèmes - cf § 6.1.5.2 Nouveaux concepts – qui traduisent différents niveaux d'implication dans le système.

7.1.2 Intervenants

L'identification des acteurs est effectuée en commun par l'équipe d'analyse. La description de chaque acteur est ensuite répartie aux différents membres.

7.1.3 Entrées

Sans objet.

7.1.4 Sorties

Diagramme général des acteurs actifs,
Diagramme général des entités passives et autres systèmes,
Description textuelle des acteurs identifiés.

7.1.5 Description

7.1.5.1 Introduction

Cette activité est un point de départ pour l'analyse elle permet un démarrage facile de cette activité d'analyse. Elle sera raffinée tout du long de la phase d'analyse par des itération successives il est en effet rare de pouvoir identifier tous les participants d'un système de but en blanc.

7.1.5.2 Nouveaux concepts

Les notions d'acteur (acteur actif, entité passive et autres systèmes) sont présentées dans les définitions. On en rappelle ici les principales différences :
Un acteur actif est une entité (humain, processus, machine, …) en interaction directe avec le système et qui déclenche une fonctionnalité du système étudié, il est stéréotypé <<acteur>>.
Une entité passive est une entité qui ne déclenche pas réellement une fonctionnalité, mais qui peut produire ou consommer des données utiles à la mise en œuvre d'une fonctionnalité sans pour autant la déclencher (c'est alors un acteur qui la déclenchera au moment qu'il considèrera opportun) ; elle est stéréotypée << entité passive >>.
Les autres systèmes sont des entités externes à la modélisation avec qui le système devra interagir ; ils sont stéréotypés << autre système >>.

7.1.5.3 Comment obtenir les sorties

Le **diagramme général des acteurs actifs** est un diagramme de cas d'utilisation où figurent les acteurs liés au système modélisé, c'est à dire les entités déclenchant une fonctionnalité du système.
Remarque : *Des acteurs n'utilisant pas les fonctionnalités du système peuvent y être présents s'ils servent à factoriser des propriétés ou des fonctionnalités (UC) liées à des acteurs directement impliqués, ces acteurs sont liés aux autres par des liens d'héritage.*

Le diagramme général des entités passives et autres systèmes est un diagramme de Use Case, qui présente les entités participant au fonctionnement du système sans déclencher directement de fonctionnalités du système, on y retrouve les entités passives et les autres systèmes.

IMPORTANT :
Les acteurs peuvent être décomposés au travers d'un arbre d'héritage afin de montrer les spécificités et/ou communautés propres à chacun d'eux.

Les acteurs sont décrits de façon textuelle en respectant les rubriques suivantes :

Type de déclenchement	définit si l'entité est passive ou active : Acteur , Entité Externe ou Autre Système
Type d'entité	définit si l'entité est une machine, un humain, un processus, …
Rôle	définit le rôle de l'entité vis à vis du système.
Responsabilité	Description des interactions entre l'entité décrite et le système (en général, dans le cas d'un acteur une responsabilité correspond a un Use Case).

Remarques :
Les acteurs inter-agissent avec les cas d'utilisation, diagrammes de séquences et collaborations (Activité C).
Les entités passives ne sont mises en œuvre que dans les diagrammes définissant le contexte général du système (activité B).

7.1.6 Règles de vérification

Le diagramme général des acteurs actifs ne doit contenir que des entités déclenchant directement des fonctionnalités du système ou dont héritent des entités déclenchant directement des fonctionnalités du système.
Le diagramme général des entités passives ne doit contenir que des entités ne déclenchant pas directement des fonctionnalités du système ou dont héritent des entités ne déclenchant pas directement des fonctionnalités du système.
Toute entité doit être dûment stéréotypée et documentée.

7.1.7 Exemple

Figure 4 - Exemple de diagramme général des acteurs

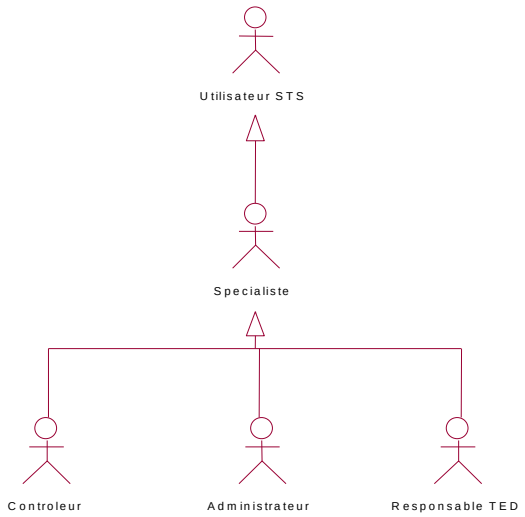
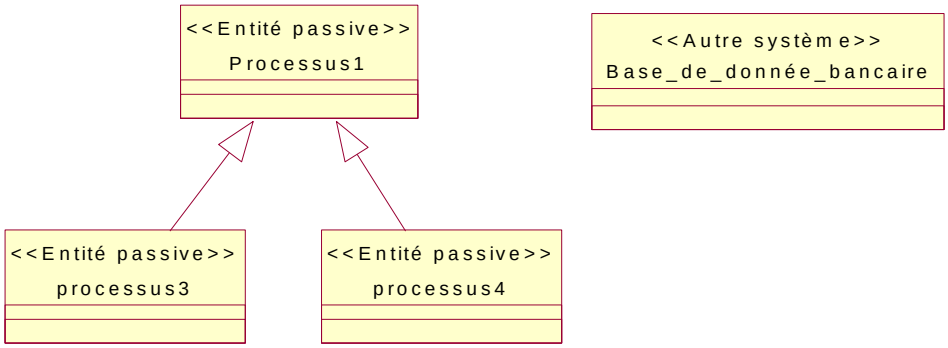


Figure 5 Exemple de diagramme général des entités passives et autres systèmes



B Définition du contexte passif

7.2 (B)-DEFINITION DU CONTEXTE PASSIF

7.2.1 Objectifs

Il s'agit de décrire et restituer le contexte passif général du système dans un environnement externe plus global. Il s'agit de décrire les relations d'échange entre les entités passives et le système, et entre le système et les autres systèmes mis en évidence à l'activité précédente.

7.2.2 Intervenants

La définition du contexte est effectuée en commun par l'équipe d'analyse.

7.2.3 Entrées

Sorties de l'activité A.

7.2.4 Sorties

Diagramme de flux de données statiques,
Diagramme(s) de flux de données dynamiques.

7.2.5 Description

7.2.5.1 introduction

Cette activité a pour but de formaliser les échanges entre les entités passives, les autres systèmes et le système étudié sous ces aspects statiques ou dynamiques suivant l'importance des aspects chronologiques des interactions entre ces éléments.
Cette activité peut déclencher une itération vers l'activité A en mettant en évidence l'existence d'une nouvelle entité passive ou un autre système.

7.2.5.2 Comment réaliser les sorties

Le **diagramme de flux de données statique** (représenté par un diagramme de collaboration) doit montrer les interactions et échanges de données réalisés entre le système à étudier et les entités passives identifiées précédemment. Le système est représenté par une classe et les données échangées sont figurées sur le diagramme par des flèches courtes surmontant les liens entre les entités passives et le système.

Le (ou les) **diagramme(s) de flux de données dynamique** (représenté par un diagramme de séquence) doit montrer les dépendances ou les relations chronologiques existant entre le système et les entités passives. Il permet ainsi de décrire de façon temporelle à quels moments sont consommées ou produites les interfaces en entrée/sortie du système.

Remarques :

Dans tous ces diagrammes, le système est représenté par une classe (au sens UML) ; cette classe est une abstraction car elle n'est jamais instanciée.

Ces diagrammes sont complétés à l'étape suivante par un diagramme général de flots de données qui lui représente non seulement les échanges de données entre le système et les entités passives, mais aussi ceux entre le système et les entités actives (acteurs), qui auront été mis en évidence par les cas d'utilisations.

7.2.6 Règles de vérification

Les entités présentes dans les diagrammes de flux de données statiques ou dynamiques sont des entités passives et doivent être stéréotypées comme tel hormis la classe représentant le système.
Toutes les entités présentes dans ces diagrammes doivent être liées au système pas un flot de données.

7.2.7 Exemples

Figure 6 - Exemple de diagramme de flux de données statique

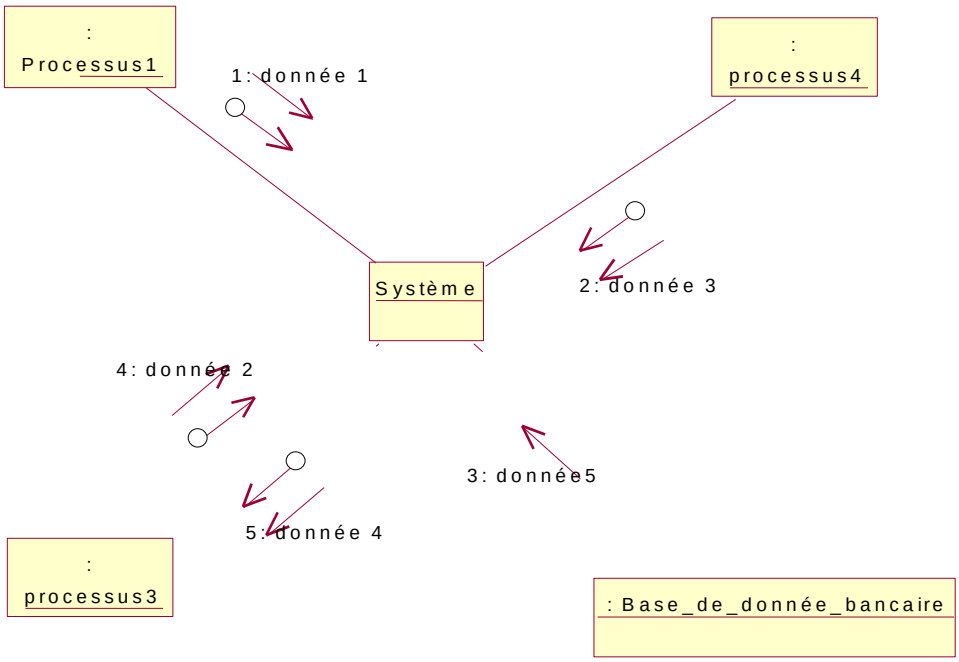
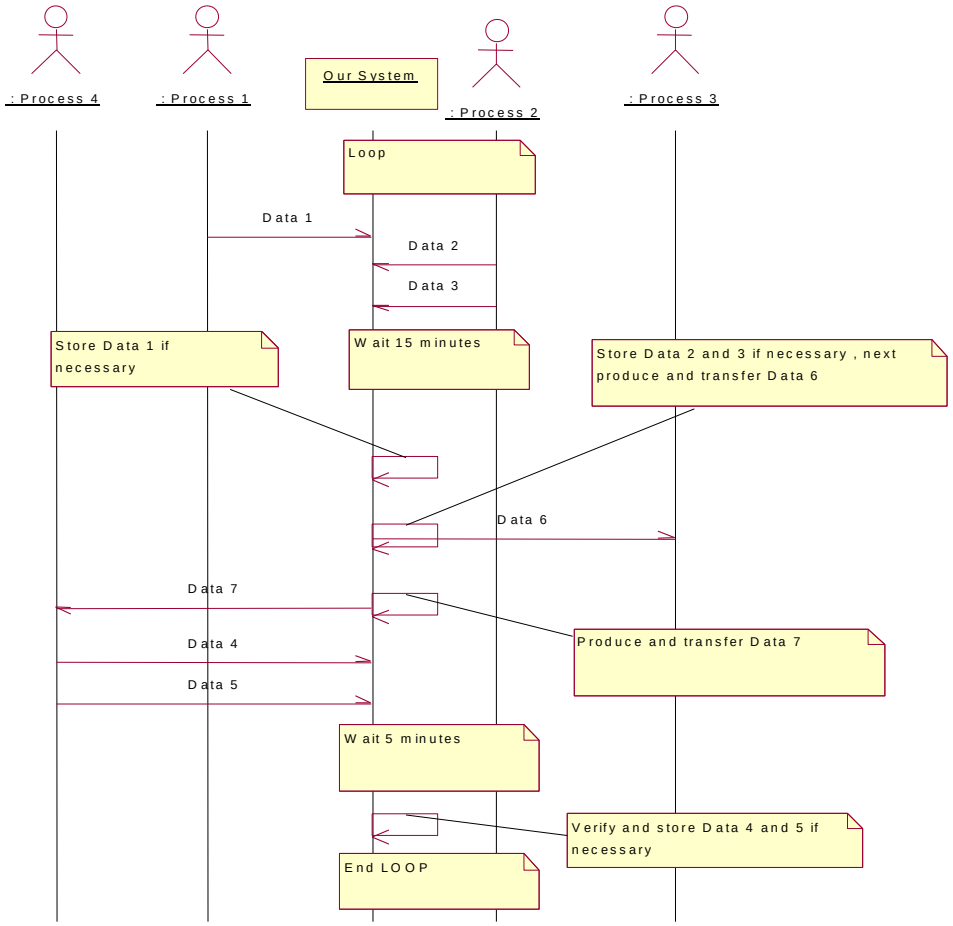


Figure 7 - Exemple de diagramme de flux de données dynamique



7.3 (C)-DESCRIPTION DU SYSTEME

C Description du système

Cette activité très riche est scindée en 3 sous-activités que sont :

C1 : détermination des cas d'utilisation,

C2 : détermination des scenarii pour chacun des cas d'utilisation,

C3 : réalisation de l'analyse du domaine.

Selon le déroulement de l'analyse du projet, les activités C2 et C3 peuvent être conduites dans l'ordre C2 puis C3, ou bien C3 puis C2, suivant que le domaine est connu C3 avant C2 et donc utile à l'élaboration des scenarii ou que les classes du domaine sont identifiées lors de la création de ces mêmes scenarii.

7.3.1 C1-Détermination des cas d'utilisation

7.3.1.1 Objectifs

Il s'agit d'identifier les différents cas d'utilisation du système et de les décrire textuellement.

7.3.1.2 Intervenants

Les cas d'utilisation sont déterminés par l'ensemble de l'équipe d'analyse ; leur description est ensuite répartie entre chacun des analystes.

7.3.1.3 Entrées

Sorties de l'activité B.

7.3.1.4 Sorties

Diagramme général des cas d'utilisation,
Description textuelle des cas d'utilisation,
Diagramme général de flots de données.

7.3.1.5 Description

7.3.1.5.1 introduction

Suite à la détermination des entités participant au fonctionnement du système à étudier, c'est-à-dire le QUI, notre processus propose d'étudier le QUOI par la mise en place des **cas d'utilisations**. Ces Use Case correspondent aux spécifications fonctionnelles du système. Grossièrement un **Use Case** représente une **fonctionnalité** ou une sous fonctionnalité du système étudié.
La détermination des Use Case conduit en général à la mise en évidence de nouvelles entités intervenant dans le système et provoque donc de nouvelles itérations : par exemple la mise en évidence d'un nouvel acteur conduit à une itération sur la phase A.
Le détail des Use Case (par la création des scénarii) fait apparaître les flots de données entre acteurs et système et donc conduit à la création du diagramme général de flot de donnée.

7.3.1.5.2 Comment réaliser les sorties

Du document des exigences on va extraire les fonctionnalités du système à étudier et donc établir les différents Use Case et créer le diagramme général des cas d'utilisation. Le **diagramme général des cas d'utilisation** (qui est un diagramme de cas d'utilisation) regroupe tous les Use Case identifiés lors de cette activité. Chaque acteur identifié lors de l'activité A doit produire n Use Case hormis les acteurs n'ayant qu'un but de représentation des héritages. Comme dans le cas des acteurs, on peut créer des Use Case dont le but est de factoriser des fonctionnalités et donc voir apparaître des arbres d'héritage des Use Case.

Un cas d'utilisation représente une fonctionnalité du système ; elle est activée par l'un des acteurs définis lors de l'activité A.
Les cas d'utilisation sont organisés en cas d'utilisation **principaux** et **secondaires**, afin de permettre :

L'utilisation par un cas d'utilisation d'un autre cas d'utilisation déjà défini,

La centralisation au sein d'un cas d'utilisation d'activités présentes dans plusieurs cas d'utilisation,

L'établissement entre cas d'utilisation de relations de type "utilise obligatoirement" ou "utilise optionnellement",

Le découpage en plusieurs cas d'utilisation de cas d'utilisation complexes.

Les cas d'utilisation sont éventuellement regroupés en paquetages, permettant d'exprimer des regroupements thématiques par exemple.

Les cas d'utilisation sont décrits de façon textuelle en respectant les rubriques suivantes :

Résumé	Présentation succincte du cas d'utilisation
Contexte d'utilisation	Conditions d'utilisation par les éléments déclenchant (fréquence d'activation, déclenchement synchrone ou asynchrone …)
Elément déclenchant	Acteur ou cas d'utilisation
Pré-conditions	Etat stable du système nécessaire avant l'accomplissement du cas d'utilisation
Données en entrée	Données consommées
Description	Description détaillée des interactions entre les éléments déclenchant et le système
Post-conditions	Etat stable atteint par le système à la fin de l'accomplissement du cas d'utilisation
Données en sortie	Données produites
Exceptions	Cas d'erreur que le système ne sait pas résoudre

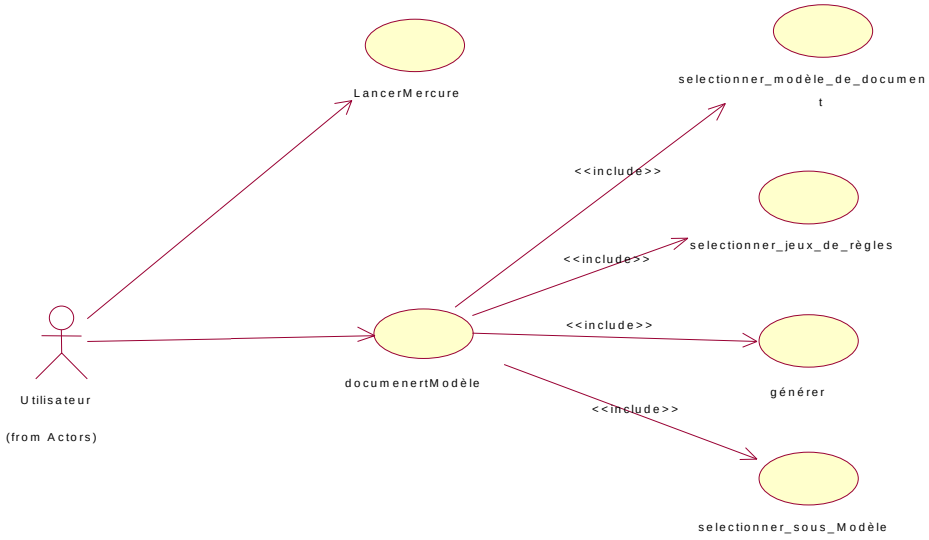
Une fois les cas d'utilisation identifiés, on peut créer le diagramme général de flots de données.
C'est un diagramme de collaboration représentant tous les échanges de données entre acteurs, entités passives, autres systèmes et le système. Il montre les limites du système dans son environnement et les limites de la modélisation.
Il peut être considéré comme la somme du diagramme de flux de données statiques et des Use Case impliquant un acteur ; de ce fait, il doit contenir les acteurs, entités passives, autres systèmes échangeant des données avec le système.

7.3.1.6 Règles de vérification :

Détermination des cas d'utilisation :
Tous les cas d'utilisation doivent être déclenchés par un acteur ou un cas d'utilisation.
Tous les cas d'utilisation doivent être documentés
Diagramme général de flots de données :
Toute entité échangeant des données doit se retrouver dans ce diagramme, soit tout acteur déclenchant un cas d'utilisation toute entité passive ou autre système présent dans le diagramme de flux de données statiques.

7.3.1.7 Exemples

Figure 8 diagramme de Use Case



7.3.2 C2-Détermination des scénarios

7.3.2.1 Objectifs

Il s'agit de détailler les cas d'utilisation dans des contextes particuliers de sollicitation par les éléments déclenchants (nominaux, dégradés).

7.3.2.2 Intervenants

C'est le membre de l'équipe chargé du cas d'utilisation à détailler.

7.3.2.3 Entrées

Sorties de la sous-activité C1.

7.3.2.4 Sorties

Diagrammes de séquence.

7.3.2.5 Description

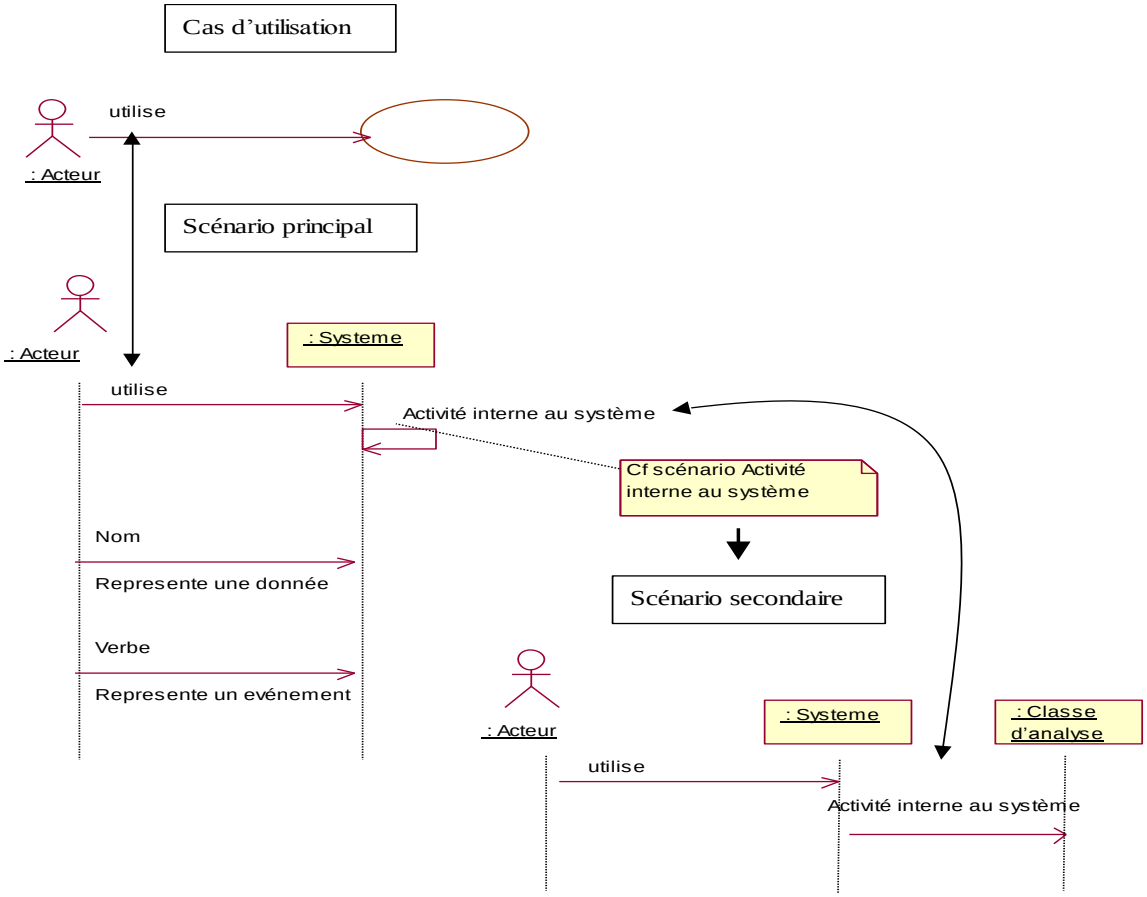
7.3.2.5.1 Introduction

Les Use Case représentant les fonctionnalités du système, il est fréquent qu'ils soient composés de plusieurs activités et donc doivent être décrits. Pour cela des diagrammes de séquence sont réalisés, ils mettent en relation les acteurs déclenchant les Use Case ainsi que toutes les entités passives, autres systèmes et classes identifiés et impliqués dans la réalisation de ces fonctionnalités. Suivant leur complexité, ces Use Case sont décrits par un scénario principal, un ou plusieurs scénarios secondaire ou comme il est dit dans le préambule par des diagrammes d'activités.

7.3.2.5.2 Comment réaliser les sorties

Les scenarii sont réalisés quand ils apportent un plus par rapport à la description textuelle des cas d'utilisation. Ils sont organisés en scenarii **principaux** et **secondaires**.
Un scénario **principal** (qui est un diagramme de séquence) a pour but de décrire temporellement les échanges entre un acteur et le système, en termes de données, de requêtes faites au système, de résultats de ces requêtes. Le scénario principal porte le nom du cas d'utilisation qu'il détaille de même le premier lien entre l'acteur déclenchant le cas d'utilisation porte lui aussi le nom du Use Case. Il met en jeux le (ou les) acteur(s) actif qui le déclenche ainsi que toutes les classes d'analyse participant à sa réalisation.
Un scénario **secondaire** (qui est un diagramme de séquence) permet de décrire une action interne du système, sous la forme d'interactions entre classes. L'établissement d'un scénario secondaire peut nécessiter d'avoir accompli au préalable l'activité C3 d'analyse du domaine. Le scénario secondaire porte le nom qui lui a été donné dans le scénario primaire de même que le premier lien entre le système et la classe d'analyse concernée.

Figure 9 Cas d'utilisation Scénarios primaires et secondaires



7.3.2.6 Règles de vérification

Les scénarii sont utilisés s'il détaillent un Use Case.

Les scenarii sont subdivisés en scénario primaires et secondaires ou remplacés par un diagramme d'activité suivant la complexité des traitements qu'ils détaillent.

Le scénario principal porte le nom du cas d'utilisation qu'il détaille.

Le premier lien entre l'acteur déclenchant le cas d'utilisation porte le nom du Use Case qu'il détaille.

Le scénario principal contient l'acteurs actifs qui le déclenche ainsi que toutes les classes d'analyse participent à sa réalisation.

Le scénario secondaire porte le nom qui lui a été donné dans le scénario primaire.

Le premier lien du scénario secondaire entre le système et la classe d'analyse concernée porte le nom du scénario secondaire.
Le nom des liens représente une donnée si c'est un nom, un évènement si c'est un verbe ou un appel de méthode si le verbe est suivit de parenthèses.

7.3.2.7 Exemples

Figure 10 - Scénario principal

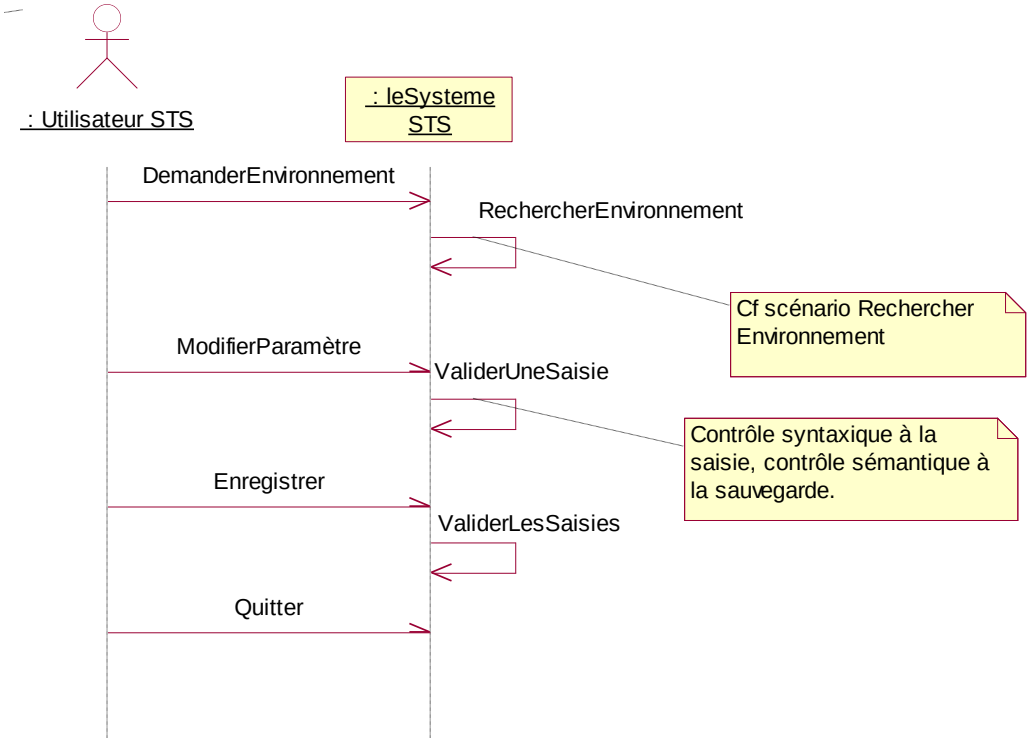
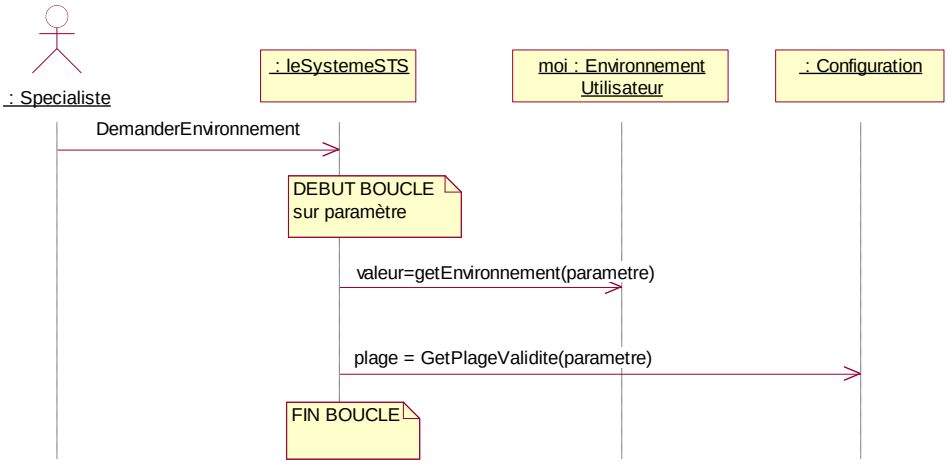


Figure 11 - Scénario secondaire



7.3.3 C3-Analyse du domaine

7.3.3.1 Objectifs

Il s'agit de déterminer les classes communes qui apparaissent suite à l'analyse des spécifications du système.

7.3.3.2 Intervenants

Les classes du domaine sont identifiées par l'ensemble de l'équipe d'analyse. Cette étape doit faire intervenir les experts du domaine.

7.3.3.3 Entrées

Spécifications du système.

7.3.3.4 Sorties

Diagramme des classes du domaine.

7.3.3.5 Description

7.3.3.5.1 Introduction

Les applications informatiques sont dédiées à des domaines particuliers qui utilisent des concepts qui leurs sont spécifiques. Ces concepts correspondent à des classes que l'on peut qualifier de classes métiers lors de la phase d'analyse. La réalisation des diagrammes de séquence liés aux Use Case conduit à mettre en évidence des classes dont certaines peuvent correspondre à ces classes du domaine, parfois c'est l'inverse, c'est à dire que les classes du domaine sont identifiées avant la mise en place des Use Case.

7.3.3.5.2 Comment réaliser les sorties

Une **classe du domaine** représente un élément de la spécification qui se doit d'être représenté dans la description du système.
La détermination des classes du domaine permet d'adopter un "vocabulaire" commun pour le projet, par l'établissement d'une et une seule représentation pour un élément de spécification.
Les classes du domaine sont représentées sur un (ou plusieurs) diagramme(s) de classes sur lequel il est important de faire figurer les liens de composition, d'agrégation, d'utilisation entre les classes.

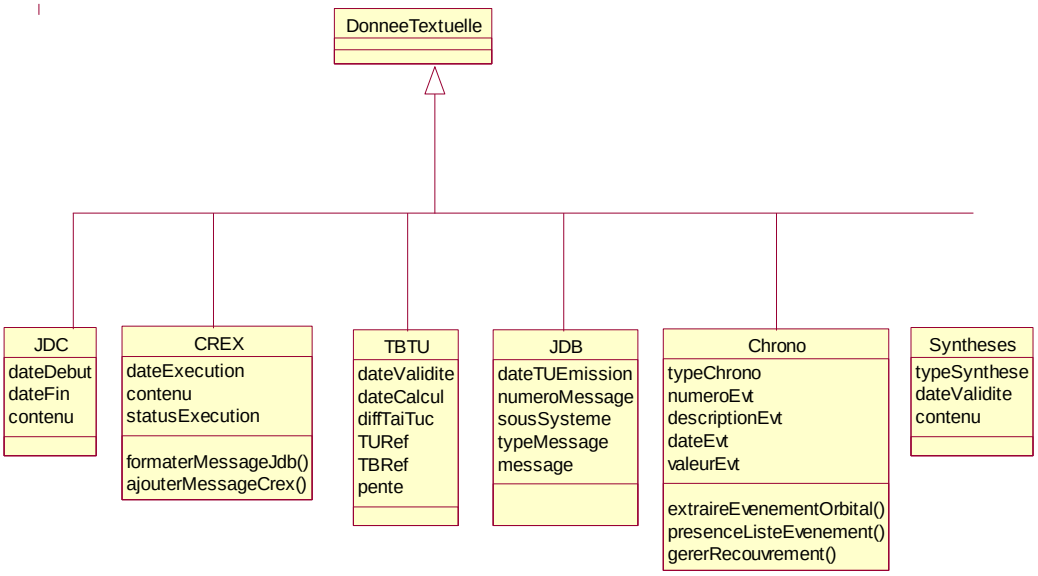
7.3.3.6 Règles de vérifications

Cette activité est réalisée par les experts du domaine ; étant donné la variété des cas possibles, le processus n'énonce aucune règle particulière, les contraintes à respecter sont les règles liées aux diagrammes de classe.

7.3.3.7 Exemples

Classe "Satellite".
Classe "Donnée manipulée par le système".

Figure 12 diagramme de classes du domaine



7.4 FIN DE LA PHASE D'ANALYSE DU BESOIN

Une fois les activités A, B et C réalisées, la phase d'analyse des besoins se termine ; elle a permis de déterminer le « QUI » et le « QUOI » c'est à dire ce que va faire le logiciel et qui interviendra dans son fonctionnement. A ce moment du processus on va tester le modèle et produire de la documentation puis passer à l'analyse objet cette dernière étape de l'analyse est un point d'incrémentation important qui va lui aussi demander le test du modèle du point de vue méthodologique ainsi que la production d'une documentation.

A ce niveau et avant de passer à l'itération suivante, il faut faire une vérification de la phase précédente en vérifiant la conformité à la méthode sur les points suivants :

Règles concernant les diagrammes d'acteurs, entités passives et autres systèmes :

Le diagramme général des acteurs ne doit contenir que des entités déclenchant directement des fonctionnalités du système ou dont héritent des entités déclenchant directement des fonctionnalités du système. Etant donné que l'on se trouve en début du processus cette vérification se fait à priori puisque aucuns cas d'utilisation n'a encore été déterminé, on peut cependant vérifier que le stéréotype des acteurs (au sens UML) présents dans ce diagramme est bien <<Acteur>>.
Le diagramme général des entités passives ne doit contenir que des entités ne déclenchant pas directement des fonctionnalités du système ou dont héritent des entités ne déclenchant pas directement des fonctionnalités du système. De la même façon les entités présentes dans ce diagramme doivent être stéréotypées soit << entité passive >> soit << autre système >>.
Toute entité doit être dûment stéréotypée et documentée.

Règles concernant les diagrammes de flux de données :

Les entités présentes dans les diagrammes de flux de données statiques ou dynamiques sont des entités passives et doivent être stéréotypées comme tel hormis la classe représentant le système.
Toutes les entités présentes dans ces diagrammes de flux de données doivent être liées au système par un flot de données.

Règles concernant les Use Case :

Tous les cas d'utilisation doivent être déclenchés par un acteur ou un cas d'utilisation.
Tous les cas d'utilisation doivent être documentés.

Règles concernant le diagramme général de flots de données :

Toute entité échangeant des données doit se retrouver dans ce diagramme, soit tout acteur déclenchant un cas d'utilisation, toute entité passive ou autre système présent dans le diagramme de flux de données statiques.

Règles concernant les scenarii :

Les scénarii sont utilisés s'ils détaillent un Use Case.
Les scenarii sont subdivisés en scénario primaires et secondaires ou remplacés par un diagramme d'activité suivant la complexité des traitements qu'ils détaillent.
Le scénario principal porte le nom du cas d'utilisation qu'il détaille.
Le premier lien entre l'acteur déclenchant le cas d'utilisation porte le nom du Use Case qu'il détaille.
Le scénario principal acteurs actifs qui le déclenche ainsi que toutes les classes d'analyse participent à sa réalisation.
Le scénario secondaire porte le nom qui lui a été donné dans le scénario primaire.
Le premier lien du scénario secondaire entre le système et la classe d'analyse concernée porte le nom du scénario secondaire.
Le nom des liens représente une donnée si c'est un nom, un évènement si c'est un verbe, un appel de méthode si le verbe est suivit de parenthèses.

7.5 MODELES DE DOCUMENTS D'ANALYSE

Ce chapitre contient les modèles de documents à produire en fin de phase d'analyse, ainsi que les règles applicables à la phase d'analyse en vue de la production de documentations spécifiques comme un plan de validation ou un manuel d'interface.
L'analyse des besoins consiste à l'extraction des demandes des clients par les personnes chargées de réaliser les logiciels ; c'est aussi, une fois l'analyse objet faite (étape suivante), le point de départ de la conception. Elle nécessite d'être validée par les clients ; pour ce, il est nécessaire de produire un document d'analyse représentant la modélisation ; notre processus propose une organisation de ce document qui se base sur la structure du modèle réalisé par l'application du processus.
Le plan d'organisation de la documentation présenté ci-dessous est un extrait de celui que l'on retrouve dans le chapitre « Documentation » ; il doit être recréé à chaque fois qu'il y a une itération pour rester fidèle au modèle. Il doit comprendre un glossaire du projet permettant la compréhension du modèle par tous les intervenants du projet soit les analystes, concepteurs ou autres comme les clients. Il doit décrire les exigences fonctionnelles le contexte du système, la description des intervenants vis à vis du système, il doit aussi présenter l'analyse du domaine, les cas d'utilisation et l'architecture physique.
Soit la proposition de plan suivante :

CONTENU	ACTIVITES	DESCRIPTION
Glossaire	A-B-C	Glossaire technique du projet
Objectifs du système	A-B-C	Définition des principales exigences fonctionnelles.
Contexte du système	B	Description des interactions entre les entités externes et le système.
Définition des acteurs	A	Définition de chaque acteur, hiérarchies d'acteurs éventuellement construites, diagrammes d'interaction les concernant.
Analyse du domaine	C	Identification des interactions entre acteurs et cas d'utilisation. Identification des objets métier.
Modèle par cas d'utilisation	B-C	La description des cas d'utilisation se base sur le modèle fourni en annexe. Elle peut être organisée par package. Pour chaque cas d'utilisation, on inclue les diagrammes utilisés : diagrammes de séquence, diagrammes de collaboration, diagrammes partiels de classes.
Architecture physique	α	Identification des principaux éléments physiques du système (diagrammes de déploiement).

Le respect des règles méthodologiques permet la génération automatique d'un tel document à partir du modèle lui même en garantissant que le modèle contient les informations requises.
La génération d'un plan de validation ou d'un manuel d'interface nécessite le respect de nouvelles règles spécifiques : à détailler.

7.6 COMPOSANTS D'ANALYSE

Comme pour chaque activité ce chapitre définit les règles de production de composants d'analyse ainsi que les règles d'utilisation de ces composants.
Il faut bien avoir conscience que l'analyse du besoin doit être le plus détaché possible des solutions techniques. Les composants que l'on sera susceptible d'utiliser en phase d'analyse ne seront pas des composants instanciés mais plutôt des patterns décrivant des architectures imposées dans les besoins pour respecter ce détachement des solutions techniques. L'utilisation et la création des patterns sont détaillées dans le préambule.

8 ANALYSE OBJET

D Analyse objet

8.1 (D)-ANALYSE OBJET

8.1.1 Objectif

Il s'agit de déterminer les classes d'analyse, classes issues de l'analyse du besoin (diagrammes de séquences dérivant des Use Case, classes issues de l'analyse du domaine) et de les regrouper en paquetages.

8.1.2 Intervenants

La mise en place des paquetages est effectuée en commun par l'équipe d'analyse. La description de chaque paquetage est ensuite répartie aux différents membres.

8.1.3 Entrées

Classes du domaine,
Classes intervenant dans les diagrammes de séquence de la sous-activité C2.

8.1.4 Sorties

Diagramme d'objets participants pour chaque Use Case,
Diagrammes de paquetages,
Diagramme de classes de chacun des paquetages.

8.1.5 Description

8.1.5.1 Introduction

Une fois réalisée l'analyse du besoin on dispose, comme description du système, de l'identification des participants c'est à dire des entités liées au système de façon active ou passive (acteurs, entités passives et autres systèmes), des cas d'utilisation du système (Use Case), de l'enchaînement de ces Use Case et des activités qui les composent (scénario). De cette analyse on va extraire les classes identifiées lors de la création des diagrammes de séquence (scénario) et de l'analyse du domaine. Avant de passer à la conception on va regrouper et organiser toutes ces classes en paquetages pour créer ainsi le matériau de base de la conception. C'est l'objectif de cette activité ; il faut noter que lors de cette activité, on pourra identifier de nouvelles classes et donc produire des itérations sur les activités précédentes.

8.1.5.2 Comment réaliser les sorties

Pour chaque cas d'utilisation, on crée un **diagramme d'objets participants**, regroupant l'ensemble des classes mises en œuvre dans les diagrammes de séquence. Ces diagrammes de classe permettent de situer les interactions des objets internes du système.
La réunion des classes présentes dans les diagrammes d'objets participants et dans les diagrammes de classes du domaine conduit à la création du diagramme complet des classes d'analyse.

Au sein de ce diagramme, on crée des **regroupements de classes** dans des **paquetages**, selon des critères qui peuvent être :

regrouper les classes de forte cohérence,

limiter le couplage des classes entre plusieurs paquetages,

isoler dans un paquetage les classes en intersection de plusieurs diagrammes d'objets participants,

isoler dans un paquetage les classes destinées à être réutilisées,

exprimer au mieux les associations entre classes, pour bien identifier les futures dépendances entre les paquetages, qui doivent être simples. Généralement, on ne découpera pas les catégories sur des associations de type :

composition,

agrégation entre classes (les classes pour lesquelles des associations de ce type existent sont indissociables),

association de type 0 - n,

limiter le nombre de classes dans un paquetage (périmètre du paquetage plus restreint donc plus facile à appréhender),

limiter les boucles qui induisent une double dépendance entre paquetages : si un paquetage s'appuie sur un autre et inversement, un ou plusieurs paquetages tiers sont nécessaires.

8.1.5.3 Règles de vérification

Pour vérifier la réalisation de cette activité, on contrôle que :

- Toutes les classes identifiées lors de l'analyse du besoin sont présentes dans le diagramme complet des classes d'analyse ;
- Tous les Use Case ont un diagramme de classe correspondant à un diagramme d'objets participants (sauf exception faite des Use Case impliquant uniquement la classe fictive représentant le système).

On vérifie aussi que les paquetages sont correctement structurés en vérifiant toutes les règles de réalisation décrites précédemment :

- les classes de forte cohérence doivent être regroupées,
- le couplage des classes entre plusieurs paquetages doit être limité,
- les classes en intersection de plusieurs diagrammes d'objets participants doivent être isolées dans un paquetage,
- les classes destinées à être réutilisées doivent être isolées dans un paquetage,
- Il ne faut pas découper les catégories sur des associations de type :
- composition,
- agrégation entre classes (les classes pour lesquelles des associations de ce type existent sont indissociables),
- - association de type 0 - n,

- le nombre de classes dans un paquetage doit être limité (plus le périmètre d'un paquetage est restreint, plus il est facile à appréhender),

- Il faut limiter les boucles qui induisent une double dépendance entre paquetages : si un paquetage s'appuie sur un autre et inversement, un ou plusieurs paquetages tiers sont nécessaires.

8.1.6 Exemples

Figure 13 - Diagramme de paquetage

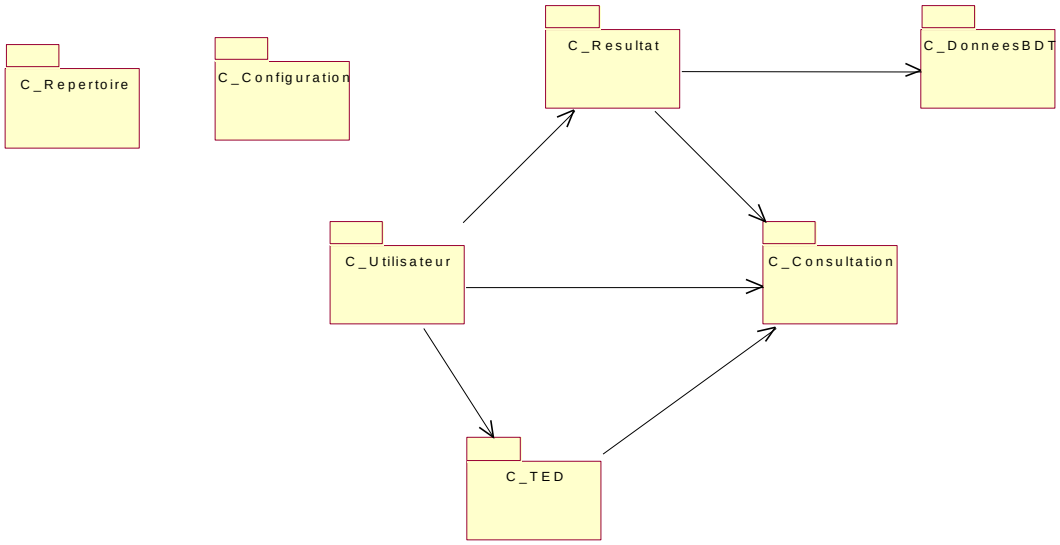
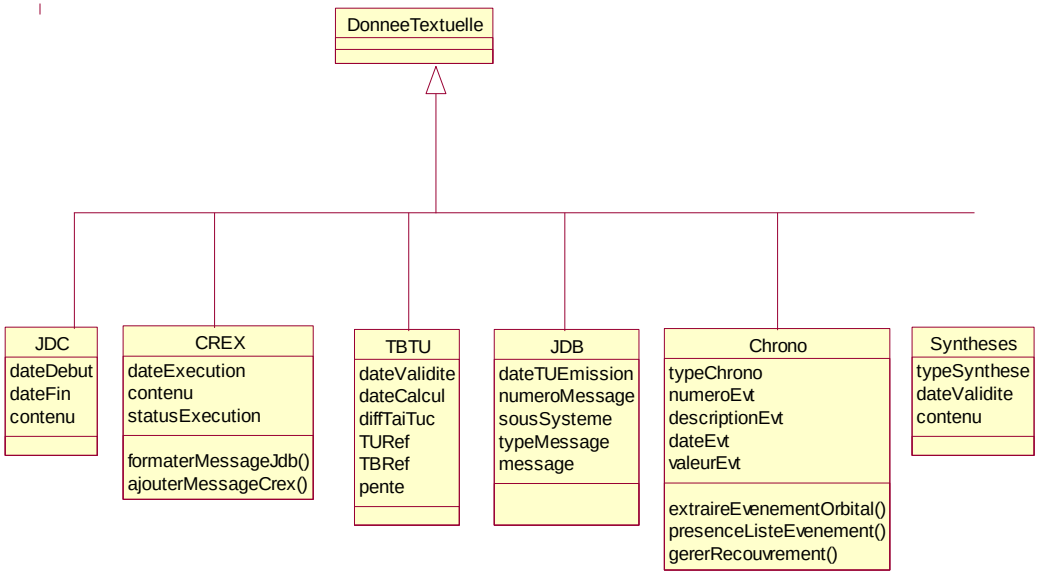


Figure 14 - Diagramme de classe



8.2 FIN DE LA PHASE D'ANALYSE OBJET

Une fois l'analyse objet réalisée, la phase d'analyse est complète. Le modèle d'analyse reflète les exigences fonctionnelles du système, ce modèle va servir de base à la conception lors de laquelle il pourra être modifié. En effet les contraintes de conception (contraintes non fonctionnelles) comme la réutilisation, l'évolutivité, le déploiement, les aspects temps-réel, … vont conduire dans certains cas à la modification du modèle.
Les vérifications à effectuer sont les mêmes que précédemment (cf chapitre « Fin de l'analyse des besoins ») complétées de celles de l'activité D (cf chapitre « Règles de vérification »).

8.3 MODELES DE DOCUMENTS D'ANALYSE OBJET

Pour les mêmes raisons que précédemment il faut produire à ce niveau une documentation ; c'est la même que celle générée en fin d'analyse des besoins incrémentée de la description de chaque paquetage et des classes qu'ils contiennent.
Soit l'ajour des éléments suivant au document existant :

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Glossaire	A-B-C	Glossaire technique du projet
Documents applicables	-	-
Documents de réference	-	-
Introduction	-	-
Objectifs du document	Démarrage	Description des points couverts par le document. Identification du niveau de détail de la conception.
Organisation du document	Démarrage	Description de la structure principale du document.
Présentation du système	-	-
Objectifs du système	A-B-C	Définition des principales exigences fonctionnelles.
Contexte du système	B	Description des interactions entre les entités externes et le système.
Définition des acteurs	A	Définition de chaque acteur, hiérarchies d'acteurs éventuellement construites, diagrammes d'interaction les concernant.
Analyse du domaine	C	Identification des interactions entre acteurs et cas d'utilisation. Identification des objets métier.
Modèle par cas d'utilisation	B-C	La description des cas d'utilisation se base sur le modèle fourni en annexe. Elle peut être organisée par package. Pour chaque cas d'utilisation, on inclue les diagrammes utilisés : diagrammes de séquence, diagrammes de collaboration, diagrammes partiels de classes.
Architecture physique	α	Identification des principaux éléments physiques du système (diagrammes de déploiement).
Architecture générale du système	**-**	**On détaille la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.**
Description des packages	**D**	**Rôles et contenu principal de chaque package.**
Identification des processus	**β**	**Diagrammes de composants.**
Traçabilité	-	-
Principes de traçabilité	Démarrage	Identification des matrices de traçabilité applicables (décrites en annexe).
Annexe 2 - Traçabilité	-	-
Exigences / cas d'utilisation	Avant D	Matrice de traçabilité entre les exigences utilisateur et les éléments d'analyse identifiés lors des activités A, B et C.

8.4 COMPOSANTS D'ANALYSE OBJET

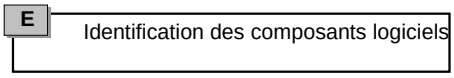
Pour les mêmes raisons (détachement vis a vis des solutions techniques) que dans la phase précédente les composants utilisés pour cette lors de l'analyse objet serons principalement des patterns plutôt que des composants instanciés. Ce chapitre à pour but de rappeler la possibilité d'utilisation de ces patterns que détailler les règles d'utilisation et de production de patterns ce qui est fait dans le préambule au chapitre « Utilisation des composants dans le processus ».

9 CONCEPTION DE L'ARCHITECTURE

L'analyse objet termine l'analyse proprement dite en extrayant du modèle d'analyse les classes d'analyse et en les structurant de façon logique en packages. Il en découle des diagrammes de paquetages ainsi que des diagrammes de classes décrivant à la fois la structure et les interactions des composants du système et la structure et les interactions des classes de chaque package.
Cette structure est reprise durant la conception ; elle est complétée et parfois modifiée comme on l'a vu précédemment. Les étapes de conception que nous allons réaliser décrivent comment mettre en œuvre ces transformations du modèle d'analyse vers le modèle de conception. Elles consistent en fait à un raffinage successif de l'architecture. En conception on ne voit pas apparaître de nouveaux diagrammes à chaque activité mais des diagrammes complétés.
Ce raffinage consiste à produire les interfaces des packages, décrire précisément le contenu des paquetages (classes), ajouter des paquetages de conception si nécessaire, appliquer des patrons d'architecture existant (patterns) le tout en favorisant la réutilisation de composants ainsi que leur production s'ils sont identifiés comme ayant un intérêt de ce point de vue. Tout ceci ce fait en respectant un certain nombre de règles que nous décrivons tout au long du processus de conception.

Les entrées dans la phase de conception sont multiples et dépendent de ce qui a été fait en analyse si le projet est de petite taille ; il se peut que les paquetages soient complets et ne nécessitent pas l'ajout de nouveau composant ou bien que les interfaces des paquetages aient été définies lors de l'analyse objet. Auquel cas, toutes les activités de la phase de conception ne sont pas réalisées. De même la découverte d'un nouveau composant dans l'une des activités de cette phase va provoquer une itération sur toute la phase. Les activités E, F, G, H peuvent être regroupées en une meta-activité qui est la conception de l'architecture ; elles ne seront pas réalisées de façon séquentielle : les liens qui les unissent sont forts et impliquent de nombreuses itérations pour être convenablement réalisées.

9.1 (E)-IDENTIFICATION DES COMPOSANTS LOGICIELS



9.1.1 Objectifs

Il s'agit de déterminer les **interfaces** et les **interactions** entre les différents paquetages issus de l'analyse objet qui deviennent des composants de conception.
Note : Durant cette phase, les paquetages issus de l'analyse peuvent être remis en cause si nécessaire.

9.1.2 Intervenants

La mise en place des interfaces et des interactions entre paquetages est effectuée en commun par l'équipe de conception.

9.1.3 Entrées

Sortie de l'étape précédente.

9.1.4 Sorties

Diagramme(s) de paquetage vue client /serveur.
Diagramme(s) d'interaction entre paquetages.

9.1.5 Description

9.1.5.1 Introduction

Lors de cette première étape de conception, on « valide » ou « invalide » l'architecture établie lors de l'activité D en réalisant les interfaces des paquetages et de fait les interactions des packages. Ceci a pour conséquence de revoir la structure des paquetages (comme nous l'avons vu précédemment, l'architecture d'analyse est généralement conservée). Pour cela on étudie la structure et les flots de données entre les paquetages d'analyse.

9.1.5.2 Comment réaliser les sorties

On peut subdiviser l'obtention des sorties en trois étapes identifier les interfaces des paquetages, analyser les flots de données, raffiner le contenu des paquetages.

9.1.5.2.1 Identification des interfaces.

Il s'agit de décliner (pour les paquetages d'analyse) les messages constituant l'interface en une ou plusieurs opérations (méthodes) ou en classes ou en objets. Pour cela, il faut raffiner les diagrammes de séquence issus de l'analyse (afin de faire apparaître les objets et les classes de conception) en décrivant comment sont réalisés les services offerts par l'interface du package.

Les contraintes liées à la mise en place de la solution informatique par le concepteur invalide parfois l'interface d'analyse. Il faut donc différencier deux parties les services offerts en phase d'analyse sans contraintes particulières et la transformation s'il y a lieu des services offerts pour satisfaire aux contraintes de conception.

Remarque : pour les paquetages de conception, l'interface est soit :

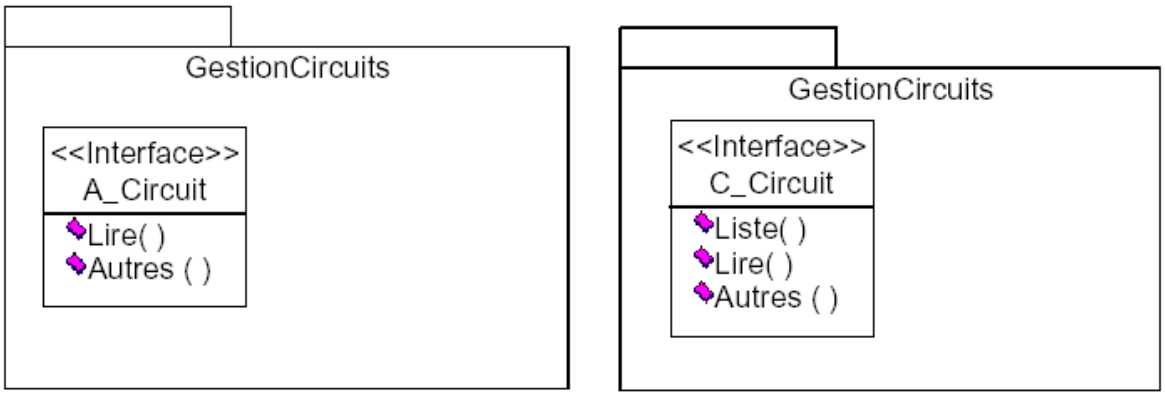
directement une classe constituée par des méthodes,

un ensemble de classes comme dans le cas de paquetages framework ou bibliothèques,

une classe façade.

Exemple :

Figure 15 Transformation des interfaces



Dans cet exemple, le message Lire défini en phase d'analyse se transforme en Liste + Lire en phase de conception. En effet, pour lire les circuits il faut tout d'abord en récupérer la liste puis sélectionner le circuit voulu et enfin le lire.

Une fois les interfaces identifiées on va pouvoir décrire les inter-actions entre les paquetages qui communiquent par le biais de ces interfaces en créant le ou les **Diagramme(s) d'interaction entre paquetages** qui est un diagramme de séquence mettant en relation les différents paquetages ; il faut aussi si nécessaire créer des diagrammes d'interaction entre paquetages à l'intérieur des sous paquetages.

On va aussi reprendre si nécessaire la structure des paquetages en observant le flux de données entre paquetage et en extrayant si besoin des paquetages communs.

9.1.5.2.2 analyse des flots de données

Dans cette étape on va analyser les flux de données circulant entre les différents paquetages (ces derniers sont composés de n classes et d'au moins une classe interface) à travers des diagrammes de séquence ou de collaboration.

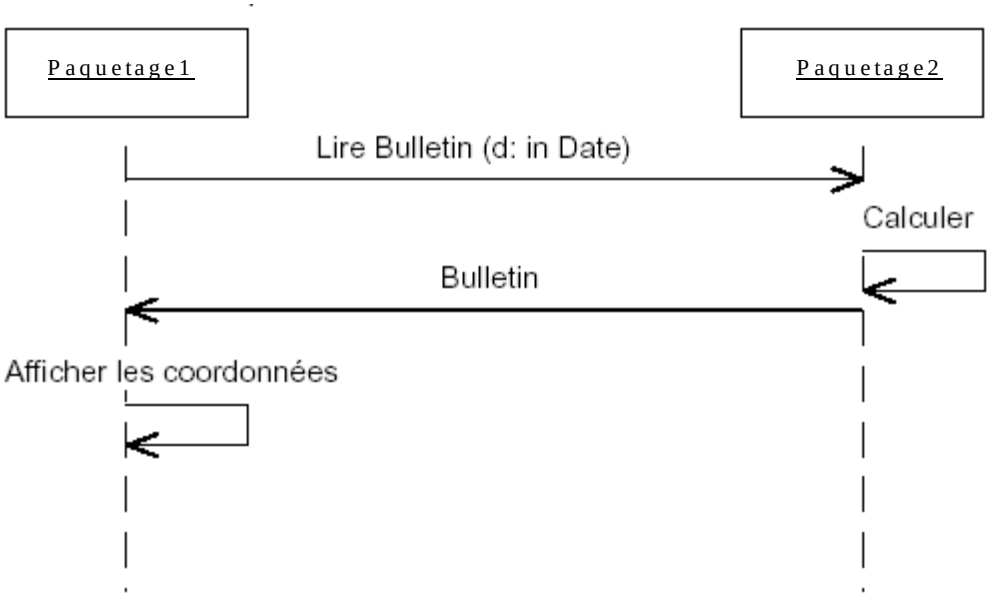
L'analyse du flux de données entre paquetages nous permet dans certains cas de créer de nouveaux paquetages dédiés aux traitements associés à la donnée échangée et ainsi de diminuer le couplage. Ces derniers sont composés par les classes du domaine issues de l'analyse ou par de nouvelles classes (non identifiées lors de l'analyse) donnant naissance à des données applicatives de niveau conception.
Cette modification d'architecture permet à l'ensemble des paquetages d'effectuer un accès centralisé et donc une meilleure portabilité.

Il s'agit d'examiner la complexité de chaque donnée échangée. Nous considérons les informations dites simples (entier, réel, date ...) et les informations dites complexes (bulletin d'orbite, circuit électrique, plan de traitement ...).
Les informations simples restent des flux de données, les informations complexes sont susceptibles de devenir des paquetages fournissant les services nécessaires au traitement de la donnée.

Remarque : Les données dites complexes (bulletin d'orbite, ...) sont souvent identifiées en phase d'analyse comme des classes du domaine. Nous pouvons traiter les échanges de données complexes comme suit :
La donnée est une instance de classe fournie par le "paquetage 1 ou 2", paquetage dédié à la gestion de cette donnée. Dans ce cas l'architecture reste identique.
La donnée est une instance de classe fournie par le "paquetage 1 ou 2", module non dédié qui réalise d'autres traitements non spécifiques à la donnée. Dans ce cas l'architecture est modifiée par la création d'un "paquetage" supplémentaire. La classe concernée et ses classes associées sont transférées dans le nouveau "paquetage " ; l'échange de l'instance entre paquetages reste identique.
La donnée est une instance unique fournie par le "paquetage 1 ou 2". Dans ce cas, l'architecture est modifiée par la création d'un paquetage supplémentaire. La classe concernée et ses classes associées sont transférées dans le nouveau paquetage ; il n'y a plus d'échange de données mais utilisation d'un paquetage unique qui gère la donnée.

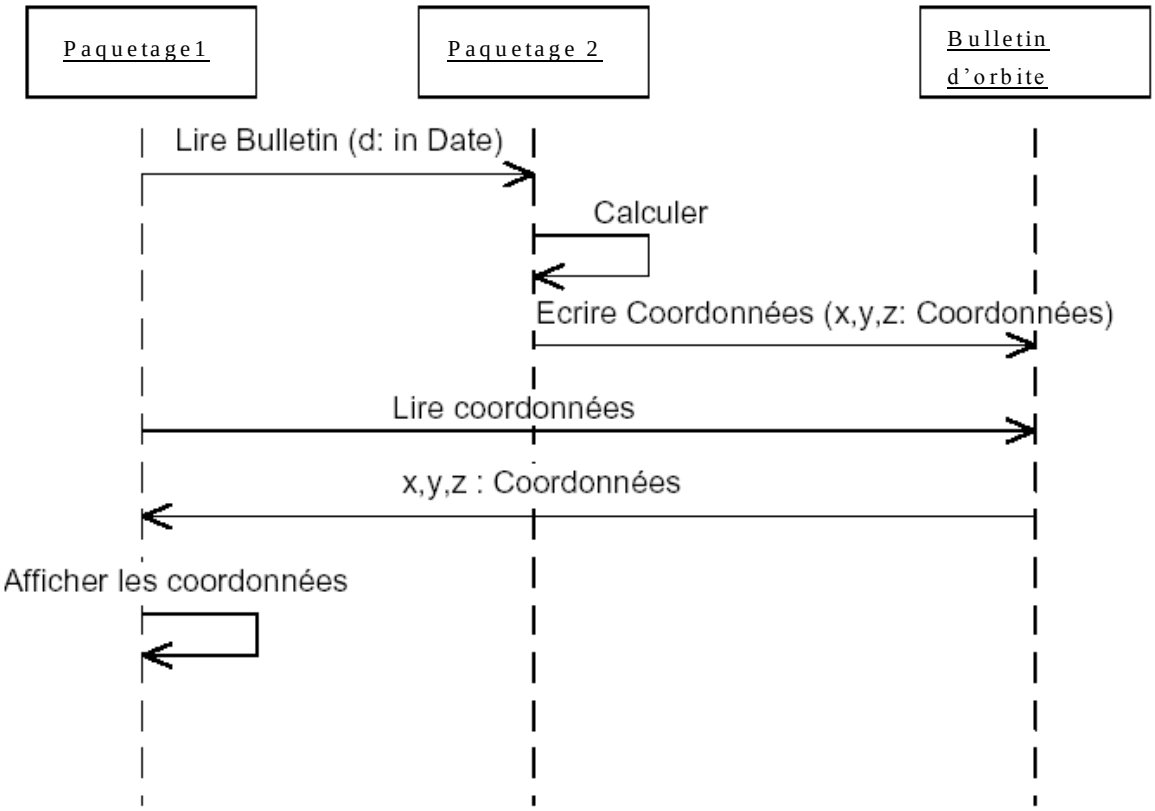
Exemple :
Dans l'exemple ci-après nous partons des hypothèses suivantes :
La classe Bulletin est une classe du domaine localisée dans "Module 2" (qui construit le bulletin) et qui doit être visible depuis "Module 1" (qui exploite le bulletin).
L'instance de Bulletin est unique.

Figure 16 Analyse du flot de données (1)



L'analyse du flux de données fait apparaître que la donnée Bulletin circule entre paquetage 1 et paquetage 2. Comme l'instance est unique et que Bulletin n'a aucun lien fort avec les classes de "paquetage 2", le concepteur choisit de créer un module dédié : "Bulletin d'orbite" (contenant la classe Bulletin et toutes les classes associées) fournissant les services nécessaires à sa manipulation (ex : lire, écrire coordonnées, fournir les accélérations ...).

Figure 17 Analyse du flot de données (2)

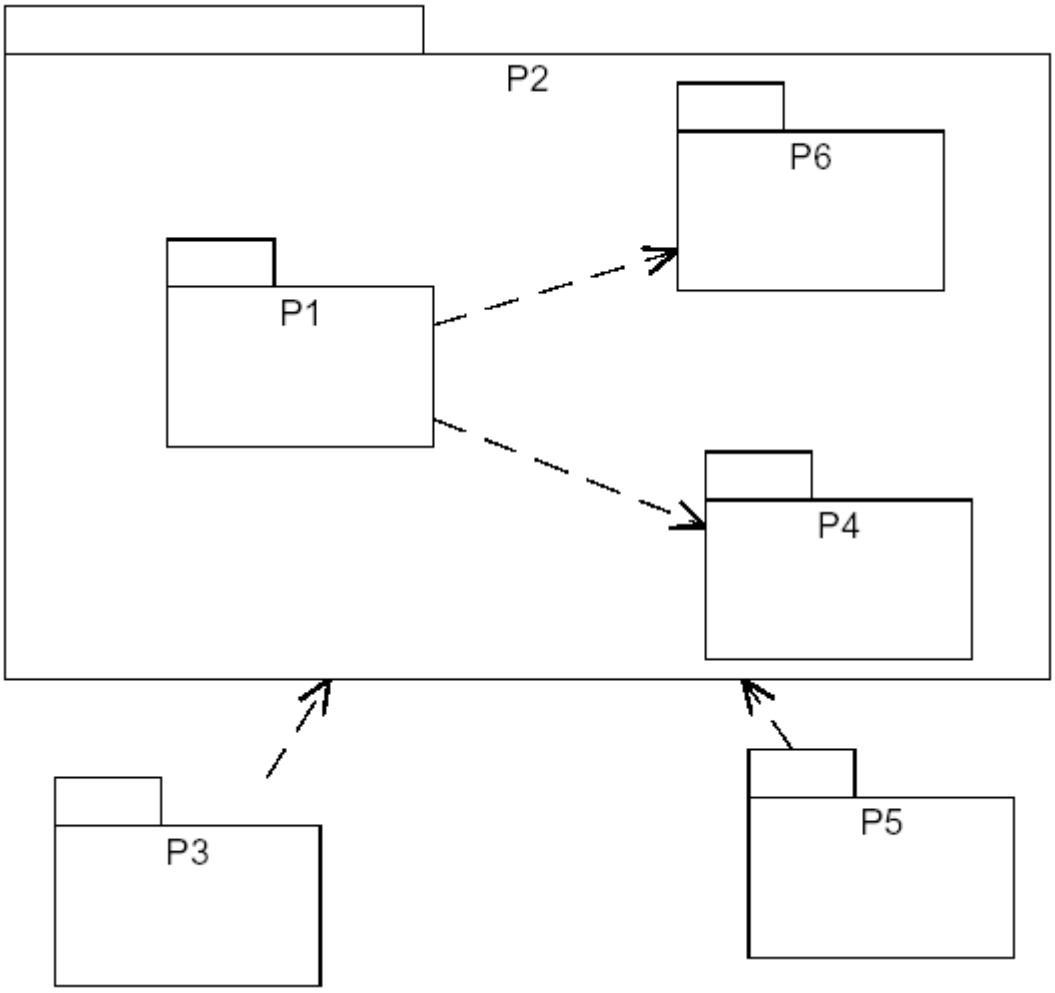


9.1.5.2.3 Raffinage du contenu des paquetages

L'étape suivante consiste à « raffiner » les paquetages obtenus en extrayant si nécessaire les paquetages communs.
Il s'agit de concevoir l'intérieur d'un paquetage et de remonter au niveau supérieur les paquetages dont un autre paquetage aurait besoin ou ceux considérés réutilisables (détectés à ce niveau par un besoin identifié en conception mais n'ayant aucun lien fort avec les autres paquetages de même niveau). Pour cela il est nécessaire de faire régulièrement le point (ou revues internes) afin de détecter très vite les paquetages communs et évaluer l'intérêt de les déplacer.
Cette approche permet de fabriquer des composants qui petit à petit, enrichissent l'architecture initiale. L'intérêt est de mieux partager le développement et de minimiser les phases de maintenance tout en facilitant la réutilisation.

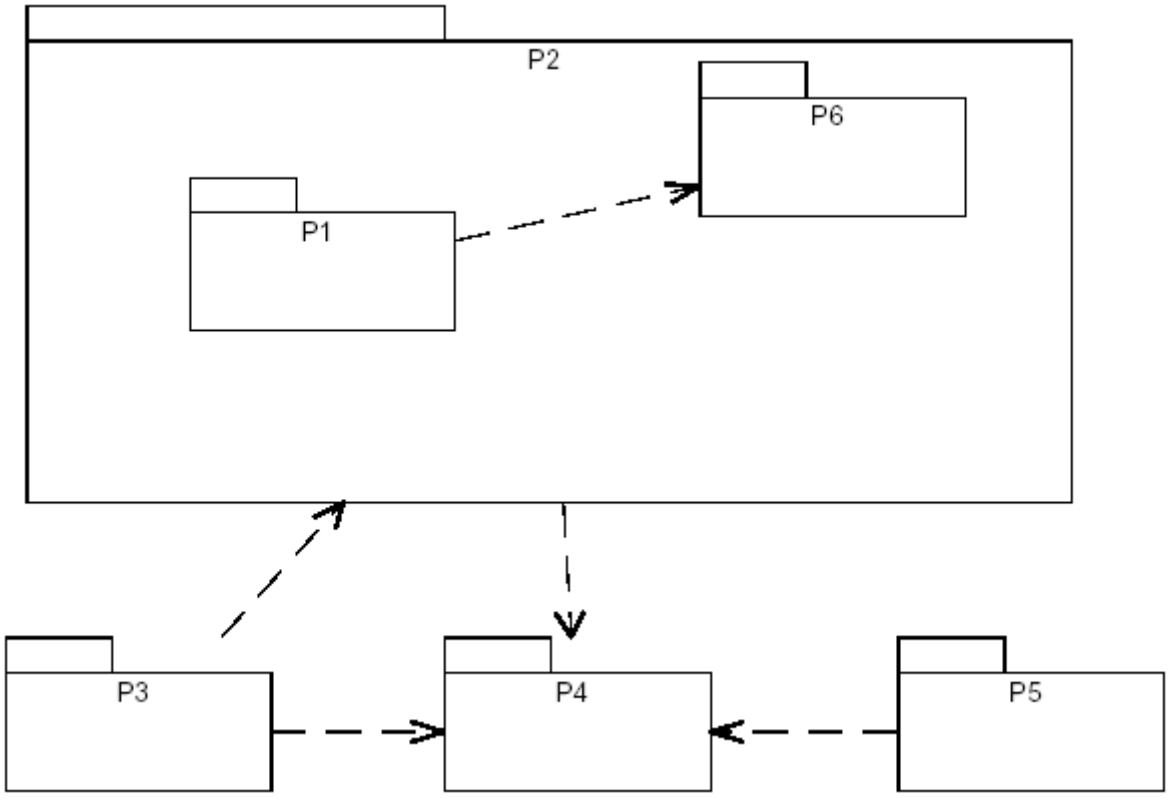
Exemple :

Figure 18. Raffinement de P2



L'exemple ci-dessus montre d'une part un paquetage P2 qui se décompose en trois paquetages P1, P4 et P6 et d'autre part, deux paquetages P3 et P5 qui utilisent P2.
Dans une telle architecture, nous pouvons extraire P4 de P2 (par exemple dans le cas où P5 n'a pas besoin des fonctionnalités offertes par P1 et P6, et que P4 est potentiellement réutilisable et non relié logiquement à P1).
P1 continuera à utiliser P4 et P6, et P5 utilisera P4 au lieu d'utiliser P2. Quant à P3, il utilisera P2 et P4.
Ce choix modifie l'architecture initiale tout en garantissant les mêmes fonctionnalités mais avec une plus value : mise en évidence d'un futur composant.

Figure 19. Modification de l'architecture initiale



Remarque : lorsque les équipes sont attentives au respect de cette règle, l'architecture initiale issue de l'analyse peut être largement modifiée, mais le plus souvent elle garde son ossature initiale.
Cependant un paquetage issu de l'analyse peut se transformer de manière importante et même disparaître en conception :
son contenu peut être organisé et distribué autrement (dans d'autres paquetages) ;
certaines parties peuvent devenir des composants ;
il peut éventuellement inclure le comportement d'une partie d'un autre paquetage et donc changer d'interface ;
les liens d'utilisation peuvent changer (ex : pour éliminer des cycles ou améliorer les performances).
...

9.1.6 Règles de vérification

Les règles méthodologiques à respecter pour cette activité sont :

Tous les paquetages doivent avoir au moins une classe interface.

Il faut identifier les flux de données inter et intra paquetage afin d'identifier de nouveaux paquetages si nécessaire.

Du fait qu'il y a une possible réorganisation des paquetages dans cette activité, on retrouve des règles rencontrées lors de l'activité D :

- les classes de forte cohérence doivent être regroupées,
- le couplage des classes entre plusieurs paquetages doit être limité,

- les classes en intersection de plusieurs diagrammes d'objets participants doivent être isolées dans un paquetage,
- les classes destinées à être réutilisées doivent être isolées dans un paquetage,
- il ne faut pas découper les catégories sur des associations de type :
- composition,
- agrégation entre classes (les classes pour lesquelles des associations de ce type existent sont indissociables),
- association de type 0 - n,
- le nombre de classes dans un paquetage doit être limité (plus le périmètre d'un paquetage est restreint, plus il est facile à appréhender),
- il faut limiter les boucles qui induisent une double dépendance entre paquetages : si un paquetage s'appuie sur un autre et inversement, un ou plusieurs paquetages tiers sont nécessaires.

9.1.7 Exemples

Figure 20 - Diagramme(s) de paquetage (composants de conception) vue client /serveur.

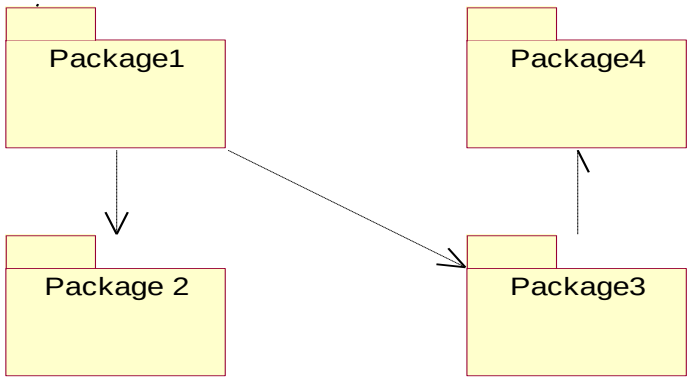
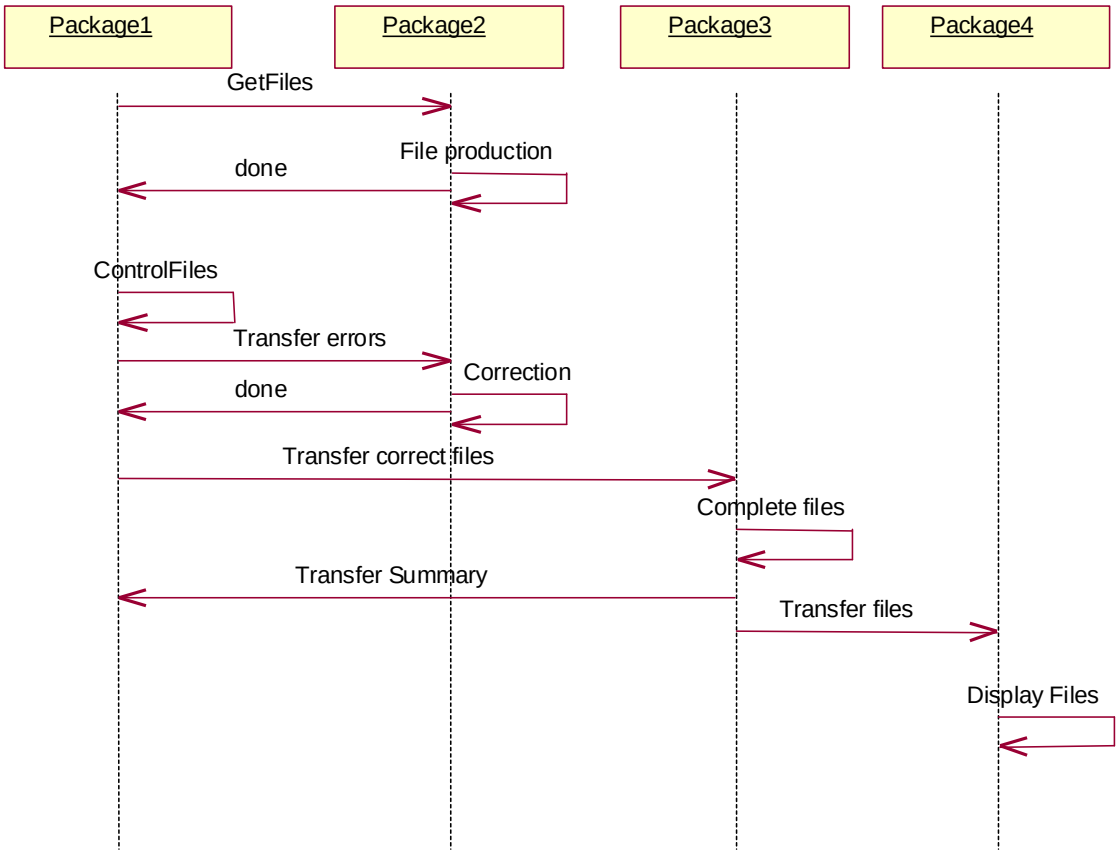


Figure 21 - Diagramme(s) d'interaction entre paquetages (composants de conception).



9.2 (F)-DESCRIPTION DES COMPOSANTS LOGICIELS

F Description des composants logiciels

9.2.1 Objectifs

Il s'agit de raffiner les paquetages (composants logiciels) obtenus dans la phase précédente.

9.2.2 Intervenants

La description de chaque paquetage (composants logiciel) est repartie entre les différents membres de l'équipe de conception.

9.2.3 Entrées

Diagramme(s) de paquetage (composants de conception).
Diagramme(s) d'interaction entre paquetages.

9.2.4 Sorties

Diagramme de classes, d'état et d'interaction pour chaque paquetage (composants de conception) si nécessaire.

9.2.5 Description

9.2.5.1 introduction

Cette activité consiste à détailler le mieux possible le contenu et le comportement du contenu des paquetages c'est-à-dire les classes. Il faut réaliser ces descriptions de façon statique (ajout de méthodes et d'attributs aux classes) et aussi de façon dynamique (par des diagrammes d'état).

9.2.5.2 Comment réaliser les sorties

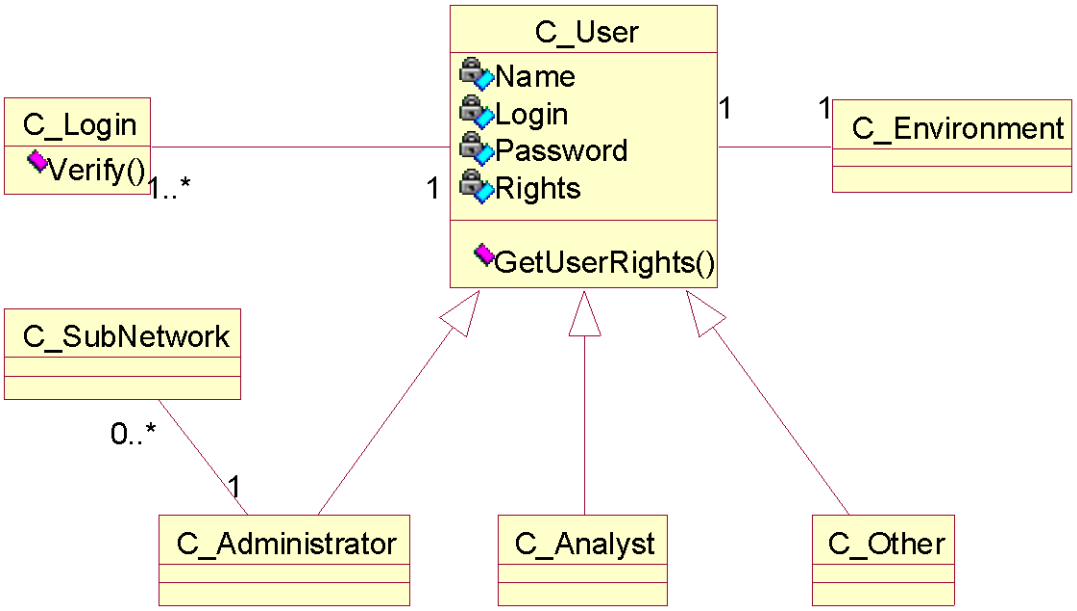
Il n'y a pas de règle spécifique pour réaliser cette activité ; à chaque paquetage est associé en fonction du besoin un ou plusieurs diagrammes d'état montrant le comportement des classes ; les attributs et les méthodes des classes doivent garantir leur aptitude à réaliser les services identifiés en analyse.

9.2.6 Règles de vérification

Là encore il n'y a pas de règles méthodologiques strictes. Il faut respecter les règles définies sur le projet comme par exemple des règles de nommage.

9.2.7 Exemples

Figure 22 – Diagramme de classes complété.



9.3 (G)-AJOUT DE COMPOSANTS DE CONCEPTION

G Ajout de composants de conception

9.3.1 Objectifs

Il s'agit de compléter l'architecture en ajoutant des composants de conception (existants ou nouveaux).

9.3.2 Intervenants

L'ajout de composants de conception et des interactions entre composants est effectué en commun par l'équipe de conception.

9.3.3 Entrées

Diagramme(s) de paquetage (composants de conception).
Diagramme(s) d'interaction entre paquetages.

9.3.4 Sorties

Diagrammes de paquetage (composants de conception) complétés.
Diagramme(s) d'interaction entre paquetages complétés.

9.3.5 Description

9.3.5.1 Introduction

La conception implique des contraintes liées à la solution mise en place ; ces solutions sont parfois communes à d'autre projets en réalisation ou déjà réalisés ce qui conduit logiquement à la réutilisation de composants déjà existants. C'est pourquoi notre méthodologie lui dédie une activité parmi les activités de la phase de conception de l'architecture.
Les composants de conception complètent le diagramme statique issu de l'analyse ou de la nouvelle architecture de conception et s'intègre dans la phase de conception en ajoutant les liens de dépendance entre les anciens et nouveaux composants.

9.3.5.2 Comment réaliser les sorties

Les composants de conception à ajouter ainsi que la façon de les créer sont décrits dans le préambule. Ils sont soit liés au choix de conception eux-mêmes (protocole TCP/IP,…) soit au métier auquel sont dédiés les application modélisées (journal de bord,…).
Nous donnons ci-après une liste indicative de composants de conception :
Composants offrant des mécanismes de type "utilitaire" regroupés la plupart du temps dans des bibliothèques réutilisables : dates, chaînes de caractères, structures de listes, algorithmes de tri, ...
Composants de communication entre processus permettant l'établissement de la connexion entre deux processus, la communication selon un protocole défini (TCP/IP, ...), le « marshalling » (ie : transformation des structures hiérarchiques en structures à plat) et le « unmarshalling » des informations ... ,
Composants offrant les services de Journal de Bord (écriture, consultation, ...), ou bien de présentation des messages utilisateurs ...,
Composants de gestion de mémoire dynamique,
Composants de traitement des erreurs,

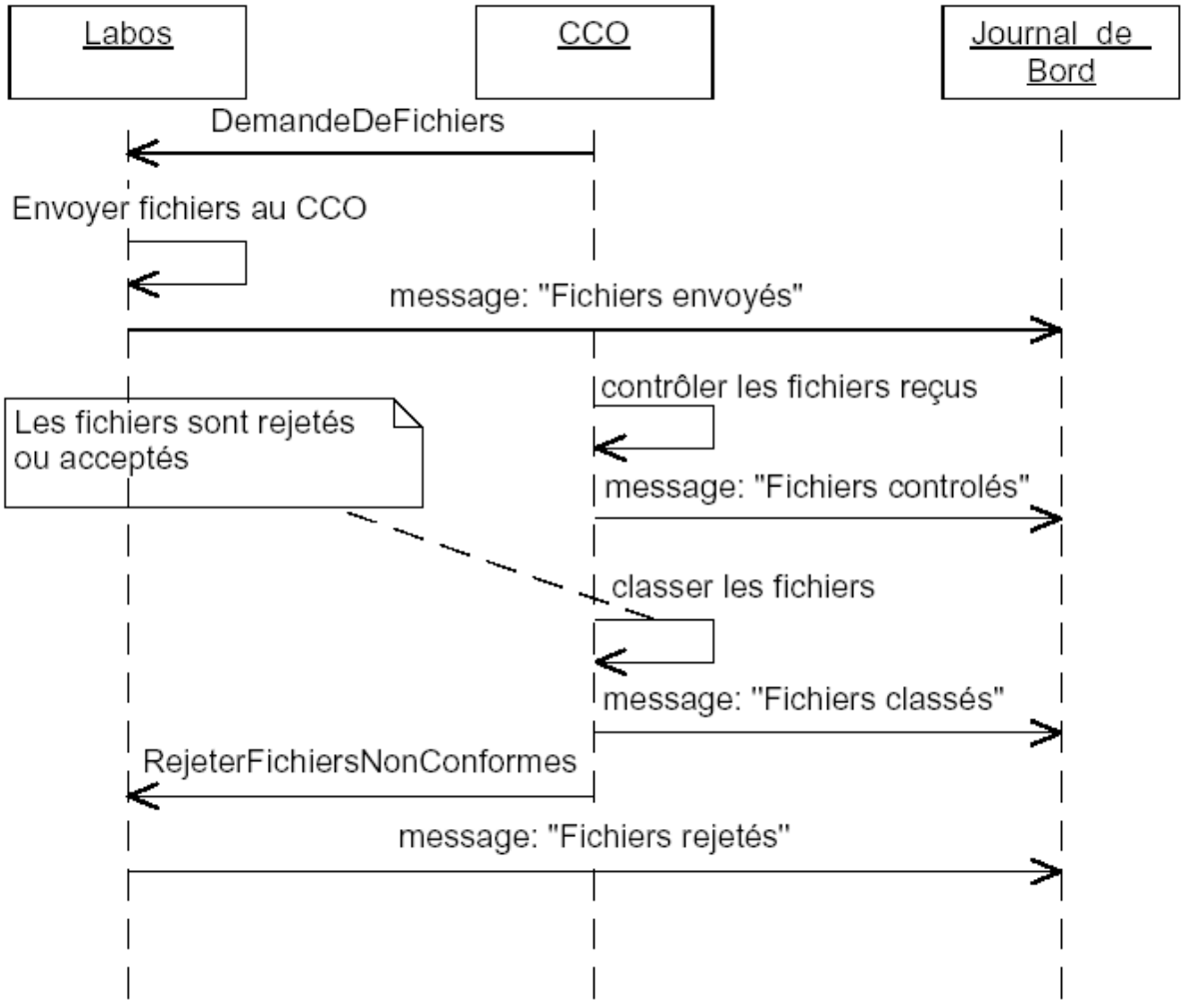
Composants offrant des ensembles de Widgets.

Remarque : la plupart de ces composants permettent non seulement de centraliser les traitements particuliers et diminuer les volumes de code, mais aussi visent à offrir un système dont les services sont homogènes pour l'utilisateur. Pour le cas des Composants de type « Widget », on pourra donc y trouver des composants graphiques réutilisables de niveau « commercial », de niveau « société » ou même de niveau « projet ».

Cette activité est mise en œuvre suite à l'identification lors de la conception (tout du long des étapes de conception) d'un composant manquant (pour assurer les besoins de la solution de mise en œuvre choisie) dés lors il faut rechercher parmi le catalogue de composants réutilisables les composants pouvant réaliser le besoin et le modifier si nécessaire.
De même, un composant dont l'intérêt est susceptible de dépasser la conception en cours doit être décrit et conçu de façon à être réutilisable par la suite (cf « Préambule utilisation des composants »).

Exemple d'utilisation :
L'analyse du projet MARS'96 (voir schéma) montre la collaboration entre des systèmes tels que CCO et Labos. Ces derniers sont représentés dans des diagrammes de séquence comme des objets instance des classes « interface » des différents paquetages (composants).
Dans notre exemple Labos et CCO sont apparus en phase d'analyse et « Journal de bord » en phase de conception. Le composant « Journal de bord » a été identifié comme un nouveau composant de conception.

Figure 23 Echange de données du système laboratoires vers le système CCO



Remarque :
"Journal de bord" est utilisé dans notre exemple comme un outil de trace permettant de mémoriser les différents traitements (arrêts corrects, avertissements, erreurs ...). Dans ce contexte il est considéré comme une classe de conception. En effet, il participe à la solution informatique mais sans être nécessaire au niveau de l'analyse.

9.3.6 Règles de vérification

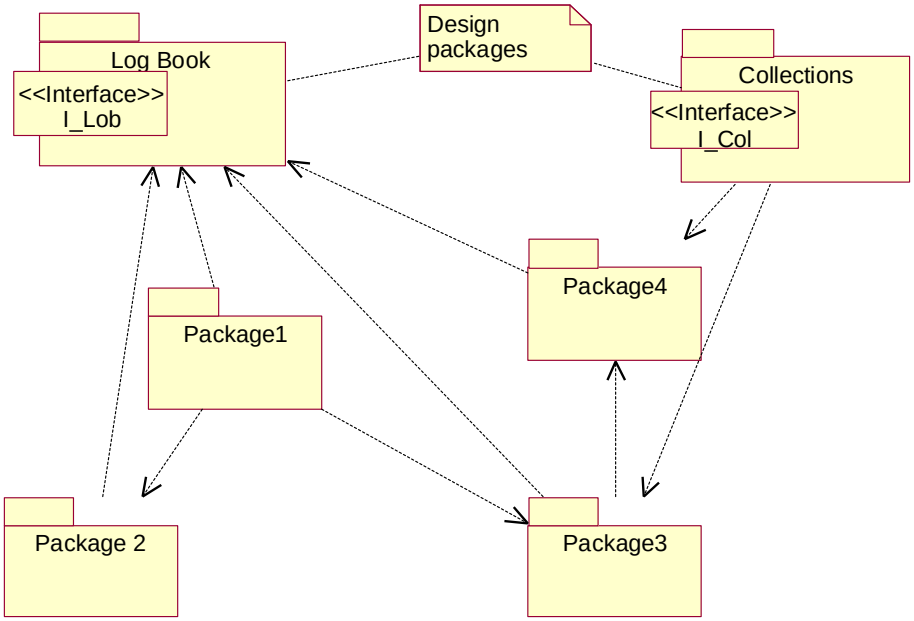
Les règles liées à l'utilisation de composants de conception : création de nouveaux composants ou réutilisation de composants existants (littérature, catalogues,…) sont exposées dans le chapitre utilisation de composants de conception ; en voici un rapide récapitulatif :

Règles pour la réutilisation d'un composant de conception :
Le composant réutilisé doit s'appliquer dans un contexte général ; il est potentiellement réutilisable sur tout projet relevant du métier pour les solutions métier et sur tout type de projet pour les solutions standards. Les composants doivent proposer une forme de généricité.
Le composant réutilisé doit être issu d'un projet opérationnel sur lequel il s'est appliqué avec des résultats satisfaisants. Les composants doivent avoir prouvé leur efficacité.
La formalisation du modèle doit apporter une aide importante aux futurs concepteurs, en termes de compréhension et d'utilisation. Les composants réutilisés doivent être documentés pour pouvoir être utilisés de façon efficace.

Règles pour la création d'un nouveau composant de conception :
La documentation d'un composant réutilisable doit suivre un plan standard qui garantit l'homogénéité, la cohérence et la lisibilité de la documentation.
La documentation doit comprendre au moins un exemple de mise en œuvre.
La définition d'un composant réutilisable doit se faire au début de sa conception ou au plus tard avant la fin de la phase.
En fin de projet la description du composant réutilisable doit être mise à jour afin de prendre en compte les évolutions dues à la conception détaillée et au codage.
La description du composant réutilisable doit être écrite par le concepteur l'ayant mis en œuvre.

9.3.7 Exemple

Figure 24 - Diagrammes de paquetage (composants de logiciels) complétés.



9.4 (H)-APPLICATION DE MODELES DE CONCEPTION

9.4.1 Objectifs

Il s'agit de réutiliser les modèles d'architecture existants (métier ou standard) pour réaliser des gains de temps et améliorer la qualité du produit (par la réutilisation de solutions éprouvées).

9.4.2 Intervenants

La réutilisation de ces architectures est effectuée en commun par l'équipe de conception.

9.4.3 Entrées

Sortie des activités E,F et G.

9.4.4 Sorties

Diagrammes complétés

9.4.5 Description

9.4.5.1 Introduction

De la même façon que pour l'activité précédente, l'application de modèles de conception est fortement lié à l'idée de réutilisation, cette activité est mise en œuvre dés qu'il y a identification d'un modèle de conception qu'il soit nouveau ou existant il obéit à des règles similaires, elles aussi décrites au chapitre « Utilisation de composants de conception ». Etant donné l'aspect particulier des Patterns, il est préférable d'identifier au plus tôt ces architectures afin de réduire le coût pour le projet de la mise en place ou de l'utilisation de ces solutions.

9.4.5.2 Comment réaliser les sorties

L'identification d'une architecture doit se faire au plus tôt ; souvent, lors de l'analyse, on a l'intuition qu'une architecture particulière va être utilisée ; cependant elle ne doit pas être mise en œuvre lors de l'analyse qui doit être exempte de toute solution technique. Utiliser ces schémas garantit un gain de temps et une qualité améliorée. En effet, les solutions proposées sont éprouvées et ont montré leur efficacité sur des projets opérationnels. On peut citer en exemple le cas d'une architecture trois ou n tiers.
C'est donc en phase de conception que ces architectures doivent être identifiées ou réalisées. Pour cela il faut puiser dans des catalogues ou dans la littérature l'architecture qui convient ou bien identifier une nouvelle architecture qui doit être décrite et formalisée de façon à être réutilisable. Il existe des documents CNES (HIBOU, EDO, ...) ainsi que la littérature reconnue (GAMMA, BOOCH, ...) décrivant des architectures de conception.
Comme pour les composants de conception, ces solutions sont soit dédiées au métier spécifique à l'application, soit génériques.

Remarque : ces solutions sont indépendantes de la méthode de conception utilisée.

9.4.6 Règles de vérification

Les règles liées à l'utilisation d'architectures de conception : création de nouvelles architectures ou réutilisation d'architectures existantes (littérature, catalogues, …) sont les mêmes que pour les composants (qui sont en fait des architectures instanciées).

9.4.6.1 Conseils sur l'utilisation et la création de composants.

Ces règles sont un récapitulatif des règle+s exposées au chapitre « Utilisation de composants de conception » .

Règles pour la réutilisation d'architectures de conception :
L'architecture réutilisée doit s'appliquer dans un contexte général ; elle est potentiellement réutilisable sur tout projet relevant du métier pour les solutions métier et sur tout type de projet pour les solutions standards.
L'architecture réutilisée doit être issue d'un projet opérationnel sur lequel elle s'est appliqué avec des résultats satisfaisants.
La formalisation du modèle doit apporter une aide importante aux futurs concepteurs, en termes de compréhension et d'utilisation. Les architectures réutilisées doivent être documentées pour pouvoir être utilisées de façon efficace.

Règles pour la création d'une nouvelle architecture de conception :
La documentation d'une architecture réutilisable doit suivre un plan standard qui garantit l'homogénéité, la cohérence et la lisibilité de la documentation.
La documentation doit comprendre au moins un exemple de mise en œuvre.
La définition d'une architecture réutilisable doit se faire au début de sa conception ou au plus tard avant la fin de la phase.
En fin de projet la description de l'architecture réutilisable doit être mise à jour afin de prendre en compte les évolutions dues à la conception détaillée et au codage.
La description de l'architecture réutilisable devrait être écrite par le concepteur l'ayant mis en œuvre.

9.4.7 Exemples

Figure 25 – Exemple d'un design pattern métier

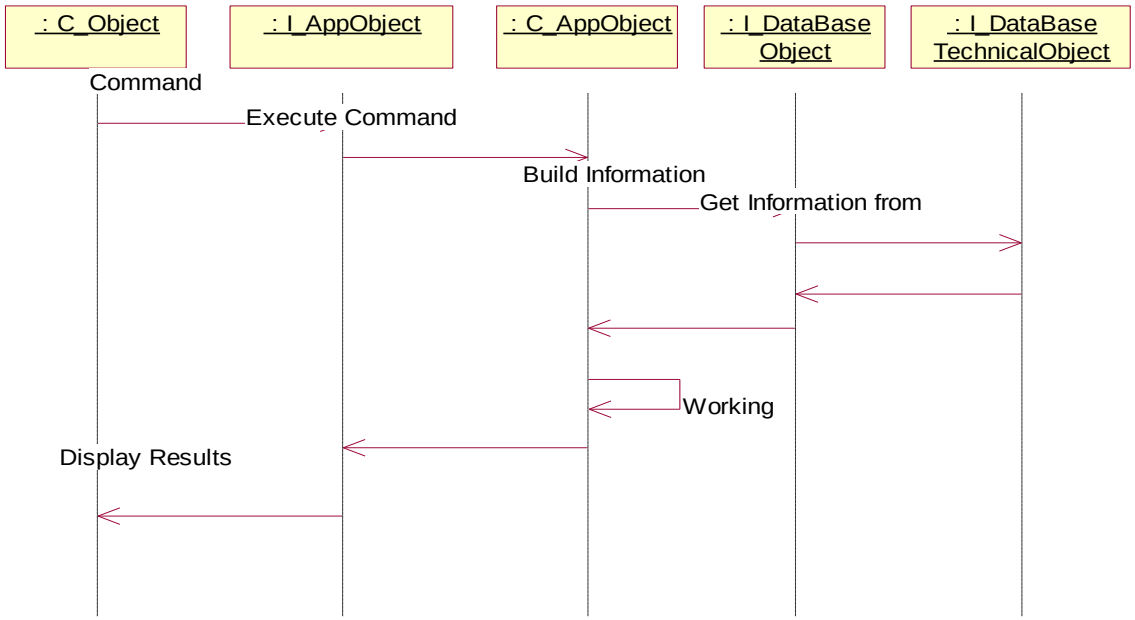
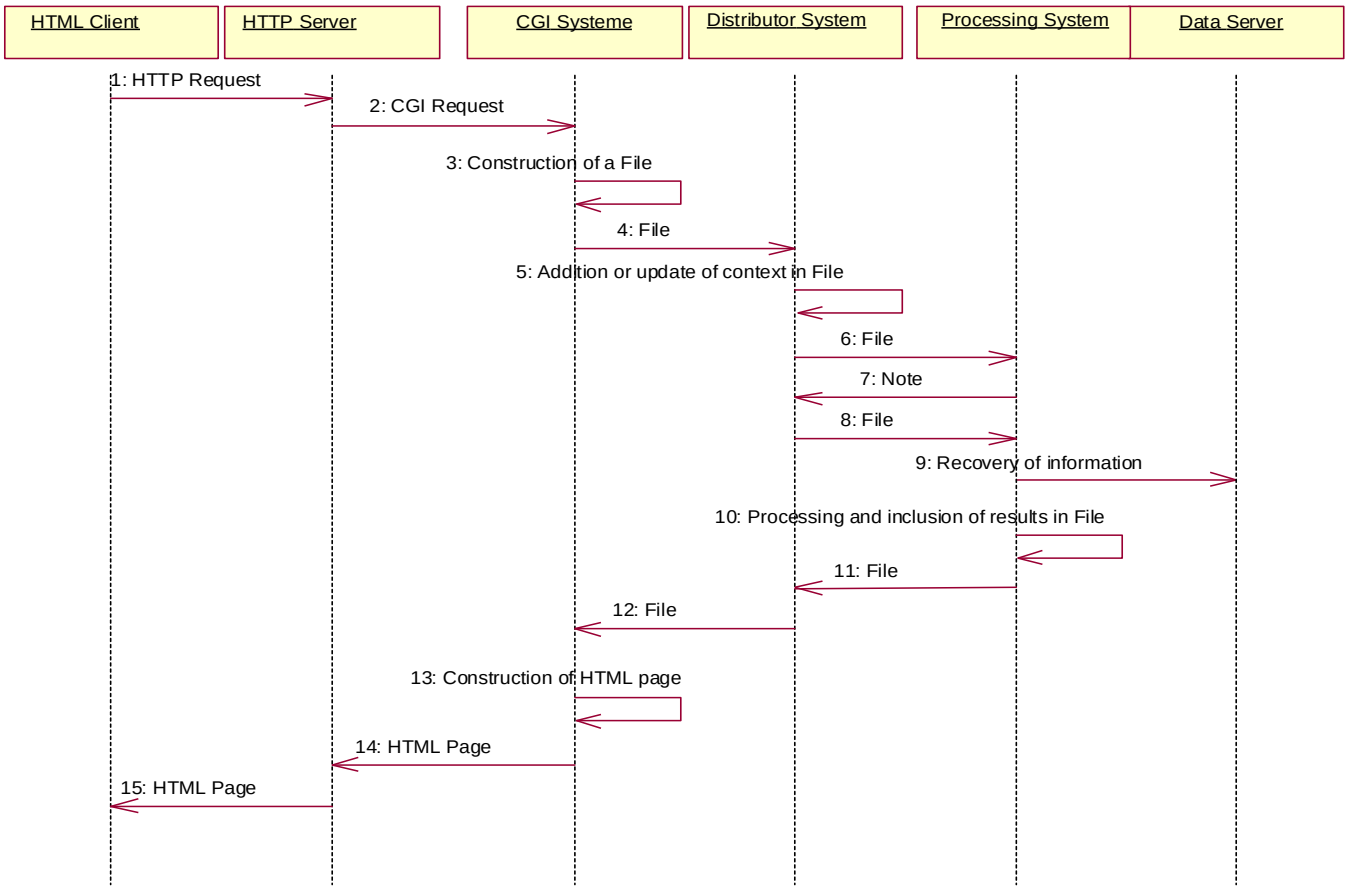


Figure 26 – Un autre exemple de design pattern métier



9.5 FIN DE LA PHASE DE CONCEPTION DE L'ARCHITECTURE

La phase de conception proprement dite est commencée tout du long de cette phase nous avons énoncés des règles et des conseils à mettre en application pour réaliser bonne architecture. Une fois cette étape réalisée et afin de la clôturer il est conseillé de vérifier que ces règles et conseils ont été appliques. Une fois cette vérification terminée on vas détailler, raffiner les classes de cette architecture mise en place (la solution technique choisie) .

9.6 MODELES DE DOCUMENTS D'ANALYSE CONCEPTION

Ce chapitre contient les modèles de documents à produire en fin de phase de conception de l'architecture. Pour ce qui est de la production de documentations spécifiques comme un plan de validation ou un manuel d'interface les éléments documentaires ils serons détaillés au chapitre « DOCUMENTATION ».

Document d'analyse conception : on peut désormais ajouter au plan du document les points suivants :

Concernant l'architecture générale du système on détaille :
la hiérarchie des paquetages, ainsi que les modèles de conception appliqués,
les interactions entre paquetages de l'activité E : Diagrammes de packages, diagrammes de séquence ou de collaboration entre packages,
l'identification des processus activité β : Diagrammes de composants (l'activité β est réalisée en parallèle de l'activité D et décrite plus bas),
Concernant architecture détaillée du système on détaille :
la hiérarchie des paquetages, ainsi que les modèles de conception appliquée,
la description de chaque paquetage des activités E-F-G-H. Pour chaque paquetage, on inclura la liste des classes contenues, les diagrammes d'interactions (séquence, collaboration), les diagrammes de classes.

Annexe 1 - Dictionnaire des classes. Le dictionnaire pourra être organisé selon la méthode la plus appropriée permettant un accès rapide à la description d'une classe.

Soit le plan global suivant :

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Glossaire	A-B-C	Glossaire technique du projet
Documents applicables	-	-
Documents de réference	-	-
Introduction	-	-
Objectifs du document	Démarrage	Description des points couverts par le document. Identification du niveau de détail de la conception.
Organisation du document	Démarrage	Description de la structure principale du document.
Présentation du système	-	-
Objectifs du système	A-B-C	Définition des principales exigences fonctionnelles.
Contexte du système	B	Description des interactions entre les entités externes et le système.
Définition des acteurs	A	Définition de chaque acteur, hiérarchies d'acteurs éventuellement construites, diagrammes d'interaction les concernant.
Analyse du domaine	C	Identification des interactions entre acteurs et cas d'utilisation. Identification des objets métier.
Modèle par cas d'utilisation	B-C	La description des cas d'utilisation se base sur le modèle fourni en annexe. Elle pourra être organisée par package. Pour chaque cas d'utilisation, on incluera les diagrammes utilisés : diagrammes de séquence, diagrammes de collaboration, diagrammes partiels de classes.

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Architecture physique	α	Identification des principaux éléments physiques du système (diagrammes de déploiement).
Architecture générale du système	-	On détaillera la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.
Description des packages	D	Rôles et contenu principal de chaque package.
Interactions entre packages	**E**	**Diagrammes de packages, diagrammes de séquence ou de collaboration entre packages.**
Identification des processus	β	Diagrammes de composants.
Architecture détaillée du système	**-**	**On détaille la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.**
Description du paquetage XXX	**E-F-G-H**	**Pour chaque paquetage XXX, on inclue la liste des classes contenues, les diagrammes d'interactions (séquence, collaboration), les diagrammes de classes.**
Allocation des classes	**γ**	**Pour chaque processus, on donne la liste des paquetages et/ou classes prises en charge.**
Traçabilité	-	-
Principes de traçabilité	Démarrage	Identification des matrices de traçabilité applicables (décrites en annexe).
Annexe 1 - Dictionnaire des classes	**-**	**Le dictionnaire pourra être organisé selon la méthode la plus appropriée permettant un accès rapide à la description d'une classe.**
Annexe 2 - Traçabilité	-	-
Exigences / cas d'utilisation	Avant D	Matrice de traçabilité entre les exigences utilisateur et les éléments d'analyse identifiés lors des activités A, B et C.
Cas d'utilisation / analyse objet	**Avant I**	**Matrice de traçabilité entre les éléments d'analyse identifiés lors des activités A, B et C, et les éléments de conception identifiés lors des activités D à H.**

9.7 UTILISATION DES COMPOSANTS DE CONCEPTION

L'utilisation de composants et leur formalisation est fortement liée à cette phase qu'est la conception de l'architecture ; les règles liées à leur utilisation sont décrites de ce fait dans les activités qui la compose. On peut toute fois se référer au chapitre «UTILISATION DES COMPOSANTS DANS LE PROCESSUS » pour avoir une vision plus globale des cas d'utilisation ou de production de composants.

10 CONCEPTION OBJET

Une fois l'architecture validée, elle va être détaillée par la phase de conception-objet composée de trois activités identiques dans leur objectif mais portant sur des éléments différentiels : les classes interface homme machine K, les classes de base de données L et les autres classes du modèle I.
Etant données les similitudes entres ces activités, nous détaillons ci-dessous le démarche à suivre pour ces activités.

10.1 DEMARCHE DE BASE UTILISE POUR CHAQUE PAQUETAGE.

10.1.1 Définition du problème

Durant l'activité I, on définit le problème du point de vue du concepteur du paquetage.

10.1.2 Description de la solution

Durant les activités I, J et K
Réalisation des diagrammes de séquence de classes… à partir des services offerts par la ou les classe(s) d'interface.

10.1.3 Justification de la solution

En fin de phase, si nécessaire, justification de la décomposition du paquetage.

10.1.4 Validation de la solution

En fin de phase, par des cycles auteur/lecteur,
pour un seul paquetage (revue de décomposition)
pour n paquetage (revue de niveau)

10.1.5 Formalisation de la solution

En fin de phase, par génération de code pour valider les interfaces
C++
Java
Ada
…

10.1.6 Exemples d'architectures

10.1.6.1 Glossaire

- L'Application (APP)

L'APP représente le logiciel à réaliser (hors MMI, BD et Interfaces), elle est constituée par toute la partie applicative (algorithmes, calculs ...).

- L'interface application (APPI)

Cette interface est constituée par un ensemble d'objets. Ces derniers représentent les données qui sont récupérées par un client (qui est généralement mais pas toujours un MMI) afin de les présenter à la visualisation ou à la modification.

- L'Interface Base de Données (DBI)

L'accès à la base de données est effectué de deux manières différentes:

- via les objets "interface d'accès aux données: ADBDI " (voir ci-après),
- ou directement à partir des écrans (ex: générés par FORMS). A partir de ce type d'écran nous pouvons activer: un autre écran, une opération fourni par un objet "ADBDI" ou un "TRIGGER".

- L'interface DBI se décompose comme suit:

L'interface d'accès aux données (ADDBI)
Cette interface est constituée par un ensemble d'objets. Ces derniers représentent les données (construites à partir du contenu - n tables - de la base) qui sont manipulées par l'application et (ou) l'IHM.
Remarque: ADDBI représente la vue de la base de données dont a besoin l'application, c'est à dire les données dynamiques construites à partir des données persistantes (ex: les tables). Il faut cependant noter que dans certains cas il peut y avoir bijection entre certains objets ADDBI et des tables de la base de données.

- L'interface d'accès à la base de données (ABDBI)

Cette interface permet de créer, détruire et exécuter des commandes pour lire, créer, modifier ou détruire des attributs dans la base de données. C'est en fait une interface générique pour exécuter du SQL.

- L'interface système (SDBI)

Cette interface est unique, elle permet de réaliser les connexions déconnexions à la base, la gestion commune des erreurs et les transactions ("COMMIT" et "ROLLBACK").

- L'Interface Homme Machine dite "Event Driven" (MMI).

Remarque: Les applications les plus couramment rencontrées aujourd'hui sont de ce type. La MMI est maître de l'enchaînement. Elle est en attente permanente de requêtes opérateur (saisie, sélection) et obéit à ces requêtes pour exécuter des traitements fonctionnels. Elle se compose des éléments suivants:

- La MMI Statique (SMMI):

Code de création des écrans. Il peut être développé grâce à la manipulation de librairies graphiques comme Motif, ou généré automatiquement par un générateur d'interfaces.

- La MMI Dynamique (DMMI):

Code permettant de gérer la dynamique de la MMI.

10.1.6.2 Exemples

Figure 27 – Types d'objets de collaboration possibles

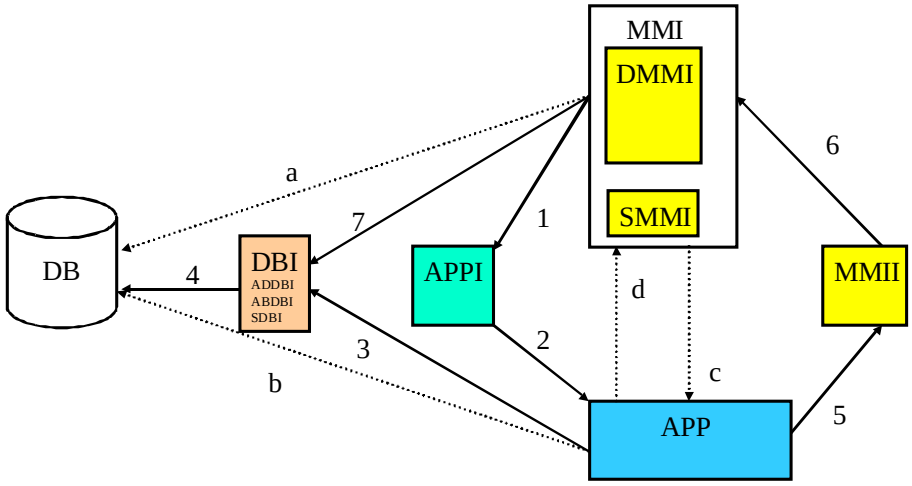


Figure 28 – un exemple courant d'architecture

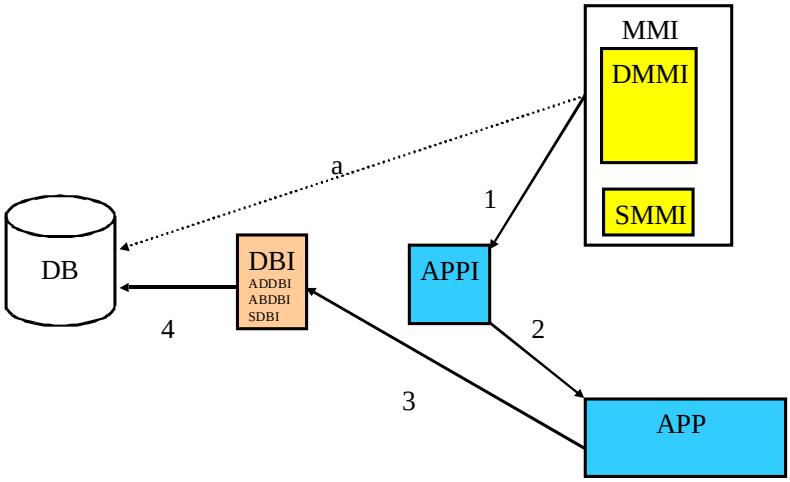
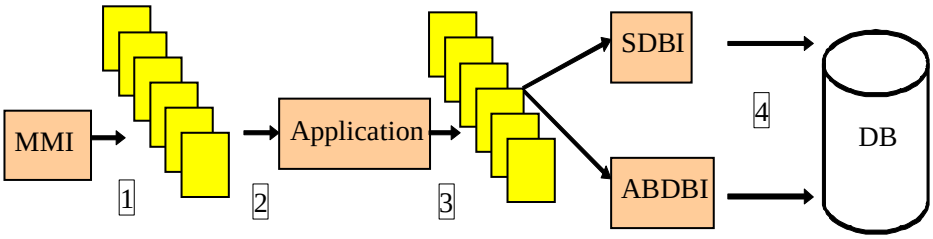


Figure 29 – Un exemple d'architecture idéale

Application Interface Database Access Objects

APPI ADDBI



I Conception des classes

10.2 (I)–CONCEPTION DES CLASSES

10.2.1 Objectifs

Il s'agit de compléter les éléments du modèle obtenus précédemment par des ajouts d'attributs, de méthodes, de diagrammes d'interaction et de paquetages et classes de conception.

10.2.2 Intervenants

La conception des paquetages (composant logiciel) est repartie entre les différents membres de l'équipe de conception.

10.2.3 Entrées

Sorties des activités précédentes.

10.2.4 Sorties

Description complète des classes.
Diagrammes internes aux paquetages (classes séquences …)
Diagrammes d'interaction entre paquetages modifiés (si nécessaire)

10.2.5 Description

10.2.5.1 Introduction

Cette activité est analogue à la l'activité F ; elle conduit au raffinage des classes et leurs interactions par l'édition de diagrammes de classes leur description textuelle complète ainsi que les diagrammes de paquetage montrant leurs interactions par le biais des interfaces. Elle obéit aux mêmes règles que l'activité F c'est à dire aux règles spécifiques du projet comme les règles de nommage.

10.2.5.2 Comment réaliser les sorties

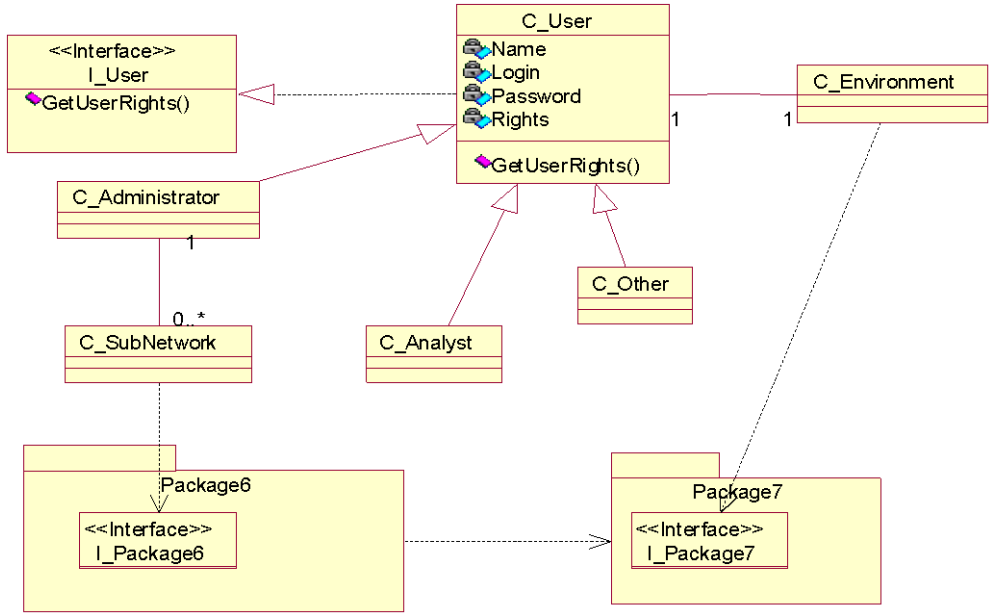
Les diagrammes de cette activité existent déjà ; ils ont été créés lors de la conception de l'architecture ; ils doivent êtres complétés. S'ils n'existent pas encore ils devront êtres créés. Pour réaliser cette activité il faut appliquer la démarche décrite en début de chapitre.

10.2.6 Règles de vérification

Il n'y a pas de règles méthodologiques à appliquer à ce niveau de la conception

10.2.7 Exemples

Figure 30 – Exemple de décomposition d'un paquetage



10.3 (J)–CONCEPTION DES CLASSES IHM

10.3.1 Objectifs

Il s'agit d'identifier les classes IHM.

10.3.2 Intervenants

Les membres de l'équipe concernés par le paquetage.

10.3.3 Entrées

Sortie des activités précédentes.

10.3.4 Sorties

Diagramme de classes complété.
Diagrammes de paquetages complétés.
Diagrammes d'interaction (classes et paquetages) complétés.

10.3.5 Description

10.3.5.1 Introduction

Cette activité est identique à la l'activité I ; elle conduit au raffinage des classes d'interface homme machine et de leurs interactions par l'édition de diagrammes de classes ; leur description textuelle complète ainsi que les diagrammes de paquetage montrant leurs interactions par le biais des interfaces de paquetage. Elle obéit aux mêmes règles que l'activité I c'est à dire aux règles spécifiques du projet comme les règles de nommage.

10.3.5.2 Comment réaliser les sorties

Les diagrammes de cette activité existent déjà ils ont été créés lors de la conception de l'architecture ils doivent êtres complétés. S'ils n'existent pas encore ils devront êtres créés. Pour réaliser cette activité il faut appliquer la démarche décrite en début de chapitre.

10.3.6 Règles de vérification

Il n'y a pas de règles méthodologiques à appliquer à ce niveau de la conception

10.3.7 Exemples

Figure 31 – Exemple de diagramme de séquence avec l'IHM

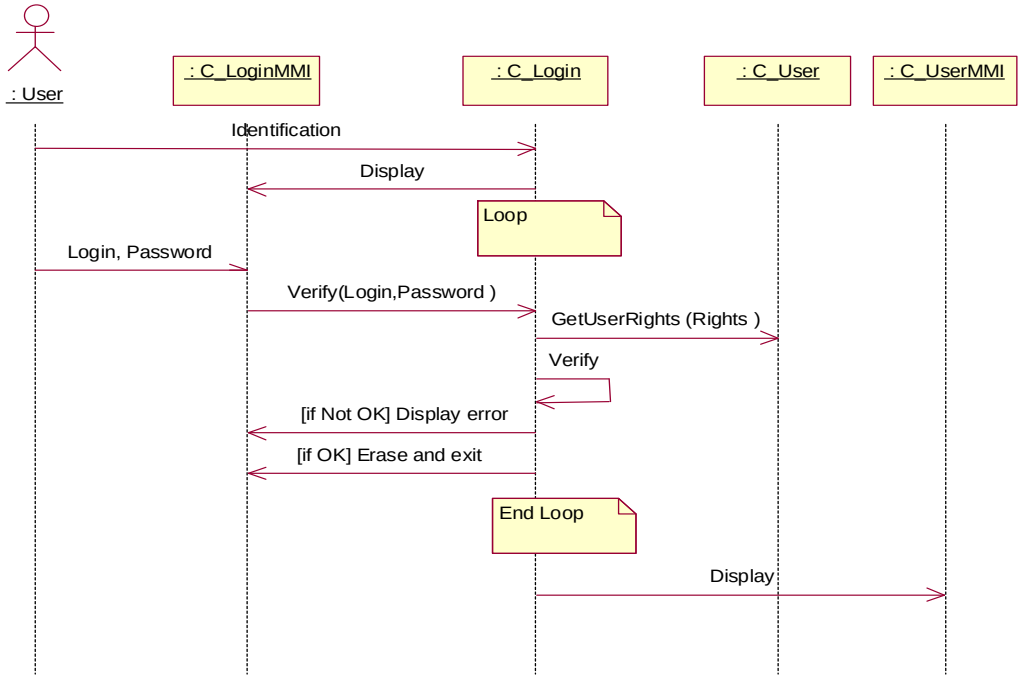
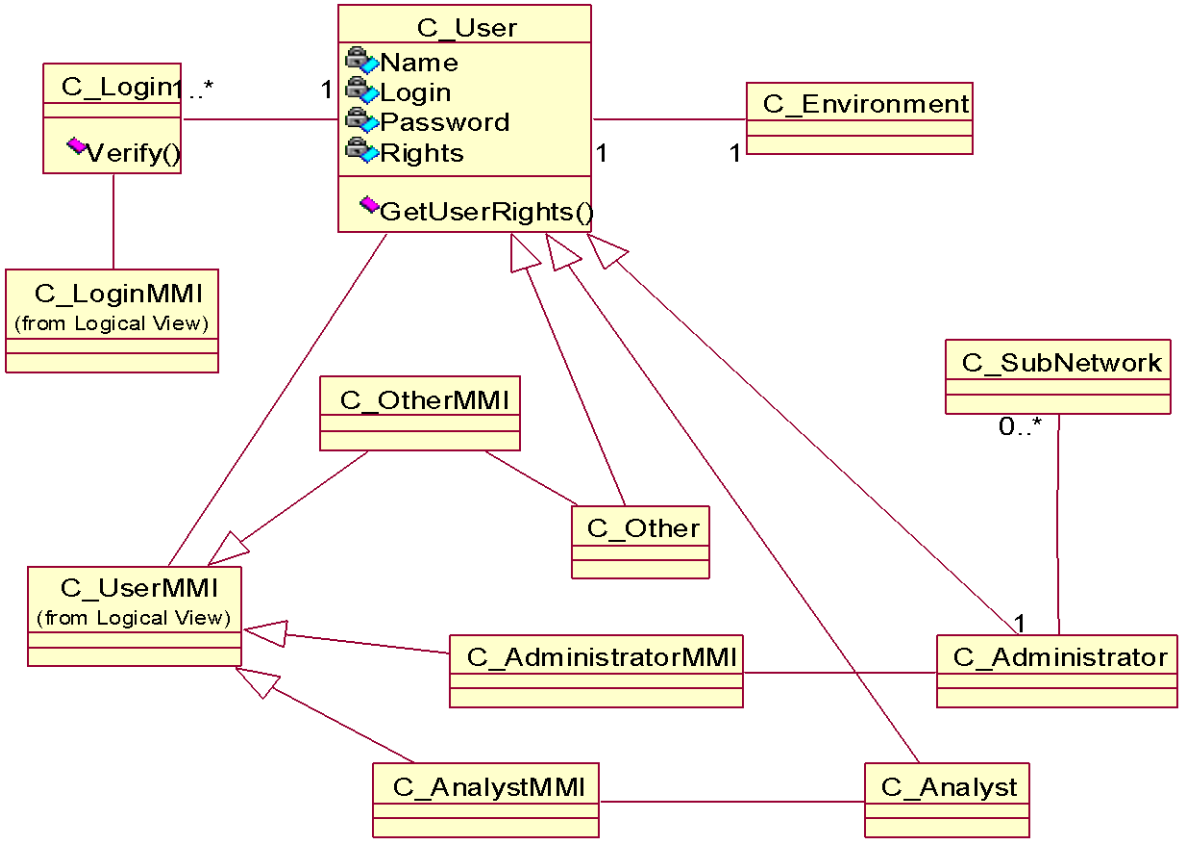


Figure 32 - Exemple de diagramme de classes avec l'IHM



10.4 (K)-CONCEPTION DES CLASSES BD

K Conception des classes BD

10.4.1 Objectif

Il s'agit d'identifier les classes persistantes qui accèdent à la base de données.

10.4.2 Intervenants

Les membres de l'équipe concernés par le paquetage.

10.4.3 Entrées

Sortie des activités précédentes.

10.4.4 Sorties

Diagramme de classes complété.
Diagrammes de paquetages complétés.
Diagrammes d'interaction (classes et paquetages) complétés.

10.4.5 Description

10.4.5.1 Introduction

Cette activité est identique aux activité I et J ; elle conduit au raffinage des classes de base de données et leurs interactions par l'édition de diagrammes de classes ; leur description textuelle complète ainsi que les diagrammes de paquetage montrant leurs interactions par le biais des interfaces de paquetage. Elle obéit aux mêmes règles que celles des activités I et J c'est à dire aux règles spécifiques du projet comme les règles de nommage.

10.4.5.2 Comment réaliser les sorties

Les diagrammes de cette activité existent déjà ; ils ont été créés lors de la conception de l'architecture ; ils doivent êtres complétés ; s'ils n'existent pas encore ils devront êtres créés. Pour réaliser cette activité il faut appliquer la démarche décrite en début de chapitre.

10.4.6 Règles de vérification

Il n'y a pas de règles méthodologiques à appliquer à ce niveau de la conception

10.5 FIN DE LA PHASE DE CONCEPTION OBJET

La conception objet, dont l'objectif principal est la conception détaillée des paquetages et des classes du système est terminée. Elle a conduit au raffinage des diagrammes crées précédemment en fonction du niveau de détail à apporter à la conception soit la mise à jour du contenu des diagrammes existants, la création des diagrammes décrivant les classes "interface homme machine" et la création des diagrammes décrivant les classes "interface base de données". Nous n'avons pas mis en évidence lors de ces phases de règles précises les règles à respecter et à vérifier lors de ces activités sont pour la plus part des règles liées au projet (langage de programmation utilisé ex : règles de codage JAVA, C, C++, … ou règles de nomages spécifiques, …)

10.6 MODELES DE DOCUMENTS DE CONCEPTION

A la fin de cette étape on vas incrémenter la documentation d'analyse conception avec de nouveaux éléments soit :

Soit le plan global suivant :

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Glossaire	A-B-C	Glossaire technique du projet
Documents applicables	-	-
Documents de réference	-	-
Introduction	-	-
Objectifs du document	Démarrage	Description des points couverts par le document. Identification du niveau de détail de la conception.
Organisation du document	Démarrage	Description de la structure principale du document.
Présentation du système	-	-
Objectifs du système	A-B-C	Définition des principales exigences fonctionnelles.
Contexte du système	B	Description des interactions entre les entités externes et le système.
Définition des acteurs	A	Définition de chaque acteur, hiérarchies d'acteurs éventuellement construites, diagrammes d'interaction les concernant.
Analyse du domaine	C	Identification des interactions entre acteurs et cas d'utilisation. Identification des objets métier.
Modèle par cas d'utilisation	B-C	La description des cas d'utilisation se basera sur le modèle fourni en annexe. Elle pourra être organisée par package. Pour chaque cas d'utilisation, on incluera les diagrammes utilisés : diagrammes de séquence, diagrammes de collaboration, diagrammes partiels de classes.
Architecture physique	α	Identification des principaux éléments physiques du système (diagrammes de déploiement).
Architecture générale du système	-	On détaillera la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.
Description des packages	D	Rôles et contenu principal de chaque package.
Interactions entre packages	E	Diagrammes de packages, diagrammes de séquence ou de collaboration entre packages.
Identification des processus	β	Diagrammes de composants.
Architecture détaillée du système	-	On détaillera la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.
Description du paquetage XXX	E-F-G-H	Pour chaque paquetage XXX, on inclue la liste des classes contenues, les diagrammes d'interactions (séquence, collaboration), les diagrammes de classes.
Allocation des classes	γ	Pour chaque processus, on donnera la liste des paquetages et/ou classes prises en charge.

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Traçabilité	-	-
Principes de traçabilité	Démarrage	Identification des matrices de traçabilité applicables (décrites en annexe).
Annexe 1 - Dictionnaire des classes	-	Le dictionnaire pourra être organisé selon la méthode la plus appropriée permettant un accès rapide à la description d'une classe.
Classe XXX	**I-J-K**	**Pour chaque classe, on décrira ses attributs, son comportement (méthodes, diagrammes d'états-transitions). Le niveau de détail est fonction du projet.**
Annexe 2 - Traçabilité	-	-
Exigences / cas d'utilisation	Avant D	Matrice de traçabilité entre les exigences utilisateur et les éléments d'analyse identifiés lors des activités A, B et C.
Cas d'utilisation / analyse objet	Avant I	Matrice de traçabilité entre les éléments d'analyse identifiés lors des activités A, B et C, et les éléments de conception identifiés lors des activités D à H.
Analyse objet / Conception des classes	Avant γ	Matrice de traçabilité entre éléments de conception identifiés lors des activités D à H, et les classes identifiées lors des activités I, J et K.

10.7 UTILISATION DES COMPOSANTS DE CONCEPTION

De la même façon que precedement on vas parfois être amené dans cette phase a utiliser des composants lors de cette phase pour cela il faudra se référer une fois de plus au chapitre «UTILISATION DES COMPOSANTS DANS LE PROCESSUS » pour avoir une vision plus globale des cas d'utilisation ou de production de composants.

11 CONCEPTION PHYSIQUE

11.1.1 Objectifs

La conception physique d'un système répond à la question « Où ? ». Elle utilise les diagrammes de déploiement et de composants.
Note : il s'agit d'un concept similaire à celui des nœuds virtuels HOOD [7].

11.1.2 Activités

La conception physique peut commencer à n'importe quel niveau du processus. Elle comporte habituellement trois activités :
Description de l'architecture physique (α)
Identification des processus et composants (β)
Allocation des classes (γ)

α Description de l'architecture physique

11.2 (α)-DESCRIPTION DE L'ARCHITECTURE PHYSIQUE

11.2.1 Objectifs

Définir et organiser les éléments matériels (calculateurs, périphériques, réseau…) qui composent le système. L'architecture physique est généralement élaborée parallèlement à l'analyse des cas d'utilisation.

11.2.2 Intervenants

Architecte système,
Ingénieur système,
Ingénieur réseau.

11.2.3 Entrées

Exigences relatives au système,
Contexte physique.

11.2.4 Sorties

Diagramme de déploiement des composants matériels

11.2.5 Description

11.2.5.1 Introduction

Cette activité est menée en début de la phase d'analyse en parallèle de la détermination des acteurs ;« le où » est mis en place en même temps que « le qui » ce qui permet d'élucider dés le début du processus les contraintes liées à l'architecture physique et donc de ne pas se retrouver avec un modèle incompatible avec l'architecture physique choisie.

11.2.5.2 Comment réaliser les sorties

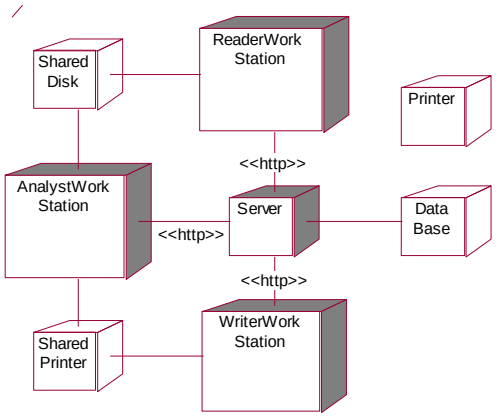
Cette sortie (le diagramme de déploiement des composants matériel) doit se faire par les spécialistes des domaines système et réseau ; ils doivent obéir aux contraintes que sont les architecture matérielles existantes et les exigences des demandeurs du logiciel.

11.2.6 Règles de vérification

Tenir compte de l'environnement dans lequel doit venir s'insérer le système et de son évolution probable. Accorder une attention toute particulière aux bandes passantes disponibles sur les réseaux mis à contribution.

11.2.7 Exemples

Figure 33 - Exemple d'un diagramme de déploiement



11.3 (β)-IDENTIFICATION DES PROCESSUS ET COMPOSANTS

β Identification des processus et composants

11.3.1 Objectifs

Compléter l'activité précédente en précisant la distribution des processus. L'identification des processus peut commencer en parallèle de l'analyse objet (activité D).
Important : l'étude des données échangées entre les différents processus aide l'analyste à identifier les classes d'interface du modèle logique.

11.3.2 Intervenants

Architecte système,
Analyste.

11.3.3 Entrées

Exigences relatives au système,
Diagramme(s) de déploiement des composants matériels.

11.3.4 Sorties

Diagramme(s) de déploiement des processus.

11.3.5 Description

11.3.5.1 Introduction

Cette activité est synchronisée avec l'activité D ce qui permet une fois « le quoi » établi de déterminer les processus qui vont être implémentés et donc de les localiser sur l'architecture physique.

11.3.5.2 Comment réaliser les sorties

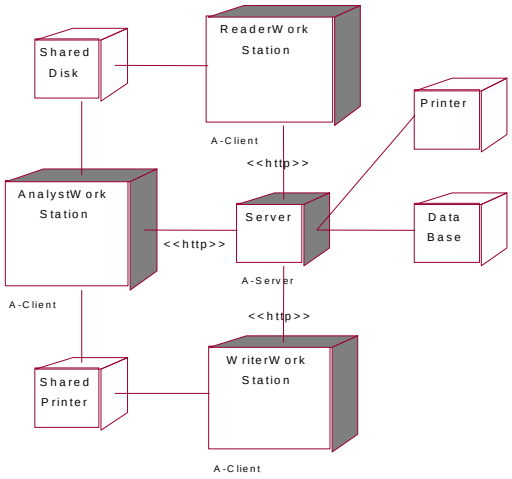
Le développement d'un système multi-processus et/ou multi-tâches nécessite l'implémentation d'une synchronisation qui constitue une charge de travail importante et une source de fragilité. Il convient donc de toujours limiter au minimum nécessaire le nombre de tâches et de processus.

11.3.6 Règles de vérification

Pour définir l'implantation des différents processus on observe les mêmes règles que pour l'activité précédente (α).

11.3.7 Exemples

Figure 34 : Exemple de diagramme de déploiement avec processus (A-Client, A-Server...)



γ Allocation des classes

11.4 (γ)-ALLOCATION DES CLASSES

11.4.1 Objectifs

Affecter les classes aux composants logiciels et aux processus identifiés lors de l'activité β.
On construit tout d'abord les processus (client, serveur…) par assemblage de composants logiques (IHM, timer, contrôleur…). Les classes sont ensuite affectées à ces composants logiques (cf. figure 35).
Cette activité est généralement menée en parallèle avec la conception logique (activités I, J et K).

11.4.2 Intervenants

Architecte système.

11.4.3 Entrées

Diagramme(s) de déploiement des processus,
Diagramme(s) de classes issu(s) de la conception.

11.4.4 Sorties

Diagramme(s) de composants.

11.4.5 Description

11.4.5.1 Introduction

Cette activité est le point de synchronisation majeur entre l'architecture logique et l'architecture physique elle permet de localiser l'endroit où vont se réaliser les processus en visualisant où se trouvent les classes qui y participent ; elle va permettre entre autres d'observer la charge pour l'architecture physique et de l'harmoniser.

11.4.5.2 Comment réaliser les sorties

Pour faciliter le travail en groupe il est souhaitable de limiter au maximum le nombre de classes par fichier (1 seule si possible).

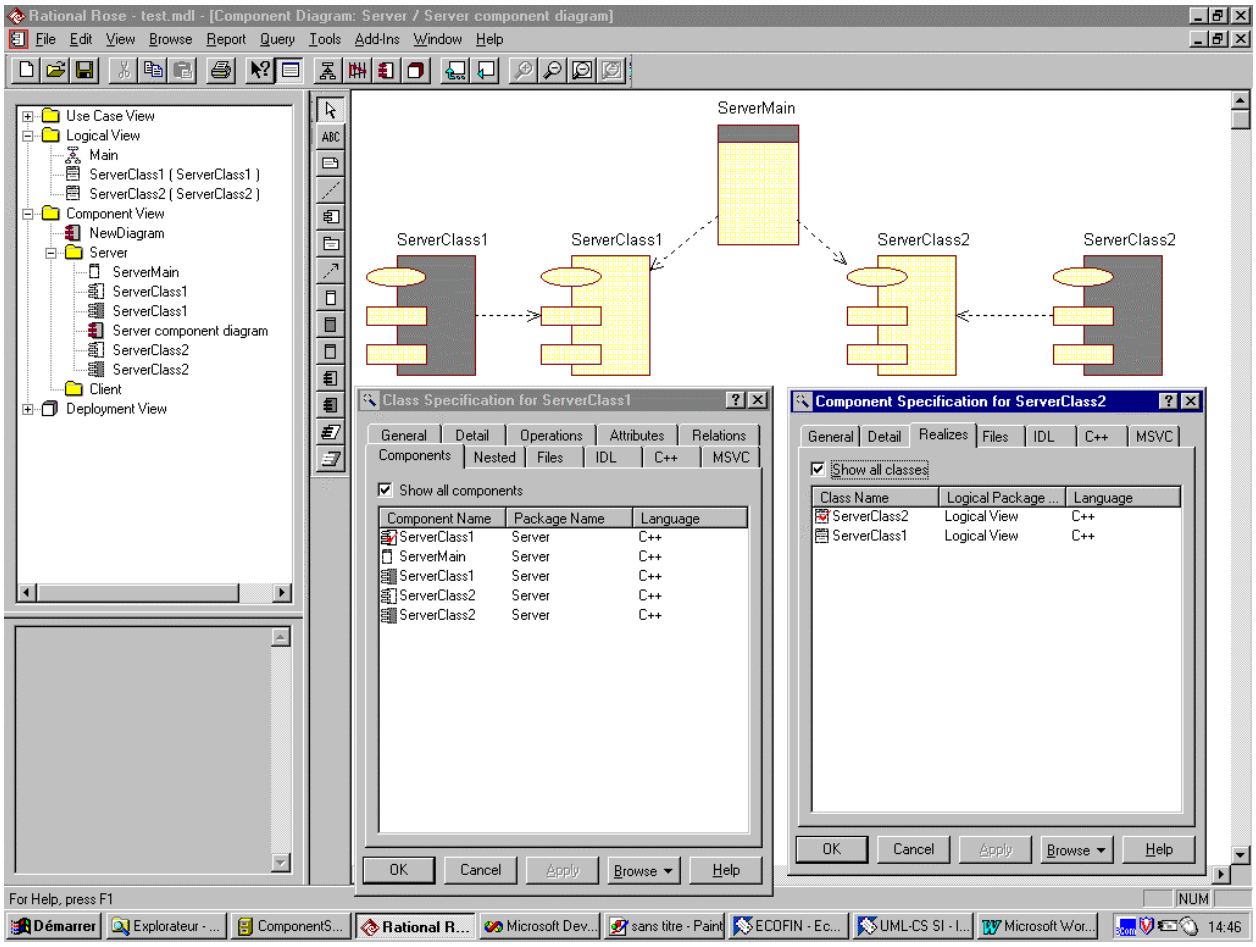
11.4.6 Règles de vérification

Il n'y a pas de règles spécifiques pour cette activité ; on peut cependant proposer une série de recommandations :

En C et C++, une partie du code peut être implémentée sous forme de bibliothèques statiques (.LIB) ou dynamiques (DLL, sous Windows uniquement). Attention, là encore, à l'augmentation de la charge de travail. D'une façon générale on utilise une bibliothèque statique pour des classes partagées par plusieurs processus ou utilisées par d'autres applications. Elle permet de diminuer le nombre de fichiers gérés par exécutable, accélère les compilations et prolonge la notion de « paquetage » réutilisable.
Sous Windows uniquement, l'utilisation d'un composant de type « DLL » (*Dynamic Linked Library*) permet de stocker du code à l'extérieur d'un fichier exécutable classique (.EXE) et de le partager. Les principaux avantages sont un gain d'espace et des possibilités de mise à jour indépendantes (ex. : gestion du multilingue d'une application). Coté inconvénients, elles sont d'une utilisation délicate et elles ne conviennent pas à tous les types de classes.
Lors de la génération de code C++ Rose ne reconnaît que deux catégories de composants : « Specification » et « Body ». Le première produira des ficher « .h » et le second des fichiers « .cpp », quel que soit le stéréotype (« Application », « Thread », « DLL » …) qui leur a été attribué. Rose ne fait donc pas de distinction entre les composants exécutables (EXE, DLL, LIB…) et les fichiers qui les composent.
Afin de d'obtenir une organisation facilement exploitable avec le compilateur, nous conseillons de recréer l'arborescence application/composants/fichiers dans la « Component view ». On peut par exemple créer un composant stéréotypé « Package specification » (fichier .h) et un composant stéréotypé « Package body » (fichier .cpp) par classe en les regroupant dans un ensemble de paquetages classiques. Chaque classe est ensuite affectée à son « Package specification ». Au moment de la génération du code, Rose reproduira la « Component view » sous forme de fichiers et sous-répertoires à partir de la racine fixée par la propriété C++ « Directory » de l'élément « Project ».
Il est possible de générer des directives « #include » en créant des dépendances entre composants (quels que soient leurs stéréotypes).

11.4.7 Exemples

Figure 35 - Exemple d'allocation de classe à des processus



11.5 FIN DE LA PHASE DE CONCEPTION PHYSIQUE

La conception physique se termine, au cours de cette phase l'architecte système structure le système en composants physiques, définit les processus exécutés, et met en correspondance ces processus avec les composants logiciels. Elle a conduit à la création d'un diagramme de déploiement des composants matériels, d'un diagramme de déploiement des processus (diagramme précédent complété) et de n diagramme(s) de composants, liant les classes aux composants logiciels et les composants logiciels (développés et/ou achetés) aux processus.
Il est important de noter que cette phase de conception physique se déroule en parallèle des autre phases ce qui permet de faire des rebouclages et de ce fait maintenir un forte cohésion entre la partie physique et la partie logique des applications développées.
Pour ce qui est de la validation de la phase les règles que l'on pourrait vouloir vérifier sont plus des règles liées au contraintes techniques (comme les bandes passante entre composants) que des règles méthodologiques.

11.6 MODELES DE DOCUMENTS D'ANALYSE CONCEPTION

Etant donné que la conception physique peut se dérouler en parallèle des autres activités les éléments documentaires à en extraire le sont aussi en parallèle ce chapitre vas donc se contenter de récapituler ces éléments:

Soit le plan global suivant :

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Glossaire	A-B-C	Glossaire technique du projet
Documents applicables	-	-
Documents de réference	-	-
Introduction	-	-
Objectifs du document	Démarrage	Description des points couverts par le document. Identification du niveau de détail de la conception.
Organisation du document	Démarrage	Description de la structure principale du document.
Présentation du système	-	-
Objectifs du système	A-B-C	Définition des principales exigences fonctionnelles.
Contexte du système	B	Description des interactions entre les entités externes et le système.
Définition des acteurs	A	Définition de chaque acteur, hiérarchies d'acteurs éventuellement construites, diagrammes d'interaction les concernant.
Analyse du domaine	C	Identification des interactions entre acteurs et cas d'utilisation. Identification des objets métier.
Modèle par cas d'utilisation	B-C	La description des cas d'utilisation se basera sur le modèle fourni en annexe. Elle pourra être organisée par package. Pour chaque cas d'utilisation, on incluera les diagrammes utilisés : diagrammes de séquence, diagrammes de collaboration, diagrammes partiels de classes.
Architecture physique	**α**	**Identification des principaux éléments physiques du système (diagrammes de déploiement).**
Architecture générale du système	-	On détaillera la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.
Description des packages	D	Rôles et contenu principal de chaque package.
Interactions entre packages	E	Diagrammes de packages, diagrammes de séquence ou de collaboration entre packages.
Identification des processus	**β**	**Diagrammes de composants.**
Architecture détaillée du système	-	On détaillera la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.
Description du paquetage XXX	E-F-G-H	Pour chaque paquetage XXX, on inclue la liste des classes contenues, les diagrammes d'interactions (séquence, collaboration), les diagrammes de classes.
Allocation des classes	**γ**	**Pour chaque processus, on donnera la liste des paquetages et/ou classes prises en charge.**
Traçabilité	-	-
Principes de traçabilité	Démarrage	Identification des matrices de traçabilité applicables (décrites en annexe).
Annexe 1 - Dictionnaire des classes	-	Le dictionnaire pourra être organisé selon la méthode la plus appropriée permettant un accès rapide à la description d'une classe.
Classe XXX	I-J-K	Pour chaque classe, on décrira ses attributs, son comportement (méthodes, diagrammes d'états-transitions). Le niveau de détail est fonction du projet.
Annexe 2 - Traçabilité	-	-
Exigences / cas d'utilisation	Avant D	Matrice de traçabilité entre les exigences utilisateur et les éléments d'analyse identifiés lors des activités A, B et C.

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Cas d'utilisation / analyse objet	Avant I	Matrice de traçabilité entre les éléments d'analyse identifiés lors des activités A, B et C, et les éléments de conception identifiés lors des activités D à H.
Analyse objet / Conception des classes	**Avant γ**	**Matrice de traçabilité entre éléments de conception identifiés lors des activités D à H, et les classes identifiées lors des activités I, J et K.**

11.7 UTILISATION DES COMPOSANTS

De la même façon que pour les autre phases on peut utiliser des composants lors de la conception physique, ces composants se présenterons probablement sous la forme de composants externes. Encore une fois il faudra se référer une fois de plus au chapitre «UTILISATION DES COMPOSANTS DANS LE PROCESSUS » .

12 NOTRE PROCESSUS ET LES AUTRES METHODES

Ce chapitre à pour but de situer notre processus sur la globalité du processus de création d'un logiciel, de mettre en évidence les particularités et les points communs entre notre processus et différentes méthodes de modélisation utilisant UML.

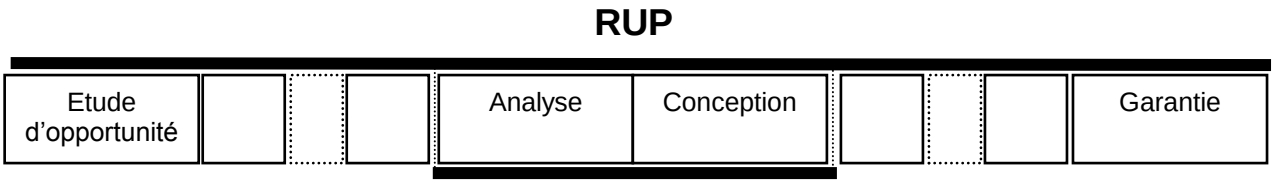
Les méthodes étudiées pour cette comparaison sont :

Unified Process de Rational
UML en action de Valtech
Modélisation avec UML de Pierre-Alain Muller

Comme pour ces trois méthodes notre processus est un processus piloté par les Use Case, un processus incrémental, on progresse par étapes incréments, un processus itératif, on procède par itérations successives.

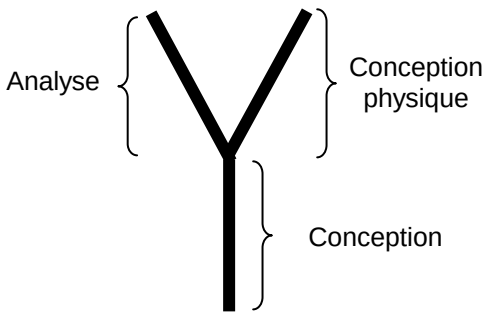
12.1 NOTRE PROCESSUS ET LE RUP (RATIONAL UNIFIED PROCESS)

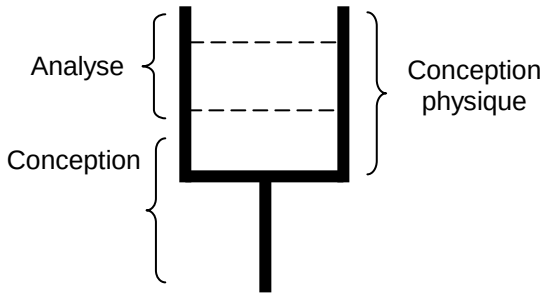
La principale différence entre le RUP et notre processus repose sur la portée de la méthode. Le RUP décrit la vie d'un projet logiciel de A à Z de l'étude d'opportunité marquant le début du projet à la garantie clôturant le projet en passant bien sur par les phases d'analyse et conception (Inception, Elaboration, Construction, Transition). Notre processus lui se limite à décrire ces phases d'analyse et conception le schéma suivant montre le recouvrement entre le RUP et notre processus.



12.2 NOTRE PROCESSUS ET UML EN ACTION (VALTECH)

On peut facilement identifier un point commun entre le processus de Valtec et notre processus. Valtec propose un cycle en Y avec mené en paralelle l'analyse et la coception physique. Notre processus propose lui aussi une mise en paralelle des activités d'analyse et conception ce ci est decrit dans le schémas ci dessous.





12.3 NOTRE PROCESSUS ET LA « MODELISATION AVEC UML » (PIERRE-ALAIN MULLER)

Le livre de Pierre-Alain Muller se différencie de notre processus par deux points premièrement ce livre ne décrit pas formellement un processus de modélisation mais présente plutôt le principes fondamentaux que l'on doit manipuler pour la modélisation. Deuxièmement il est présenté comme un accompagnement à un cour. Notre processus lui décrit de façon très formelle un processus d'analyse et conception étapes par étapes il est destiné a une utilisation dans le cadre industriel.

En conclusion on peut dire que notre processus est une instance du RUP adapté a une utilisation dans le cadre industriel du formalisme UML.

13 DOCUMENTATION

Ce paragraphe propose un récapitulatif des éléments qui permettent de construire la documentation d'analyse et de conception d'un projet réalisé selon notre processus. La documentation dépend étroitement du type de projet, et des contraintes et exigences contractuelles. La documentation est rédigée en plusieurs passages, selon le modèle itératif de notre processus. Cette approche permet d'obtenir une documentation fidèle de l'état final de la conception, puisque son évolution aura été parallèle à la conception.

13.1 DOCUMENTATION D'ANALYSE ET DE CONCEPTION

Nous proposons ci-dessous un plan-type de la documentation d'analyse et de conception, qui sera regroupée en un seul document. Nous indiquons la ou les principales activités qui produisent de l'information destinée à chaque chapitre et paragraphe. Chaque élément d'analyse ou de conception doit être justifié, de manière à conserver une trace des choix de conception retenus.

Tableau 1 - Plan-type de la documentation de conception

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Glossaire	A-B-C	Glossaire technique du projet
Documents applicables	-	-
Documents de réference	-	-
Introduction	-	-
Objectifs du document	Démarrage	Description des points couverts par le document. Identification du niveau de détail de la conception.
Organisation du document	Démarrage	Description de la structure principale du document.
Présentation du système	-	-
Objectifs du système	A-B-C	Définition des principales exigences fonctionnelles.
Contexte du système	B	Description des interactions entre les entités externes et le système.
Définition des acteurs	A	Définition de chaque acteur, hiérarchies d'acteurs éventuellement construites, diagrammes d'interaction les concernant.
Analyse du domaine	C	Identification des interactions entre acteurs et cas d'utilisation. Identification des objets métier.
Modèle par cas d'utilisation	B-C	La description des cas d'utilisation se basera sur le modèle fourni en annexe. Elle pourra être organisée par package. Pour chaque cas d'utilisation, on incluera les diagrammes utilisés : diagrammes de séquence, diagrammes de collaboration, diagrammes partiels de classes.
Architecture physique	α	Identification des principaux éléments physiques du système (diagrammes de déploiement).
Architecture générale du système	-	On détaillera la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.
Description des packages	D	Rôles et contenu principal de chaque package.
Interactions entre packages	E	Diagrammes de packages, diagrammes de séquence ou de collaboration entre packages.
Identification des processus	β	Diagrammes de composants.
Architecture détaillée du système	-	On détaillera la hiérarchie des paquetages le cas échéant, ainsi que les modèles de conception appliqués.
Description du paquetage XXX	E-F-G-H	Pour chaque paquetage XXX, on inclue la liste des classes contenues, les diagrammes d'interactions (séquence, collaboration), les diagrammes de classes.

PLAN-TYPE	ACTIVITES	DESCRIPTION / CONTENU-TYPE
Allocation des classes	γ	Pour chaque processus, on donnera la liste des paquetages et/ou classes prises en charge.
Traçabilité	-	-
Principes de traçabilité	Démarrage	Identification des matrices de traçabilité applicables (décrites en annexe).
Annexe 1 - Dictionnaire des classes	-	Le dictionnaire pourra être organisé selon la méthode la plus appropriée permettant un accès rapide à la description d'une classe.
Classe XXX	I-J-K	Pour chaque classe, on décrira ses attributs, son comportement (méthodes, diagrammes d'états-transitions). Le niveau de détail est fonction du projet.
Annexe 2 - Traçabilité	-	-
Exigences / cas d'utilisation	Avant D	Matrice de traçabilité entre les exigences utilisateur et les éléments d'analyse identifiés lors des activités A, B et C.
Cas d'utilisation / analyse objet	Avant I	Matrice de traçabilité entre les éléments d'analyse identifiés lors des activités A, B et C, et les éléments de conception identifiés lors des activités D à H.
Analyse objet / Conception des classes	Avant γ	Matrice de traçabilité entre éléments de conception identifiés lors des activités D à H, et les classes identifiées lors des activités I, J et K.

Le contenu de la documentation vit tout au long des phases d'analyse et de conception. Nous proposons le planning suivant pour les différentes livraisons de la documentation.

Tableau 2 - Livraisons de la documentation de conception

LIBELLE DE LA DOCUMENTATION	QUAND	CONTENU
Document d'analyse projet	Après C	Eléments de modélisation consécutive aux étapes A, B, C et α.
Document de conception préliminaire	Après H	Document d'analyse projet, complété par les éléments de modélisation produits lors des étapes D à H et β.
Document de conception détaillée	Fin du processus	Document de conception préliminaire, complété par les éléments de modélisation produits lors des étapes I, J, K et γ.

Ces deux tableaux présentent ce que la méthode garantie en terme de contenu pour produire une documentation d'analyse conception convenable, quand produire cette documentation elles est adaptable aux besoins particulier suivant la culture des modeleurs.
Notre méthode doit pouvoir garantir de la même façon le contenu nécessaire pour produire un plan de validation ou un manuel d'interface.

13.2 PLAN DE VALIDATION

Précisons tout d'abord que notre processus préconise de documenter les cas d'utilisation de façon textuelle en respectant les rubriques suivantes qui seront utile pour la spécification du modèle de document:

Résumé	Présentation succincte du cas d'utilisation
Contexte d'utilisation	Conditions d'utilisation par les éléments déclenchant (fréquence d'activation, déclenchement synchrone ou asynchrone, …)
Elément déclenchant	Acteur ou cas d'utilisation
Pré-conditions	Etat stable du système nécessaire avant l'accomplissement du cas d'utilisation
Données en entrée	Données consommées

Description	Description détaillée des interactions entre les éléments déclenchant et le système
Post-conditions	Etat stable atteint par le système à la fin de l'accomplissement du cas d'utilisation
Données en sortie	Données produites
Exceptions	Cas d'erreur que le système ne sait pas résoudre

Le plan de validation contient des cas de validation qui contiennent des cas de test qui sont exécutés (cas d'exécution). Un cas de validation correspond à un cas d'utilisation de premier niveau d'analyse. Un cas de test correspond à un jeu de scénarios. Un cas d'exécution est un scénario particulier. Il y a au moins autant de cas de validation que de cas d'utilisation.
Le plan de validation présente la liste des cas de validation et, pour chacun d'eux, le test à exécuter si une anomalie est déceler. Une anomalie majeure (bloquante) peut stopper le plan de validation.

Le modèle de document du plan de validation, qui doit donc présenter au moins la liste des cas de validation de tous les cas d'utilisation , se présente comme ceci :

TEST TYPE (*)	TEST IDENT	USE CASE REFERENCE	USE CASE AND VALIDATION TEST DESCRIPTION	NEXT TEST IF ANOMALY	TEST TRACEABILITY (fulfils)
		Récupérer le *nom* du cas d'utilisation	Récupérer le champ *description* du cas d'utilisation		

(*) Type de test (Fonctionnel, Robustesse, Performance, Ergonomie, Documentation)qui pourrait être ajouté dans le modèle au niveau des cas d'utilisation.

Les test sont décrit décrient avec des Tests Description Form (TDF). Les cas de validation sont décrient dans la TDF Form 1 et les cas de test ainsi que les cas d'exécution sont décrit dans la TDF Form 2. Pour chaque cas d'utilisation, on va générer une TDF Form1 à partir des différents champs de la description du cas d'utilisation. Ensuite on va exécuter le test en suivant les instructions décrites dans la TDF Form 1, on compare les résultats obtenus avec les résultats attendus et on rempli les TDF Form 2.
Les templates des TDF sont les suivants :

Test Description Form 1	
Project : Récupérer le *nom* du projet	**Software version :**
Test ident :	**Test title :** Récupérer le *nom* du cas d'utilisation
Test type : ☐ Functional ☐ Robustness ☐ Performances ☐ Ergonomic ☐ Documentation	
OBJECTIVES : Valider la fonctionnalité : "champ *Résumé*"	
CONTEXT : Champ *Contexte d'utilisation*	
PRE-REQUIREMENTS : Champ *Pré-conditions*	
PROGRESS INSTRUCTIONS : Champs *Description*	

INPUT : Champs *Données en entrée*
EXPECTED RESULTS : Champs *Exceptions*

Test Description Form 2		
Project : Récupérer le *nom* du projet	**Software version :**	**Execution date :**
Test ident :	**Test title :** Récupérer le *nom* du cas d'utilisation	
OBSERVED PROCESS :		
OBSERVED RESULTS :		
DIAGNOSTIC :		
COMMENT :		
CONCLUSION : ☐ Test Accepted	☐ Test Partly Accepted	☐ Test Refused

13.3 MANUEL D'INTERFACE

PLAN-TYPE	INFORMATION A RECUPERER
Objet	-Texte formaté *Objet*
Documentation associée	-Texte formaté *Documentation associée*
Documents applicables	-Texte formaté *Documents applicables.*
Documents de référence	-Texte formaté *Documents de référence.*
Description des interfaces externes	
Environnement	- Diagramme général des entités passives et autres systèmes et sa documentation -Pour chaque entités passives et chaque autre système : son nom et sa documentation -Diagramme de flux de données statique et dynamique et leur documentation
Liste des interfaces	Pour chaque entités passives et chaque autre système: -extraire du diagramme de flux de données statique un diagramme spécifique de flux de données pour l'entité passives ou l'autre système -Pour chaque messages envoyé ou reçus : leur nom et leur documentation qui doit comprendre (à rajouter dans la méthode) le producteur, le consommateur, le rôle, le Format, les contrôles auxquels il sont soumis lors du chargement et les point particulier
Description des Interfaces internes	
Liste des interface	-diagrammes de paquetage et diagrammes de séquence traduisant les échanges entre paquetages -Extraire les classes d'interface de chaque paquetage, lister les méthode, les données consommées et produites et présenter leur documentation
Variable d'environnement	-Récupérer les classes stéréotypées *Variable d'Environnement* et leur documentation. (à rajouter dans la méthode)
Type et constante	-Récupérer les classes stéréotypées *Type* ou *Constantes* et leur documentation. (à rajouter dans la méthode) -Construire éventuellement un diagramme de classe pour chaque Type ou Constante pour visualiser les classes les utilisant.

14 ACCOMPAGNEMENT ET SUPPORT

Notre processus guide les Analystes, Architectes Système et Concepteurs dans leur pratique d'UML, ce qui induit une productivité et une qualité accrues. De manière à faciliter les premières applications de ce processus sur des projets, les experts peuvent assister les équipes de développement à différentes étapes de leurs projets.
Il faut tout d'abord assigner un expert qui accompagne l'équipe sur la durée du projet.
Ses interventions dans les phases d'analyse du projet consisteront à :
Installer et adapter l'outillage support au processus selon les exigences propres du projet ;
Former l'équipe au langage UML et au processus (2 jours) ;
Accompagner les analystes dans leurs modélisations – 2 jours par exemple peuvent être alloués pour chaque activité des deux phases d'analyse, analyse des besoins et analyse-objet.
L'expert interviendra ensuite dans les activités E et F de la phase de Conception de l'architecture, à hauteur de 2 jours par exemple.
A cette étape du projet, l'architecture du système est définie et l'équipe projet a acquis les compétences nécessaires à l'exécution des activités de conception détaillée.
L'expert interviendra alors en tant que relecteur des modèles de paquetage élaborés en activités G et H, en consacrant par exemple 2 heures de relecture à un paquetage simple et 1 jour à un paquetage complexe.
A la fin de la phase de Conception de l'architecture, l'architecture de l'application est validée et les constituants de haut niveau sont identifiés. L'expert agit aussi en tant que relecteur dans la suite du projet sur les activités de conception des classes, en association avec les responsables Qualité. Les points critiques sont étudiés plus particulièrement.
Sur les phases de codage et de test, l'expert pourra aussi intervenir afin d'optimiser l'exploitation des modèles de conception et garantir la cohérence entre modèles de conception et code de l'application.

15 MISE EN ŒUVRE DU PROCESSUS

Le processus décrit ici résulte d'une longue expérience dans le domaine du développement d'applications logicielles complexes, associée à une expertise confirmée en techniques de Génie logiciel. Les phases et activités identifiées dans ce processus se réfèrent à un développement *idéal*, démarrant par la formalisation des besoins faite avec le client et allant jusqu'aux activités d'implémentation et de test, et pour une application *équilibrée* qui combine à la fois architecture, traitement de données et interfaces graphiques. Dans la réalité, sur un projet concret, le chef de projet doit appliquer et adapter ce processus en fonction de ses besoins propres.

15.1 ADAPTATION DES ACTIVITES EN FONCTION DU PROJET A REALISER

Si le projet est avant tout un problème d'analyse des besoins et d'architecture du système, avec fort degré de réutilisation et acquisition massive de composants disponibles sur étagère, les efforts seront mis sur les phases d'analyse et de conception de l'architecture, alors que les activités de conception objet seront réduites.
A l'inverse, le projet peut démarrer en phase de conception seulement. Dans ce cas, il appartiendra au chef de projet de vérifier que les éléments nécessaires à la bonne exécution des activités de conception sont disponibles en entrée. Si ce n'est pas le cas, une tâche de production de ces éléments (par adaptation de l'existant) devra être planifiée. Sinon, les bénéfices promis par ce processus pourraient ne pas être atteints.

15.2 ADAPTATION DES ACTIVITES EN FONCTION DE LA NATURE DE L'APPLICATION

De même, selon la nature réelle de l'application à développer, des activités peuvent être réduites ou au contraire renforcées. Il est évident que les activités K et J dépendent directement de la complexité des bases de données et interfaces graphiques à concevoir. Dans le cas d'applications distribuées avec de fortes contraintes temps-réel, la phase de conception de l'architecture sera plus importante, ainsi que les activités "alpha", "bêta" et "gamma" de la phase de conception physique. Dans le cas d'applications non temps-réel s'exécutant sur une machine standard, comme une station UNIX ou Windows NT, sans connections physiques spécifiques, la phase de conception physique sera très réduite.

15.3 ROLES DES INTERVENANTS AU COURS DU PROCESSUS

En ce qui concerne les intervenants sur le projet, les compétences demandées dépendent directement des phases en cours. Les analystes interviendront dans les activités A, B, C et D. Les architectes Système interviendront dans les activités E, F, G et H, ainsi qu'en conception physique dans les activités "alpha", "bêta" et "gamma" (conjointement avec les concepteurs). Les concepteurs interviendront dans les activités I, J et K, et en conception physique avec les architectes Système.

15.4 APPLICATION DU PROCESSUS AUX SYSTEMES COMPLEXES

Dans le cas de systèmes complexes, ce processus présente l'avantage de combiner les apports des techniques d'analyse et de conception par objets et les techniques éprouvées de raffinements successifs (telles que supportées dans HOOD par exemple). Il fournit des moyens pour répondre à des questions comme « Comment organiser des centaines de classes ? », « Comment organiser l'équipe ? ».

15.5 CONFORMITE AU « UNIFIED PROCESS »

Notre processus est conforme au « Unified Process » tel que défini dans [2] ; il en est en fait une instanciation. Cette conformité permet aux équipes utilisant notre méthode de l'appliquer dans des contextes où le Unified Process est requis, soit car il a été exigé par le client, soit car il s'agit d'une collaboration avec d'autres partenaires pour laquelle le Unified Process a été choisi.

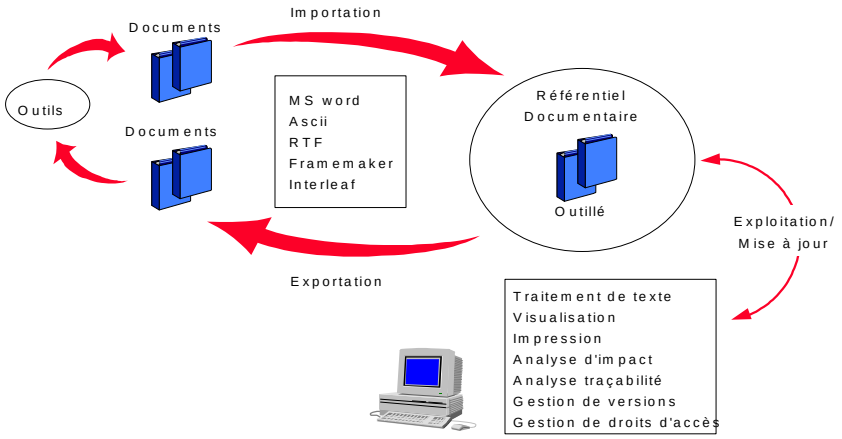
15.6 OUTILLAGE SUPPORT

Enfin, lors de l'élaboration du processus, il a été pris grand soin de vérifier que les outillages appropriés sont disponibles (un processus sans support n'apporte pas les bénéfices escomptés). Un grand nombre d'outils commerciaux UML sont disponibles à des tarifs attractifs. De plus, pour augmenter sensiblement les gains en qualité et productivité, ce processus se dotera progressivement d'utilitaires complémentaires concernant essentiellement la vérification de modèles et la génération de code, afin d'obtenir des modèles d'analyse et de conception de qualité et afin d'automatiser autant que possible les activités d'implémentation.

15.7 TRAÇABILITE

La mise en place de la traçabilté dans le processus s'intègre dans le schéma général suivant :

Figure 36 – Schéma général de traçabilité



Chaque outil de production (ici l'atelier support au processus) est complété par un ensemble de macros ou de scripts permettant de saisir les informations liées à la traçabilité. Il est possible d'associer l'outil de production à un outil de traçabilité tel que DOORS ou Rectify.

La documentation obtenue (générée par l'outil support ou de traçablité) est importée par un outil capable:
d'extraire les informations de traçabilité,
de les gérer dans une base de données,
- de réaliser des analyses d'impact, des matrices de traçabilité, …

Dans le cas de notre processus, les éléments de traçabilité sont les suivants :
l'exigence initiale référencée par un identifiant,
le lien entre un élément du modèle UML (un cas d'utilisation, un acteur, une classe,…) qui est lui aussi référencé par un identifiant différent de l'exigence et une ou plusieurs exigences initiales,

les exigences dérivées issues de la production du modèle UML qui sont elles aussi référencées.

Important : *Le lien de traçabilité est saisi par le concepteur via l'outil support du processus (à l'aide de macros ou de scripts) ou l'outil de traçabilité, dans les rubriques textuelles associées aux éléments du modèle.*

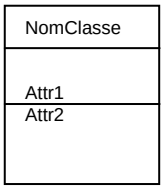
L'approche proposée permet donc de suivre les exigences (études d'impact, matrices…) depuis la description initiale du système jusqu'au plus petit élément de conception voir de code

16 ANNEXES

16.1 ANNEXE 1 - NOTATION UTILISEE / DIAGRAMMES

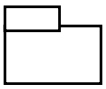
Classe

L'élément de base est une classe, représentée par son nom, caractérisée par ses attributs et ses méthodes. Une instance de classe est un objet.
Une classe est désignée par un nom ou un substantif représentatif de l'objet qu'elle modélise. Une méthode est représentée par un verbe à l'infinitif (éventuellement suffixé par un complément d'objet) indiquant l'action à accomplir.



Catégorie

Une catégorie (ou paquetage ou package) est un regroupement de classes.



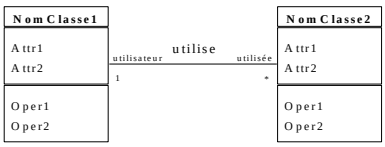
Relations

Les relations entre les classes peuvent être :

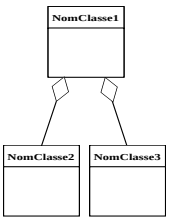
des relations d'association, scindées en deux types : coopération et composition,

des relations d'héritage.
Une relation est nommée par une forme verbale soit active, soit passive.
Une **relation de coopération** est également appelée relation d'utilisation ; elle est représentée par un trait simple, caractérisée par un nom. Un rôle peut être associé à chacune des extrémités de la relation, de même que la cardinalité, c'est à dire le nombre d'instances de chacune des classes impliquées dans la relation.
Une **relation de composition** est également appelée agrégation. Elle est représentée par un trait dont une extrémité est un losange, indiquant la classe de regroupement.
Une **relation d'héritage** est une relation orientée, qui traduit une spécialisation dans un sens et une généralisation dans l'autre sens.

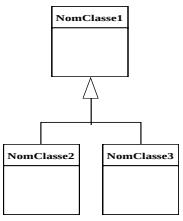
Relation de coopération



Relation de composition

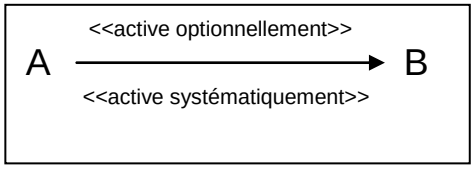


Relation d'héritage



Relations "Uses" et "Extends"

Afin de lever tout problème d'interprétation possible sur les relations Use et Extend entre cas d'utilisation, la définition suivante peut être considérée :
"B étend A" signifie que A active B de manière conditionnelle (optionnelle) en fonction du contexte d'activation de A. Le stéréotype utilisé est : << active optionnellement >>.
"A utilise B" signifie que A active B obligatoirement. Le stéréotype utilisé est : << active systématiquement >>.



16.2 ANNEXE 2 - BIBLIOGRAPHIE

[1] Pierre-Alain Muller : ***Modélisation objet avec UML*** ; éditions Eyrolles ; 1997.

[2] Grady Booch, James Rumbaugh and Ivar Jacobson ; ***The Unified Modeling Language User Guide*** ; éditions Addison-Wesley ; 1998.

[3] Grady Booch, James Rumbaugh and Ivar Jacobson ; ***The Unified Software Development Process*** ; éditions Addison-Wesley ; 1998.

[4] Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides ; ***Design Patterns - Catalogue de modèles de conception réutilisables*** ; édition Thomson Publishing ; 1996.

[5] Cetus links (Internet) : ***http ://www.cetus-links.org/***, avec une page dédiée à UML.

[6] UML Documentation Resources (Internet) : ***http ://www.rational.com/uml/resources/documentation/*** ; cette page fournit un lien vers la documentation de référence (OMG) de UML.

[7] Jean Pierre Rosen ; HOOD ***An Industrial Approach for Software Design*** ; 1997.

[8] Ralph E.Johnson ; **Design Patterns, Elements of Reusable OO Software** ; 1995

[9] W.Pree ; **DESIGN PATTERNS et ARCHITECTURES LOGICIELLES** ;1997

[10] A.Canals, M.Heitz ; **CATALOGUE DE SOLUTIONS DE CONCEPTION REUTILISABLES** ; 1998