CONSERVATOIRE NATIONAL DES ARTS ET METIERS

PARIS

MÈMOIRE

Présenté en vue d'obtenir le

DIPLÔME D'INGÈNIEUR C.N.A.M

en

INFORMATIQUE

par

Laurent DONGÈ

Réalisation d'un projet de merchandising

Soutenu le jeudi 6 décembre 2007

Jury

Présidente : Isabelle COMYN-WATTIAU, Professeur

Membres : Tatiana AUBONNET, Maître de conférence

Jacky AKOKA, Professeur

Pierre AUDOIN, Chef de service

Pierre ESTIVALET, Merchandiser Senior

Résumé :

Aujourd'hui, afin d'être plus compétitifs, les groupes de distribution ont besoin de se doter d'outils d'optimisation de la présentation des produits sur leurs surfaces de vente. C'est ce que permet l'application Merchandising. On gagne en productivité en automatisant les tâches manuelles et en simplifiant les outils.

Pour rester souple par rapport à l'utilisation d'une solution propriétaire, nous avons fait le choix de gérer les notions « métier » propres à l'entreprise dans une solution spécifique.

Cette application a dû être intégrée dans un Système d'Information d'entreprise très hétérogène. L'utilisation d'un ETL permet de réduire la complexité du système en définissant un cadre de développement précis et en masquant les différentes implémentations techniques.

Plus on en élargit l'emploi, plus la complexité du système diminue. Il nécessite cependant certains outils annexes pour répondre complètement aux besoins.

D'autre part, nous avons utilisé J2EE pour implémenter les interfaces WEB. C'est une architecture qui répond à nos besoins d'IHM. Elle permet de concevoir des applications structurées, maintenables, mais avec l'inconvénient d'engendrer des coûts de développement importants.

L'architecture WEB utilisée n'est pas propriétaire mais basée sur des produits Open Source. Elle présente un avantage financier certain du fait de l'absence de dépenses liées aux licences.

Enfin, des séries de tests de montée en charge ont permis de valider la qualité de services rendus aux utilisateurs.

La mise en œuvre de l'application répond aux objectifs essentiels. Il est cependant nécessaire de mieux répondre aux besoins de supervision, de reporting et de gestion de la qualité des données.

Mots-clés :

ETL, conception d'application J2EE, architecture, Open Source, tests

Summary :

Today, in order to be more competitive, distribution groups need tools to optimize product presentation in their stores. The Merchandising application allows this. With the automatisation of manual tasks and the use of simple tools, productivity has been increased.

To remain flexible compared to the use of an owner solution, we made the choice to manage the concepts with a solution specific to the company.

This application has been integrated into a heterogeneous information system with an ETL. The ETL reduces the system complexity, as it defines a precise development framework and it hides the various technical implementations.

The more the ETL is used, the more the system complexity decreases. However, ETL requires additional tools to satisfy all our needs.

In addition, we use J2EE to implement the WEB interfaces. It is an architecture which meets our HMI needs. It allows the development of structured applications, easy to support, but the development costs generated are very high.

The WEB architecture used is based on Open Source products. There is a financial advantage in using these products as there is no licence to pay.

Stress Tests have allowed the validation of the quality of services given to the users.

The implementation of the application has satisfied the essential needs. However, some evolutions are needed in order to have tools capable of answering requests of supervision, reporting and management of data quality.

Keywords :

ETL, J2EE application design, architecture, Open Source, tests

REMERCIEMENTS

J'ai effectué mon « stage » au sein du service Reporting et Pilotage du département administratif de la Direction de l'informatique du Groupe MONOPRIX, sous la responsabilité de Jean-François Lemaire, chef de service et en collaboration avec Pierre Audoin, responsable du Merchandising PGC. Ce « stage » a débuté en septembre 2004.

Suite à la réorganisation de la Direction de l'informatique, en début 2007, le projet Merchandising est désormais rattaché au service référentiel du département Marchandise dont la responsable est Isabelle Renoncet.

A ma demande, j'ai été transféré dans l'équipe d'Isabelle Renoncet à partir du premier avril 2007 pour continuer à piloter le projet.

Ce mémoire n'aurait pu voir le jour sans le soutien de mes professeurs du CNAM, de mes collègues, de mes amis et de ma famille.

Parmi toutes ces personnes, je tiens à exprimer mes remerciements à Monsieur Jacky Akoka à qui je dois d'avoir pu présenter ce mémoire après neuf années de cours du soir.

Je remercie tout particulièrement Monsieur Pierre Audoin pour son aide précieuse et son soutien sans faille tout au long du projet, ainsi que les ingénieurs d'étude qui ont travaillé sur le projet.

Je remercie également Madame Isabelle Renoncet pour la compréhension dont elle a fait preuve dès mon arrivée dans son service.

Un travail de si longue haleine suppose le soutien des proches. Je remercie mon épouse Xiao ping pour le soutien qu'elle m'a toujours apporté et sa patience. Merci également à Léona et Roland pour leur encouragement.

Introduction

Aujourd'hui, de nombreuses entreprises choisissent les meilleurs outils informatiques pour répondre à leurs besoins afin d'être le plus compétitif possible. Elles se dotent ainsi d'ERP (Entreprise Resource Planning) ou de suites logicielles spécialisées. En suivant cette démarche, leurs systèmes d'information gagnent en complexité. Choisir la meilleure solution pour répondre à chacun de ses besoins multiplie le nombre de technologies et de standards utilisés au sein de l'entreprise.

Nous allons nous intéresser aux problématiques liées à cette démarche d'entreprise, à travers la mise en œuvre de la suite logicielle SMART Office au sein du groupe de distribution de proximité MONOPRIX. Cette suite permet d'élaborer des dossiers de merchandising fournissant des préconisations de mise en rayon des produits à l'attention des magasins.

Dans un premier temps seront présentés le projet, les grands principes du domaine fonctionnel lié au merchandising, les raisons de l'origine du projet, ses contraintes, ainsi que les étapes importantes du projet, l'étape pour arrêter un choix sur une offre logicielle particulière.

Une présentation des méthodologies, technologies et concepts utilisés sera suivie par l'exposé de la mise en œuvre de l'application merchandising. La démarche pour mener à bien le projet sera précisée.

Après avoir choisi la suite logicielle qui semble correspondre le mieux à nos besoins, il reste à l'intégrer dans le Système d'Information existant. Il faut alors récolter l'information nécessaire au bon fonctionnement de la suite logicielle, puis fournir les éléments permettant de diffuser les dossiers de préconisations de mise en rayon aux magasins. Après une présentation du Système d'Information existant, nous nous attacherons aux moyens d'intégrer différentes applications d'entreprise, en particulier à l'aide d'ETL (Extract Transform and Load) et d'EAI (Entreprise Application Integration), dont le point fort est de faire cohabiter des systèmes hétérogènes.

Mais ces produits ne suffisent pas à eux seuls à fournir une solution applicative complète, il faut également se doter d'interfaces, afin de pouvoir administrer l'intégration des informations récoltées, de diffuser les documents de préconisations réalisés et de suivre l'élaboration des dossiers de préconisations. Il est alors nécessaire d'organiser son application [JOU05a] et de choisir des outils de développement. Nous avions une volonté forte d'utiliser des outils permettant de rendre le code plus lisible et ainsi de simplifier la maintenance de ce dernier afin de ne pas se retrouver avec des développements de type « spaghetti ».

Se pose ensuite la question de l'architecture à mettre en place, dans laquelle nous allons déployer les développements réalisés. En fonction du nombre d'utilisateurs que nous souhaitons supporter, de la lourdeur des développements, du temps de réponse attendu, du niveau de sécurité désiré et de la tolérance de panne exigée, quelle architecture devons-nous mettre en œuvre ? Nous présenterons alors la solution Open Source basée sur Apache et Tomcat.

Enfin, comment pouvons-nous savoir que les développements réalisés et déployés dans l'architecture, ainsi que cette dernière, vont vraiment offrir la qualité de service attendu ? La réalisation de stress tests permettra de répondre à cette question. Nous présenterons alors ce que sont des stress tests, leur utilité, des offres du marché permettant de les réaliser, la démarche utilisée et leurs limites.

1. Le projet

Objectif

Le projet a pour but de renouveler l'application de merchandising PGC (Produit Grande Consommation). Cette application est utilisée au sein de l'enseigne de distribution de proximité MONOPRIX par une cellule merchandising de 10 personnes.

Elle permet d'élaborer des dossiers de préconisations de mise en rayon à destination des 250 magasins du groupe.

En se dotant d'un outil plus perfectionné, l'entreprise souhaite réduire les coûts associés aux processus « métier » liés à l'élaboration des dossiers de préconisations. Elle compte ainsi augmenter la productivité en limitant les interactions humaines et en raccourcissant la durée de ces processus.

Les acteurs

L'élaboration des dossiers de préconisations implique de nombreux acteurs. Elle est faite par la cellule merchandising en collaboration avec les bureaux d'achat, le marketing, les fournisseurs et la direction des ventes.

En amont, l'achat élabore les collections. Le marketing fournit une stratégie. Les fournisseurs font des propositions dans leurs spécialités.

En aval, la cellule merchandising organise la mise en scène des produits et la vente exploite les dossiers de préconisations.

Enfin, les merchandisers travaillent étroitement avec les fournisseurs qui procurent les visuels et les caractéristiques de leurs produits.

Contexte

Le schéma ci-dessous positionne l'application de merchandising par rapport aux principaux acteurs, applications et systèmes centraux auxquels elle se rapporte.

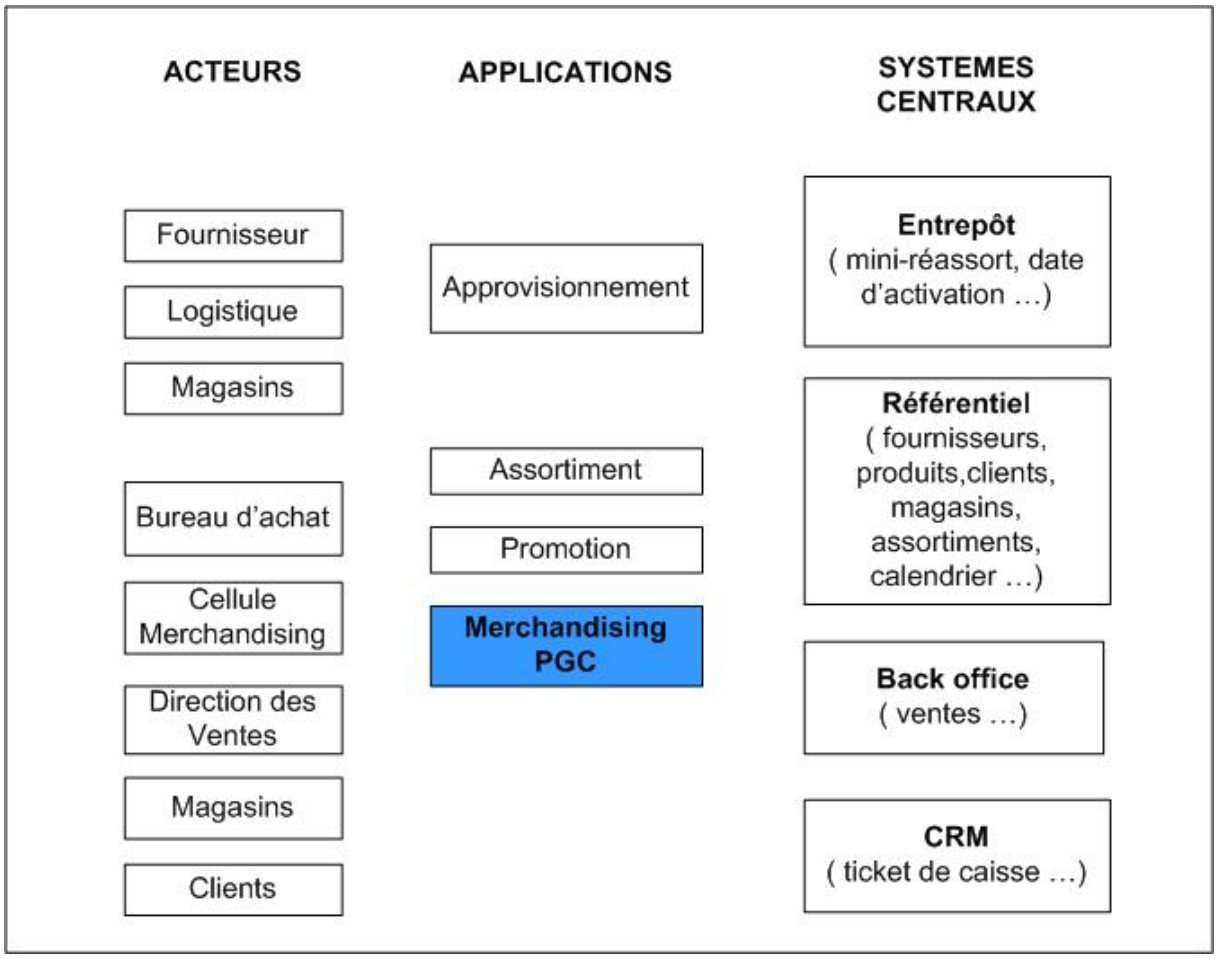


Figure 1 : Vision Globale Acteurs, Applications, Systèmes centraux

Sujet du mémoire

Le sujet de ce mémoire est la « réalisation d'un projet de merchandising ». Le merchandising n'est pas le sujet principal de ce mémoire. Nous allons nous intéresser plutôt aux aspects techniques qu'aux aspects fonctionnels et à la démarche qui a été suivie pour mener à bien ce projet.

Les problématiques auxquelles nous allons nous attacher sont :

- La gestion et le management de projet
- La façon d'intégrer une suite logicielle au sein d'un Système d'Information hétérogène existant.
- Le choix de technologies WEB et d'outils de développement. Ainsi que les raisons de ce choix.
- Le choix d'une architecture WEB et la validation de cette architecture.

Le projet merchandising va permettre d'illustrer ces quatre problématiques. L'objectif n'est pas d'exposer dans ses moindres détails les problématiques liées à ce projet mais plutôt de se servir de ce cas concret pour apporter des réponses à ces problématiques.

Le but de ce mémoire n'est pas non plus de traiter de ces sujets de façon exhaustive, mais de se concentrer sur les points réellement abordés lors du projet et qui ont permis de le mener à bien et de répondre de façon satisfaisante aux besoins des utilisateurs. Compte tenu des délais et des impératifs du projet, il est difficilement possible de maîtriser pleinement l'ensemble des technologies et des logiciels qui vont être utilisés lors de ce projet.

L'important est de saisir l'essentiel de ce qu'apporte chacun des outils utilisés et de s'en servir le plus simplement possible afin que le projet reste évolutif et facilement maintenable.

1.1. *Le merchandising*

Présentation

Le merchandising correspond à l'ensemble des techniques visant à optimiser la répartition et la présentation des produits en magasin afin d'optimiser la rentabilité et le trafic et de contribuer à l'expression de l'image de l'enseigne.

Cette définition paraît très simple, mais derrière cette simplicité se cache un certain niveau de complexité.

En effet, pour arriver à élaborer des dossiers de préconisations de mise en rayons à destination des magasins, le merchandiser a besoin de nombreuses informations relatives :

- aux produits : prix, dimensions, nomenclature, caractéristiques, images des produits
- aux points de ventes : mobilier, capacité des linéaires, assortiments des produits correspondant au point de vente, historiques de vente du produit, colisage de livraison, périodicité de livraison
- à la politique de l'entreprise, l'état du marché national et son évolution
- d'un modèle de vente : écoulement des ventes sur la semaine

Le travail du merchandiser consiste, à partir de ces informations et en partenariat avec les bureaux d'achat de l'entreprise et la direction des ventes, à proposer une répartition des produits en magasin afin de répondre pleinement aux ventes et à ne pas avoir de rupture de produit en rayon. Il va faire des préconisations sur le nombre de facings à accorder à un produit dans le linéaire du magasin et sur son emplacement. Ces préconisations se font à partir de données quantitatives dont nous avons fait une liste non exhaustive ci dessus et qui vont être extraites du Système d'Information de l'entreprise mais également de notions qualitatives telles que les couleurs des visuels correspondant aux différents produits. Le merchandiser va par exemple chercher à obtenir le meilleur rendu visuel afin de rendre la répartition des produits la plus agréable possible pour le consommateur. L'aspect visuel est donc une composante importante de l'élaboration des dossiers de préconisations.

Dossiers de préconisations

Les dossiers de préconisations à destination des magasins présentent toujours le même type de structure :

- une page de garde (*Figure 2*)
- un argumentaire (*Figure 3*)
- une présentation du marché national et de l'enseigne, l'évolution du marché
- des plans de masse (*Figure 4*)
- les planogrammes en couleurs avec les visuels des produits (*Figure 5*). Il existe plusieurs planogrammes en fonction du linéaire du magasin. Un planogramme correspond à une représentation graphique des mobiliers en magasin sur lesquels on a

positionné des produits. Les produits sont facilement identifiables à l'aide de photos les représentant.

- Liste des produits du planogramme (*Figure 6*): numéro de produit sur le planogramme, ean du produit, désignation, nombre total de facing, colisage de réassortiment du produit
  (correspond à la quantité minimale que l'on peut livrer au magasin).

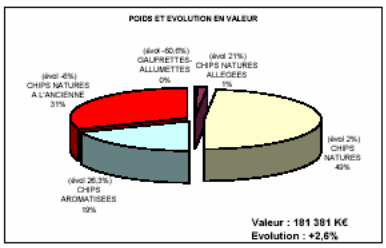
Le magasin, à l'aide de ces dossiers, optimise de manière rapide la rentabilité de son point de vente sans avoir à faire des analyses complexes et fastidieuses. En effet, ces dossiers sont le fruit d'une analyse conjointe du merchandising, des bureaux d'achats et de la direction des ventes qui tient compte de la capacité des linéaires, de la rotation des stocks et des nouveaux assortiments des produits. Ces dossiers facilitent donc le travail des magasins mais permettent également d'avoir une offre homogène sur l'ensemble des points de vente de l'enseigne. Ils facilitent donc la mise en place de la politique de marketing de l'entreprise.

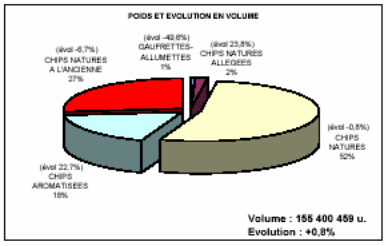
Figure 2 : Page de garde

MONOPRIX
dans ville il y a vie

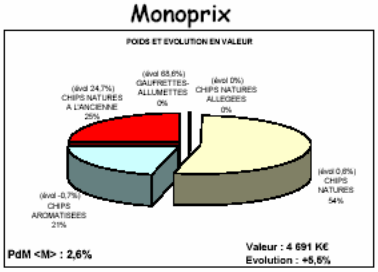
Le Marché des Chips

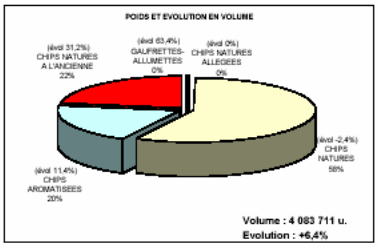
National





Monoprix





- Marché dynamique tiré par les chips aromatisées au national et par les chips à l'ancienne chez Monoprix.
- Monoprix reste sous-vendeur sur l'ensemble de la catégorie mais évolue mieux que le national.

SOURCE : IRI-SECODIP CUMUL ANNUEL MOBILE A FIN MARS 2005

Figure 3 : Le marché

MONOPRIX
dans ville il y a vie

La perception du rayon Chips

- Un rayon favorable à la découverte ➔ 2 produits sur 3 sont découverts en rayon
- Une décision qui se fait face au rayon ➔ près de 75% des décisions d'achat se font devant le linéaire
- Un temps limité consacré à l'achat ➔ 1,5 produits achetés en 52 secondes

Les critères d'achat :

❶ Le type de Chips
- Les classiques ou sels
- Les chips à l'ancienne
- Les chips aromatisées

❷ Le format

❸ La marque

Le plan de masse :

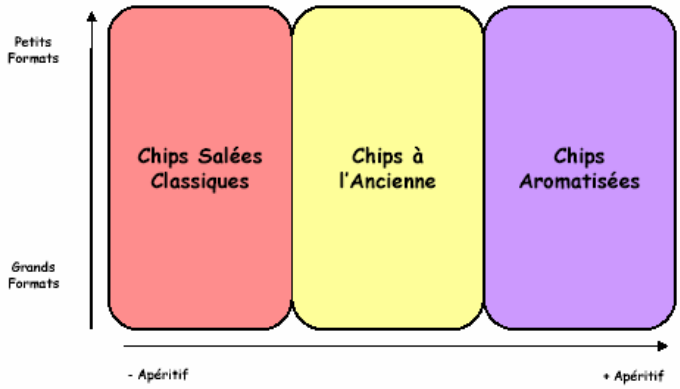


Figure 4 : Plan de masse

MONOPRIX
dans ville il y a vie

Planogramme Chips 2005

PLAN MODULE 50

Linéaire au sol : 2.5 m
Linéaire développé : 15 m
==========>
Sens de circulation

Date : 13/06/2005

Service Merchandising
Laetitia Chalifour
01.55.20.74.16

Page : 1/2

1 Les numéros noirs sur fond blanc correspondent aux marques nationales et premiers prix

2 Les numéros blancs sur fond noir correspondent aux marques propres

Figure 5 : Exemple de planogramme

MONOPRIX
dans ville il y a vie

Rapport d'Implantation Chips 2005

N° Produit	EAN	Désignation	Module	Total Facing	UR
1	3162089809046	150G CHIPS EXTRA CRAQUANTES	30	4	20
2	0728229123613	TERRA CHIPS BEUES 110G	40	2	12
3	5017764119010	150G CHIPS VINAIGRE BALSA KETL	10	3	15
4	5017764114008	150G CHIPS KETTLE POIVRE NOIR	10	3	15
5	3168930000525	135G CHIPS GOUT BARBECUE LAY'S	10	3	20
6	3350031664908	6X30G CHIPS CROUST/FONDANT MPX	10	2	20
7	3350031664892	6X30G CHIPS A L'ANCIENNE MPX	04	2	20
8	3168930000921	6X25G CHIP AROM.BBQ+BOLO+OIGNO	50	2	14
9	3350031664885	100G CHIPS CROUSTI/FONDANT MPX	10	3	24
10	3350031518867	150G CHIPS SUPER CRAQUANTES MX	05	3	20
11	3350031664915	150G CHIPS A L'ANCIENNE MPX	04	4	20
12	3350031664922	90G CHIPS ANCIEN.MOUTARDE MPX	10	2	24
13	3350031518935	90G CHIPS BACON SCHT MONOPRIX	04	3	24
14	3336970409060	120G CHIPS LEG.-30% M.G VICO	30	3	20
15	3497911900159	150G CHIPS AU SEL DE GUERANDE	20	3	16
16	3231380300009	150 G CHIPS ANCIENNE CROKY	06	4	20
17	3497911114129	125G CHIPS ONDULE SAVEUR OLIVE	30	2	20
18	3350031518966	90G CHIPS PAPRIKA SCHT MONOPRI	04	3	24
19	3168930000037	150G CHIPS FINEMEN.SALEES LAY'	10	3	20
20	3168930000181	150G CHIPS ONDULEES LAY'S	30	3	20
21	3168930000266	ST 150G CHIPS PAYSANNE CROKY	30	4	20
22	3168930001522	150G CHIPS JAMB.FUME CROKY	50	2	24
23	3168930000150	135G CHIPS LAY'S BOLOGNAISE	50	3	20
24	3162080514109	200G CHIPS NAT.1ER PRIX	10	5	28
25	3162089405040	150G CHIPS ANCIENNE NATURE VIC	10	4	20
26	3162089802030	150G CHIPS A L'ANCIEN.SAV MOUT	30	2	20
27	3168930001508	135G CHIPS LAY'S POULET/THYM	20	3	20

Page : 2/2

Figure 6 : Liste des produits du planogramme

D'autres exemples de dossiers sont présents dans l'Annexe H.

1.2. Pourquoi ce projet ?

La solution existante.

L'outil existant utilisé par les marchandisers est une vieille version de l'outil SPACEMAN de la société Nielsen qui se compose d'une application autonome installée sur chacun des postes utilisateurs.

Les merchandisers alimentent cette application à l'aide de fichiers au format EXCEL ou TXT (texte).

Le processus « métier » d'élaboration utilisé pour élaborer les dossiers de préconisations à l'attention des magasins est le suivant (*Figure 7*) :

(1) Les informations relatives aux produits et aux ventes sont obtenues à l'aide de BI Query anciennement nommé GQL qui est un requêteur du marché. Les données sont extraites sous la forme de fichier EXCEL.
(2) Les classements des magasins par chiffre d'affaire par périmètre de produits sont extraits à partir du produit Arthur Décision de Comshare. Cette solution repose sur une base Oracle. Elle permet entre autres d'avoir une vision multidimensionnelle des ventes selon les axes produits, magasins, temps et manifestations (les campagnes promotionnelles telles que Noël, Halloween sont des manifestations). Ces classements sont utilisés pour constituer des groupes de magasins qui représentent au mieux le parc de points de vente. Il est en effet impossible de faire l'ensemble des dossiers de préconisations pour chacun des magasins. Il faudrait 2 merchandisers par magasin pour mettre à jour ces dossiers une fois par an, ce qui n'est pas envisageable (la cellule merchandising ne comporte que 10 personnes et il y a 250 magasins). Le merchandiser doit effectuer de nombreuses manipulations sur les classements des magasins et les ventes correspondant aux produits afin de pouvoir utiliser ces informations dans l'application et réaliser les planogrammes.
(3) Un référentiel des différents périmètres de planogrammes et des linéaires associés est géré à l'aide de fichiers au format EXCEL.
(4) Le merchandiser a également besoin des visuels correspondant à son périmètre de produit. S'il lui manque des visuels, il doit lui-même prendre les photos manquantes des produits. Les photos sont enregistrées sur chacun des postes des merchandisers et ne concernent que leur périmètre.
(5) Le merchandiser peut alors se consacrer à l'élaboration des planogrammes et des dossiers en collaboration avec les bureaux d'achat et la direction des ventes.
(6) Le dossier une fois terminé est envoyé à la reprographie qui va en tirer un exemplaire papier pour chacun des magasins. Un exemplaire de chaque dossier est livré en magasin sans tenir compte du fait que le magasin est concerné par tout ou partie du dossier de préconisations : on envoie tout à tout le monde.

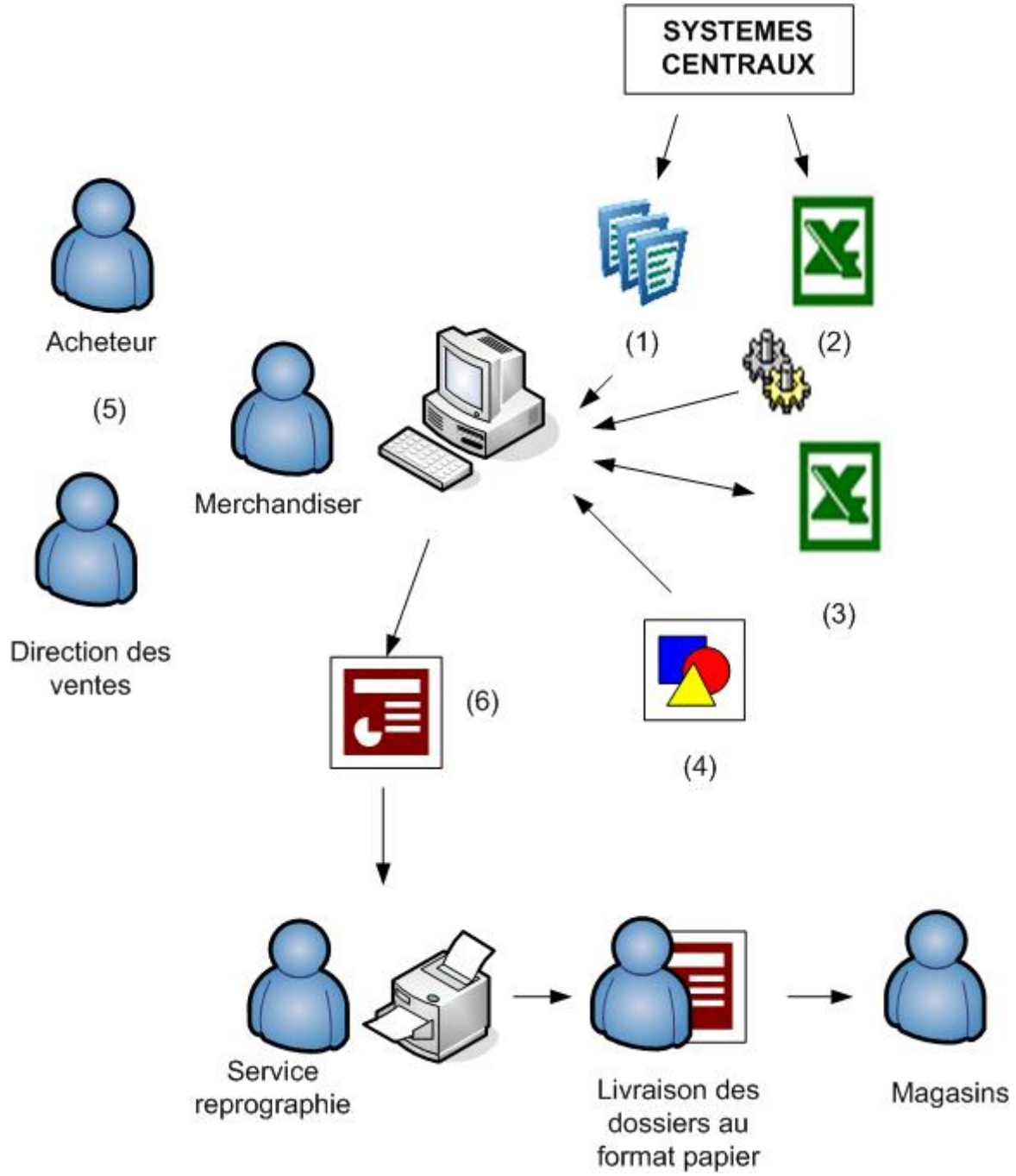


Figure 7 : L'outil d'élaboration existant

Les axes d'optimisation

Au vu de ce processus « métier », il est possible d'identifier un certain nombre d'axes d'optimisation.

On peut réduire les coûts liés à ce processus d'élaboration des dossiers de préconisations, en minimisant les interventions manuelles effectuées par les merchandisers. Cela passe par l'automatisation systématique des procédures manuelles qui peuvent l'être.

La centralisation de l'information relative aux planogrammes dans une base de données, la mise en place d'un référentiel image unique et pas de déploiement sur les postes clients de l'application ou un déploiement minimaliste permettent de simplifier et de minimiser les tâches d'administration.

Le choix d'une solution logicielle possédant une interface simple et conviviale diminue la durée de montée en compétence des merchandisers.

Si, de plus, cette solution répond à de nouveaux besoins non couverts par la solution actuelle tels que l'élaboration de dossiers de préconisations pour les surgelés qui sont présentés dans des meubles spécifiques, il est alors possible d'élargir les périmètres concernés par les dossiers de préconisations.

Le fait de souscrire un abonnement images auprès d'une société extérieure pour les périmètres nouvellement traités évite aux merchandisers de prendre eux-même en photo les produits pour lesquels on ne possède pas de visuel.

L'abandon de l'impression papier pour la diffusion des dossiers planogrammes au profit d'une diffusion au format PDF permet de ne plus imprimer et livrer des dossiers aux magasins qui ne sont pas concernés.

La solution cible

Dans la solution cible, rien ne doit être installé sur le poste client. L'utilisateur accède à un serveur distant à partir de son poste de travail. Il s'identifie sur ce serveur et accède à l'application merchandising qui permet d'élaborer les dossiers planogrammes.

Le processus d'élaboration cible est le suivant (*Figure 8*) :

(1,2) Les informations correspondant aux produits et aux ventes sont directement disponibles dans l'outil d'élaboration. Le classement des magasins est géré de façon transparente, le merchandiser n'a plus à s'en soucier. Il n'a plus à faire de manipulation fastidieuse pour obtenir de l'information utilisable.

(3) Les informations anciennement gérées dans des fichiers autonomes seront gérées dans une base de donnée au travers d'une interface WEB.

(4) Les visuels correspondant aux produits sont livrés par un prestataire extérieur auprès duquel on souscrit un abonnement. Chaque mois, on lui envoie la liste des produits. En retour, les images manquantes sont envoyées puis intégrées dans la base image.

(5) Le merchandiser peut alors se consacrer à l'élaboration des planogrammes et des dossiers de préconisations en collaboration avec les bureaux d'achat et la direction des ventes.

(6) Une fois qu'un merchandiser a achevé un dossier de préconisations, un administrateur fonctionnel met à disposition le dossier dans l'application de diffusion au travers d'une interface d'administration WEB. Le magasin peut alors récupérer le dossier à l'aide d'un navigateur WEB et l'imprimer au besoin. Les dossiers ne sont plus envoyés au service de reprographie, ni livrés sous format papier à l'ensemble des magasins.

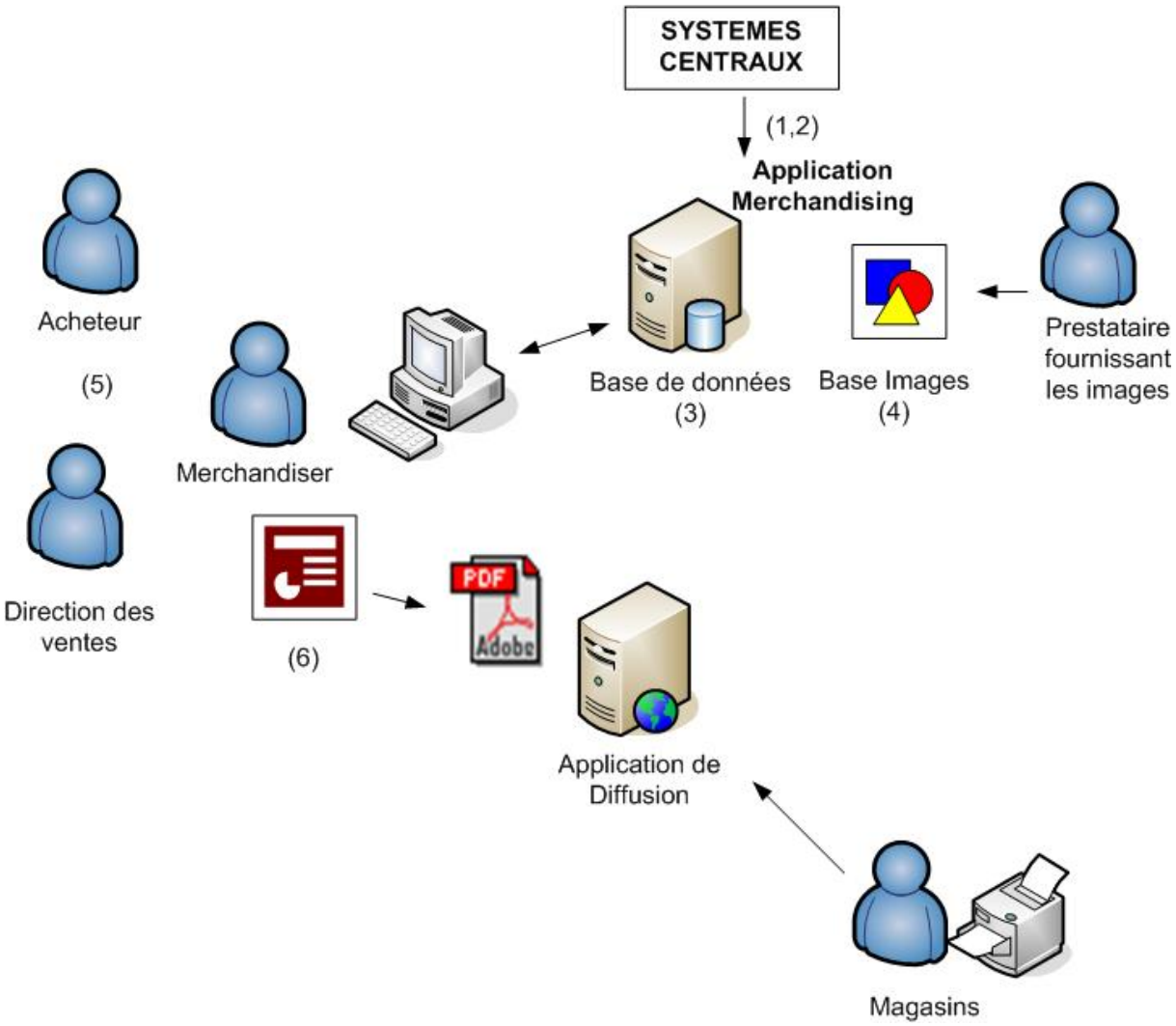


Figure 8 : L'outil d'élaboration cible

Les gains

L'automatisation d'un certain nombre de traitements permet d'éliminer de nombreuses interventions humaines et ainsi de réduire la durée du processus.

La gestion centralisée évite d'avoir un éclatement de l'information sur l'ensemble des postes utilisateurs. Cette gestion centralisée facilite les tâches d'administration.

La souscription d'un abonnement image et la mise à jour automatique de la base image libèrent les merchandisers de la gestion des images.

La mise en place d'une diffusion numérique des dossiers de préconisations permet aux magasins de les avoir plus rapidement. Il n'y a plus de livraison au magasin. Le magasin n'a plus qu'à choisir ce dont il a besoin via l'Intranet. Il peut alors imprimer les dossiers directement en magasin. De plus, il existe des gains financiers puisque cette diffusion entraîne une consommation de papier moins importante et permet de faire des économies en ne sollicitant plus le service de reprographie et le service courrier.

Ce nouveau processus « métier » va donc permettre de gagner du temps et de supprimer des tâches inutiles. On améliore ainsi la productivité du service merchandising. Le nombre de dossiers de préconisations de mise en rayon pourra être plus important en conservant le même effectif dans la cellule merchandising.

1.3. Les contraintes

Les contraintes pour mener à bien ce projet sont nombreuses et de natures diverses. Ces contraintes sont essentiellement techniques, politiques et financières.

Le Système d'Information est hétérogène. Toutes les informations nécessaires ne sont pas gérées dans ce dernier. De nombreuses informations relatives au merchandising sont dispersées dans des fichiers EXCEL sur chacun des postes des merchandisers. Il n'existe pas de référentiel centralisé contenant ces informations.

Les dossiers planogrammes nécessitent l'existence d'un référentiel image. Or, ce référentiel devrait se composer d'environ 46 000 images occupant plus de 10 Go d'espace disque. Ce qui exclut le fait de dupliquer ce référentiel sur chacun des postes, la taille de ce référentiel étant trop volumineuse.

La qualité visuelle de ces images est très importante, il est nécessaire que l'applicatif puisse au moins afficher 65 000 couleurs pour obtenir un rendu visuel acceptable.

Outre ces contraintes techniques, la DSI (Direction des Systèmes d'Information) préconise fortement l'utilisation d'un certain nombre de solutions et de standards. Le choix d'ORACLE est préconisé pour les bases de données relationnelles, ainsi que de J2EE pour les développements Web. Ces préconisations ont pour but de limiter l'hétérogénéité du système.

La DSI a également adopté une politique de déploiement minimaliste des différentes applications à mettre en place sur les postes utilisateurs. C'est pourquoi les solutions Web sont préférées aux applications Clients – Serveur.

Enfin, malgré l'importance du projet, les moyens mis à notre disposition pour mener à bien ce projet sont très limités. Et la pression sur les délais à respecter est très grande.

1.4. Les étapes importantes du projet

Choix de la suite logicielle

La première étape de ce projet a consisté à choisir la suite logicielle qui répondait au mieux aux besoins identifiés des utilisateurs ou qui pouvait facilement être adaptée aux spécificités propres de leur métier.

Nous tenions à ce que le logiciel présente une interface intuitive, simple, conviviale, ceci pour diminuer la durée d'appropriation de l'application, dans le but de raccourcir la courbe de montée en compétence des merchandisers. Ils seront plus vite opérationnels si on met à leur disposition un outil simple, plutôt qu'un outil complexe et peu ergonomique.

Nous avons donc comparé les offres des principaux acteurs du marché : Prospace de JDA, SMART Office de KLEE Commerce, Apollo d'IRI et Spaceman de Nielsen. Les offres ne reposant pas sur des bases de données relationnelles et ne permettant pas de gérer un référentiel centralisé ont immédiatement été écartées.

Les deux offres qui sont alors restées en liste étaient Prospace de JDA et SMART Office de KLEE commerce. Elles ont été analysées de manière plus approfondie. Nous avons identifié l'ensemble des points fonctionnels et techniques auxquels devait répondre la suite logicielle. Des pondérations ont été attribuées à chacun de ces points en fonction de leur niveau d'importance. L'attribution de notes nous a alors permis d'effectuer un scoring des deux solutions et de choisir, en fonction des résultats, la suite qui correspondait le mieux à nos attentes. Le scoring utilisé est fourni dans l'Annexe L.

Notre choix s'est porté sur l'offre SMART Office de KLEE Commerce. C'est cette suite qui semblait répondre le mieux aux contraintes que nous avions et qui permettait d'améliorer au mieux le processus d'élaboration des planogrammes, selon les axes d'optimisation identifiés préalablement.

Cependant, cette suite ne permettait pas d'éviter les interventions sur les postes clients. En effet, on doit installer un applicatif client sur chacun des postes des utilisateurs. Ce problème a été contourné par l'utilisation de la technologie TSE de Microsoft. Cette technologie permet d'accéder à partir de n'importe quel poste Windows 2000, 2003 ou XP à des bureaux distants gérés par un serveur distant. Nous avons donc pu être en conformité avec les attentes de la DSI qui ne souhaitait pas d'intervention sur les postes clients.

D'autres critères plus subjectifs ont également joué un rôle dans le choix du produit, tels que la disponibilité et la réactivité de l'éditeur, et aussi le fait que l'éditeur puisse fournir des prestations complémentaires comme un abonnement image. L'éditeur livre chaque mois les visuels des produits sous forme numérique correspondant à un certain périmètre.

L'offre de KLEE repose sur un référentiel composé d'une base image, d'une base relationnelle de type ORACLE et d'un ensemble de modules. Les modules vont permettre d'administrer la base image, d'élaborer les dossiers planogrammes, de gérer la base de données et d'extraire de l'information de façon automatique.

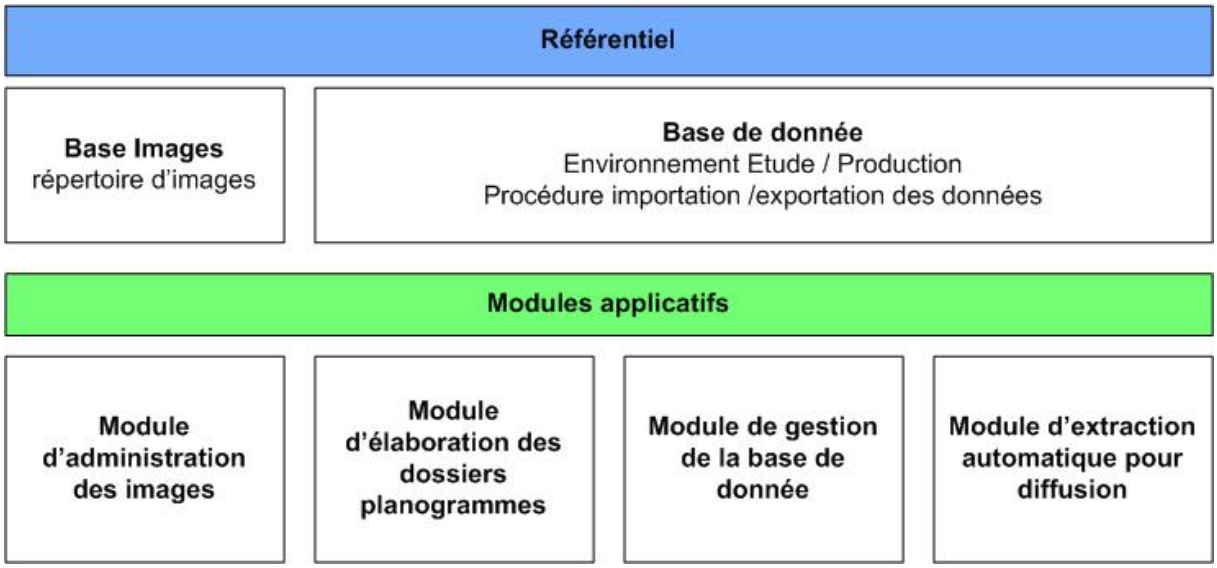


Figure 9 : Architecture de la suite logicielle SMART Office

Intégration de cette suite dans le système

La seconde étape consiste à intégrer cette suite logicielle dans le Système d'Information existant. Il est possible d'alimenter les bases de données à l'aide de fichiers « texte » en paramétrant des fichiers XML, ce qui permet d'être assez souple au niveau de l'alimentation des données. Il ne reste donc plus qu'à collecter l'information nécessaire à l'élaboration des planogrammes.

Nous avons donc identifié les traitements à mettre en place pour alimenter le référentiel et précisé la fréquence à laquelle ils devaient être exécutés.

Les grands groupes de traitements sont les suivants :

- alimentation des informations relatives aux produits
- récupération des classements des magasins en terme de ventes par ensemble de produits
- récupération des ventes par produit correspondant à chacun des magasins
- envoi de la liste des produits présents dans le référentiel à l'éditeur afin de recevoir les visuels des produits correspondant à l'abonnement image souscrit.

Hormis l'alimentation des informations relatives aux produits qui est exécutée chaque jour de la semaine, les autres traitements sont mensuels.

Etant donné que l'information présente dans le Système d'Information n'est pas directement exploitable, nous avons, avant de créer les fichiers qui serviront à alimenter le référentiel de l'offre de KLEE Commerce, mis en place un Datamart.

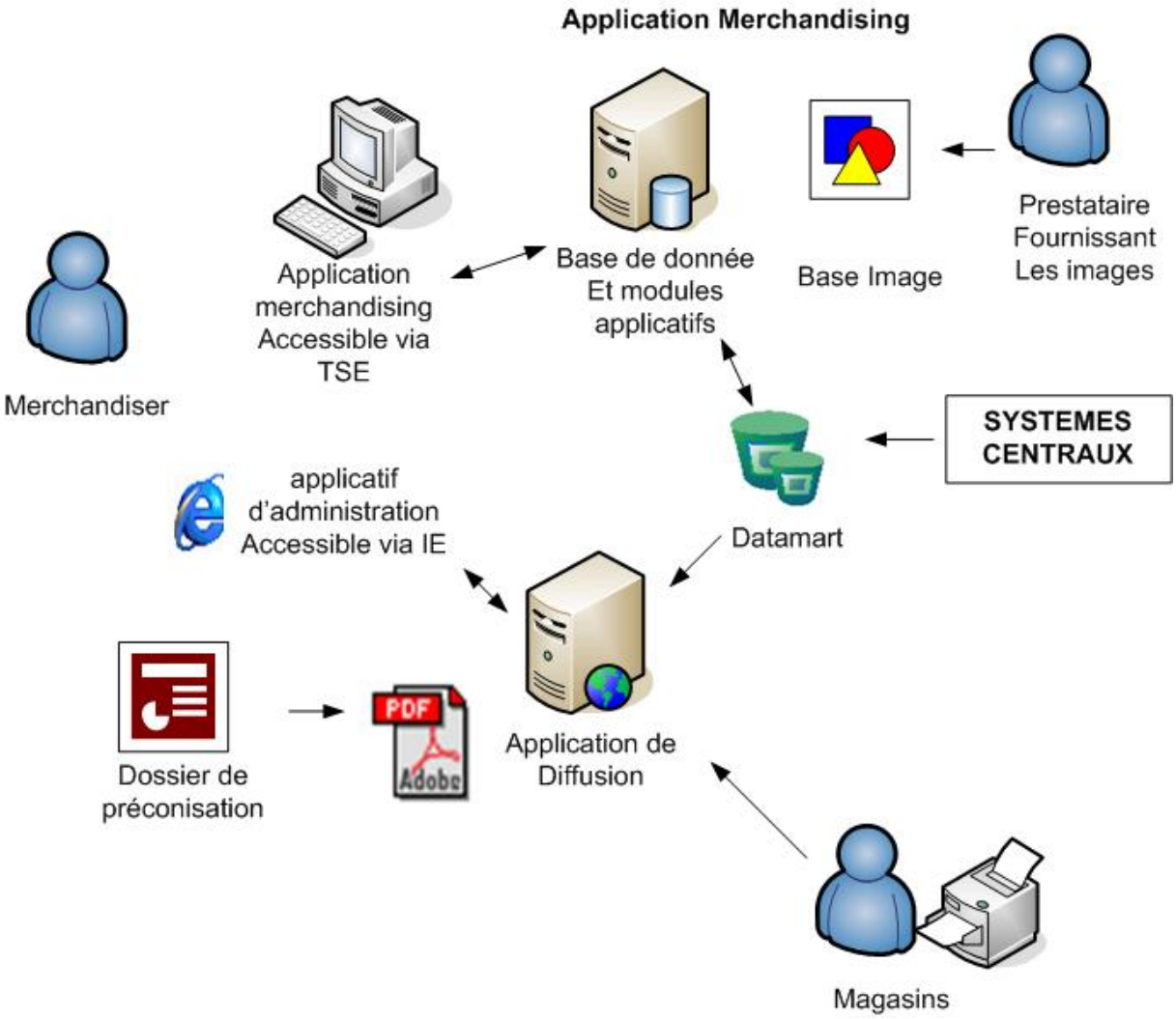


Figure 10 : Mise en place d'un Datamart

C'est une base de données qui contient toutes les informations relatives au merchandising et à l'historique des ventes sur une période d'un an permettant l'élaboration des planogrammes. Ce Datamart était initialement prévu pour fournir le bon niveau d'agrégation de l'information nécessaire à la suite SMART Office. Mais, la prise en compte d'évolutions fonctionnelles telles que la gestion de regroupements particuliers de produits en univers ont eu pour conséquence d'élargir son utilisation à la gestion de notions propres au merchandising qui n'étaient pas gérées dans le Système d'Information existant.

La gestion des univers, qui sont des regroupements de produits propres au merchandising, permet de gérer des périmètres de produits correspondant exactement aux dossiers planogrammes envoyés aux magasins. Ce qui n'était pas le cas jusqu'alors, les périmètres des dossiers correspondaient à des regroupements de sous-ensembles de nomenclature produit utilisés dans l'entreprise. Ces regroupements étaient difficilement gérables par les merchandisers.

On veut également gérer, dans cette base, la liste des dossiers planogrammes et leurs caractéristiques telles que leurs fréquences de mise à jour ou la personne en charge du planogramme. Jusqu'à présent, cette gestion était faite à l'aide de fichiers EXCEL.

Le fait de vouloir diffuser les planogrammes sous forme de fichiers PDF via l'intranet oblige également à gérer un ensemble de notions : une arborescence de diffusion, une identification des magasins pour pouvoir ensuite filtrer la liste des dossiers proposés en fonction de leurs caractéristiques.

La gestion de ces notions ne pouvait pas se faire dans l'offre SMART Office de KLEE Commerce car elles revêtent un caractère « métier » qui n'a pas lieu d'être géré dans un tel outil. Il est important que cet outil reste indépendant des données propres de l'entreprise.

Mise en œuvre d'interfaces Web

Le fait de gérer ces notions au niveau du Datamart oblige à se doter d'interfaces qui vont permettre de les administrer et de suivre la bonne exécution des différents traitements.

Pour cela, il est nécessaire de choisir des normes et des outils de développement, de mettre en place une architecture et de valider cette architecture à l'aide d'un outil de stress test que nous aurons préalablement sélectionné. Cette validation servira à s'assurer que l'application offre la qualité de service attendu pour éviter de se retrouver avec un système indisponible le jour de sa mise en production parce qu'il n'aura pas tenu la charge générée par les utilisateurs.

2. Présentation des méthodologies, technologies et concepts utilisés

2.1. Gestion et management de projet

La gestion de projet informatique est fondée sur des travaux de modélisation et de collaboration. Un client demande à un fournisseur, qu'il soit interne ou externe à l'entreprise, de mettre en place une solution à sa problématique au travers d'un modèle. Les deux parties prenantes doivent alors collaborer en bonne intelligence afin d'atteindre un objectif commun.

L'Informatique est une discipline jeune en comparaison des domaines millénaires du bâtiment et des systèmes de collaborations militaires. La réalisation d'un projet informatique traditionnel suppose en général la maîtrise des techniques de planification, des connaissances en ISO 9001 et d'UML. Il existe de nombreuses normes et méthodes utilisables dans la réalisation d'un projet informatique. On constate cependant que ces normes et ces méthodes convergent aujourd'hui vers certaines méthodes et techniques telles que les Design patterns, UML ou Java. Par le respect de règles et méthodes, on cherche à produire des solutions de qualité documentées afin de répondre aux problématiques rencontrées par les entreprises. La documentation est perçue comme un facteur permettant de contribuer à l'obtention d'une solution de qualité mais elle représente parfois un coût important.

La qualité de la collaboration et le maintien du dialogue entre le client et les fournisseurs d'un projet sont des conditions indispensables au bon déroulement du projet. La bonne réussite d'un projet informatique repose sur cette collaboration. Si le client est satisfait, il sera confiant et, en général, permettra au fournisseur d'être plus libre dans sa gestion de projet. Pour remplir pleinement son rôle le fournisseur doit se mettre à la place de son client. Il pourra ainsi effectuer une analyse pertinente de la problématique à laquelle il est confronté et répondre pleinement aux contraintes fonctionnelles et techniques. Le fournisseur doit avant tout préciser avec le client le contexte du projet et en définir le contenu. Il est nécessaire de bien définir l'alignement stratégique du projet et de dégager les différents processus « métier » cibles, ainsi que les règles « métier » associées.

Suite à cette phase de pré étude, les dirigeants d'une entreprise décident ou non de lancer le projet.

Dans le cas d'une décision favorable, il faut conduire le projet. Il est alors nécessaire de mettre en place une organisation et une méthodologie de gestion de projet. Les grandes lignes de cette méthodologie consistent à analyser les besoins, évaluer les charges, définir et suivre les différents incréments à mettre en place, identifier et gérer les risques, définir l'architecture technique et applicative, gérer au mieux la réutilisation, et maîtriser les ressources humaines ainsi que les ressources logicielles.

Il existe plusieurs types de projet : les projets de développement, les projets d'intégration et les projets de déploiement. Certains projets peuvent appartenir à plusieurs de ces types.

C'est entre autres le cas du projet merchandising. Dans ce projet, il a été nécessaire de mettre en place une solution spécifique au travers de développement, d'intégrer un progiciel dans le Système d'Information de l'entreprise et de déployer ces solutions au sein de l'entreprise.

Il est crucial que le projet ait un objectif clairement défini, limité et unique. En général, un projet est caractérisé par un niveau de complexité important. On peut caractériser cette complexité en identifiant les charges globales, les savoir-faire requis et les risques encourus.

Les limites du projet doivent être définies de manière précise en terme de délais, de moyens financiers, de ressources et de personnels.

La gestion d'un projet Informatique commence par la précision du contexte, des acteurs, de leurs interactions et l'identification des processus « métier ». On s'intéresse tout d'abord à définir la démarche projet. Le projet suit un cycle de vie. Des modèles et méthodes sont utilisés pour décrire son contenu.

Le cycle de vie comporte plusieurs étapes :

- Pré étude
- Evaluation des besoins et des charges
  Décomposition en tâches et détermination de lots.

- Budgétisation
  Associer des coûts aux tâches. Déterminer des rubriques budgétaires globales et des réserves.
- Rédaction de l'offre et du contrat
- Kick Off
  Démarrage officiel du projet pour canaliser l'énergie des différents acteurs.
- Planification
  Détermination d'un échéancier et du calendrier.
- Conception
- Implémentation et tests
- Recette
- Mise en production
- Achèvement
- Maintenance

On peut différencier les cycles de vie correspondant à des développements spécifiques ou à des développements de progiciel.

Ces deux types de développements se différencient essentiellement sur la phase de développement. Les phases de définition, d'exploitation, d'utilisation, de maintenance et d'évolution sont gérées de manière identique. Dans le premier cas, il s'agit de mener une étude préalable, suivie d'une étude détaillée, d'une étude technique de mise en œuvre et de la mise en œuvre à proprement parler. Dans le second cas, il s'agit de choisir un progiciel, d'effectuer le paramétrage afin de répondre au besoin, de tester les adaptations effectuées.

D’autre part, dans le cas du développement d’un logiciel spécifique au sein d’une entreprise, la phase de « rédaction de l’offre et du contrat » est moins formelle.

2.2. Les modèles et les méthodes

Il existe plusieurs modèles de déroulements type de projet. Le déroulement d’un projet dépend de sa nature et de son contexte. Il n’y a pas de méthode universelle qui puisse être appliquée à tous les projets. On peut différencier deux types de méthodes de gestion de projets. Les méthodes traditionnelles suivent des cycles séquentiels et sont basées sur des contrats. Les méthodes plus modernes positionnent l’utilisateur comme un acteur clef du projet et préfèrent satisfaire ce dernier plutôt qu’honorer les clauses d’un contrat.

Ci-dessous, un diagramme illustrant le déroulement type d’un projet traditionnel [MAN06].

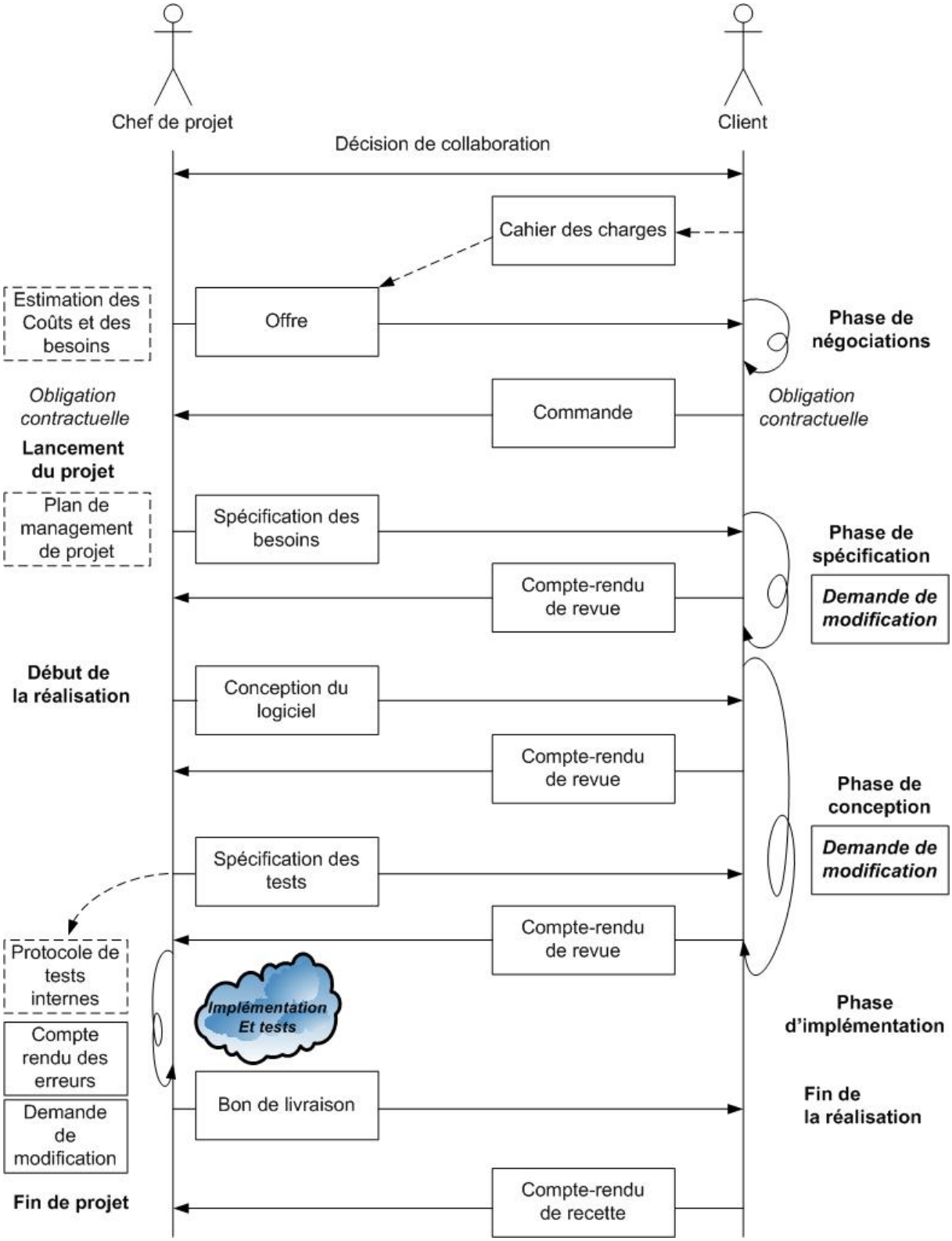


Figure 11 : Déroulement type d'un projet traditionnel

Cycles de projet

Lors de la réalisation d'un logiciel, plusieurs types de cycles de développement sont possibles. Les grandes familles sont les cycles en cascade, les cycles en V et les cycles itératifs.

Cascade

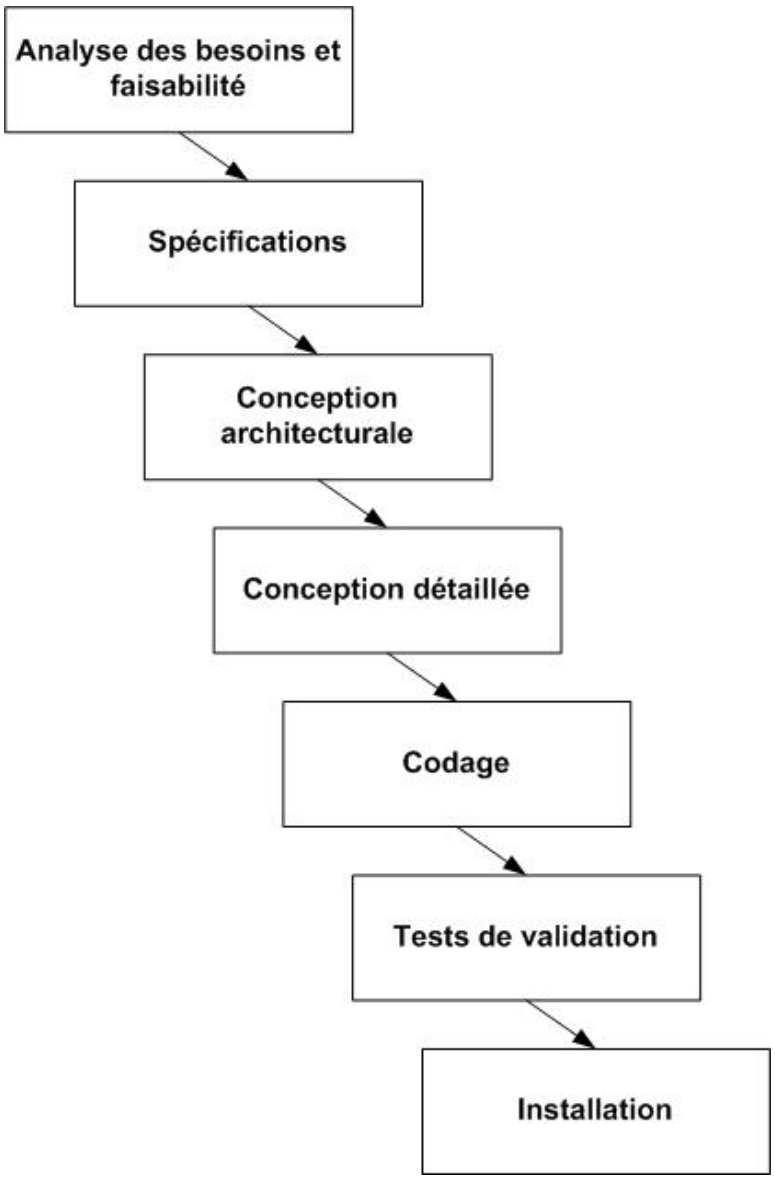


Figure 12 : Cycle de vie en cascade

Ce cycle de vie en cascade (*Figure 12*) est hérité du bâtiment. Les phases traditionnelles de développement sont effectuées les unes après les autres, avec un retour sur les précédentes, voire au tout début du cycle. Chacune des phases produit des livrables définis au préalable et se termine à une date précise. Une étape de validation et de vérification des livrables permet de terminer une phase.

V

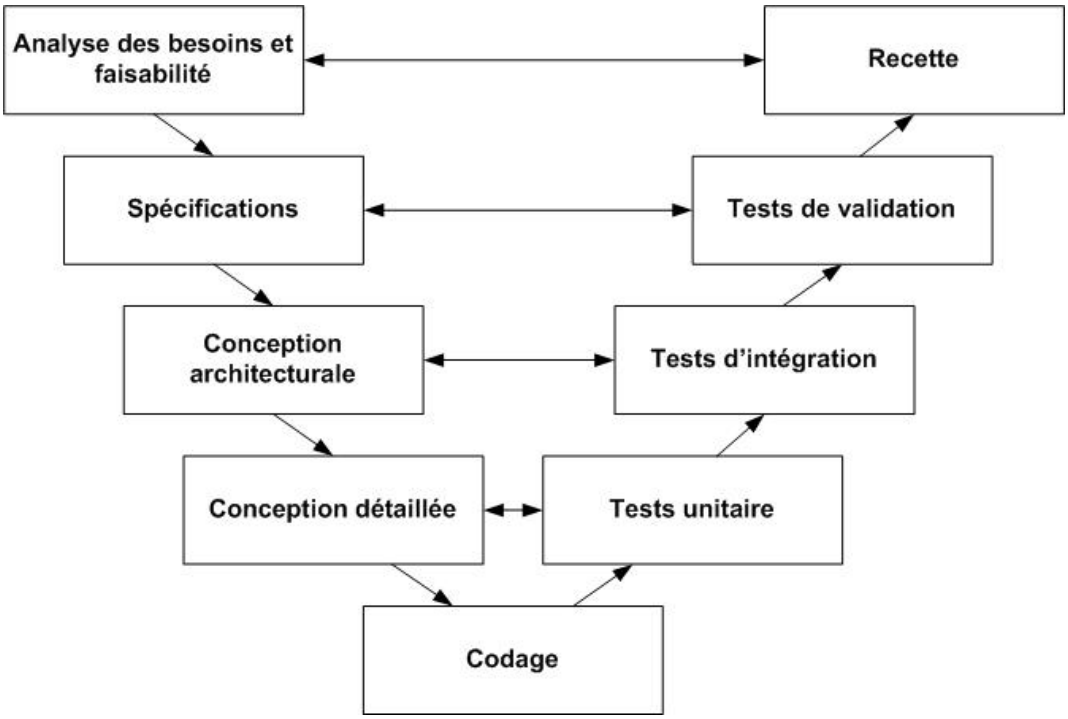


Figure 13 : Cycle de vie en V

Depuis les années 1980, le cycle en V (*Figure 13*) est devenu un standard de l'industrie du développement de logiciel et de la gestion de projet.

Ce cycle limite les retours aux étapes précédentes afin de pallier le problème de réactivité du modèle en cascade. Les phases de la partie montante, renvoient de l'information sur les phases en vis-à-vis. Les livrables des étapes montantes sont préparés dans les étapes descendantes. Par exemple, les livrables des tests de validation sont définis lors des spécifications.

Itératif

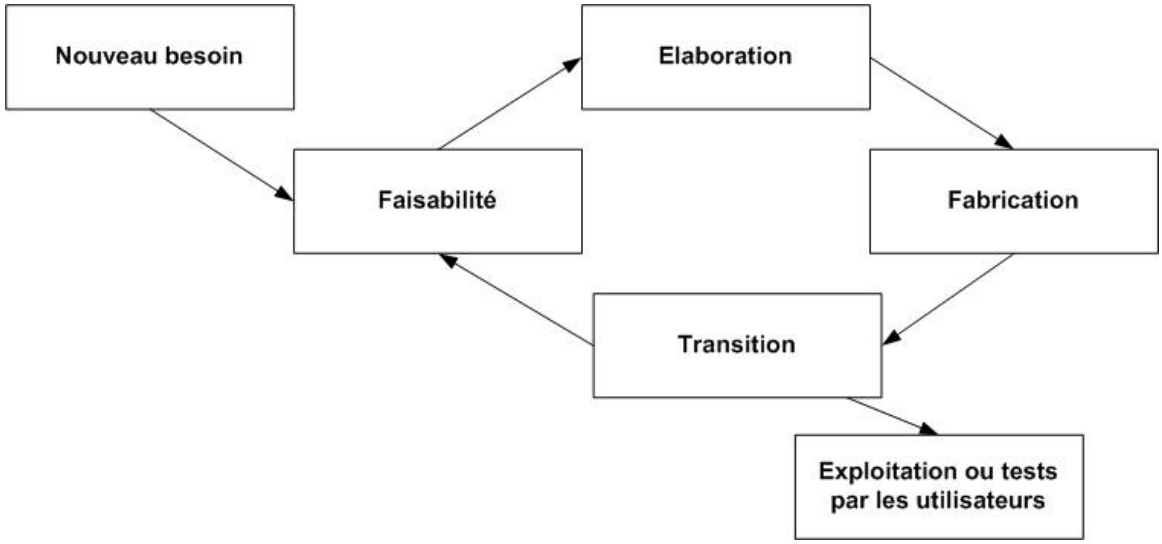


Figure 14 : Cycle de vie itératif

Le produit issu d'une activité est appelé un artéfact. On distingue les activités des artéfacts. Un cycle de type roue de Deming est appliqué sur la production des livrables : documentation, code, test.

Dans le cas d'une gestion de projet, on étudie la faisabilité d'un nouveau besoin. On élabore une solution. Une phase de fabrication permet ensuite de construire cette solution. La livraison au client correspond à la transition.

Dans un cycle itératif (*Figure 14* ci dessus), les phases de faisabilité, d'élaboration, de fabrication et de transition correspondent respectivement aux phases de spécifications, de détermination de l'architecture, de développement et de tests. L'objectif est d'effectuer au plus tôt les livraisons au client afin qu'il puisse effectuer une recette.

Une itération est plus courte et régulière qu'une roue de Deming (PDCA) qui, appliquée à une organisation importante, peut prendre plusieurs années.

Comparaison

Le cycle en V a pour origine l'industrie lourde. Les phases de validation sont importantes car les phases successives du projet sont de plus en plus lourdes. Dans le cas de gros projets réunissant un nombre important de personnes, les décisions de la direction ou des architectes ont tellement d'impact en terme de charge et de durée qu'il est important de s'assurer de la validité de chacune des étapes.

Dans le cas d'un projet logiciel impliquant peu de personnes sur des durées courtes de l'ordre d'une à deux années, on dispose d'une plus grande réactivité du fait de la proximité géographique et d'une communication facilitée compte tenu de la taille de l'équipe. De plus, les coûts sont plus limités entre chaque étape. Dans ce contexte, il est possible d'utiliser des méthodes de développement dites agiles en diminuant le formalisme et en multipliant le nombre de cycles.

Tests et validation des logiciels

Les tests sont une activité centrale. Ils permettent d'améliorer la qualité des logiciels. Le maître d'œuvre d'un projet peut mettre une stratégie de test en place qui se compose de tests unitaires, tests d'intégration, de vérification et de validation.

La validation met en évidence les défauts d'un logiciel et sa conformité par rapport à ses spécifications. Cette étape permet de gérer les anomalies, d'évaluer la conformité fonctionnelle et la fiabilité du logiciel.

Il existe deux grandes familles de tests. Les tests fonctionnels qui permettent d'évaluer l'ergonomique, la robustesse, la réponse au stress ou le niveau de performance d'un logiciel et les tests structurels qui permettent des analyses de la couverture du code, des analyses statistiques ou des analyses de complexités. Ces différents tests peuvent être automatisés. Des outils permettent l'automatisation de tests, la capture et le rejeu (Opensta, Selenium), ou encore l'analyse de couverture (RCOV pour ruby).

D'autre part, des techniques de vérification telles que les revues peuvent être effectuées par le client. A l'aide de ces revues, les dérapages sont limités, voire évités. Les coûts sont réduits par la correction des problèmes des résultats intermédiaires. Le résultat obtenu correspond d'avantage au résultat attendu. De plus, ces revues permettent de partager les responsabilités du projet entre le client et le fournisseur.

La réalisation de test permet de détecter les erreurs. Mais cette détection doit être réalisée au plus tôt. Plus une erreur est détectée tôt moins elle est coûteuse à corriger. Le coût de la correction d'une erreur au niveau de l'analyse est beaucoup moins important que le coût de la correction d'une erreur au niveau de la conception. Si la correction de cette erreur s'effectue dans les phases d'implémentation ou d'exploitation, les coûts de correction peuvent être 1000 fois plus importants.

Méthode Agile : panorama des principales méthodes

Une méthode agile est une méthode de développement informatique [BEN05]. Le client est impliqué un maximum dans le projet. La réactivité aux demandes est donc très bonne. Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles. On cherche à satisfaire un client plutôt que les clauses d'un contrat.

La méthode recherche la production d'un code de qualité. Les développements sont guidés par les tests.

L'objectif de ces méthodes est de permettre de mieux maîtriser les délais, les coûts et la production des projets informatiques.

Les développements sont effectués de façon itérative et incrémentale.

La méthode RAD est à l'origine des méthodes agiles. Les méthodes agiles sont le résultat de la recherche d'approches plus adaptées aux nouvelles technologies dans lesquels des cycles courts sont favorisés.

Les méthodes agiles prônent 4 valeurs fondamentales qui sont présentées dans le manifeste agile:

- L'équipe : « Personnes et interaction plutôt que processus et outils »
- L'application : « Logiciel fonctionnel plutôt que documentation complète »
- La collaboration : « Collaboration avec le client plutôt que négociation de contrat »
- L'acceptation du changement : « Réagir au changement plutôt que suivre un plan »

Les 4 valeurs se déclinent en 12 principes :

- « Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles ».
- « Le changement est bienvenu, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client ».
- « Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte ».
- « Les gens de l'art et les développeurs doivent collaborer quotidiennement au projet ».
- « Bâtissez le projet autour de personnes motivées. Donnez leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail ».
- « La méthode la plus efficace de transmettre l'information est une conversation en face à face ».
- « Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet ».
- « Les processus agiles promeuvent un rythme de développement soutenable. Commanditaires, développeurs et utilisateurs devraient pouvoir maintenir le rythme indéfiniment ».
- « Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité ».
- « La simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle ».
- « Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto organisent ».
- « À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accordé et ajuste son comportement dans ce sens ».

Un panorama des principales méthodes : RAD, Crystal Clear, Scrum, DSDM, XP, UP, RUP est présenté ci-dessous.

RAD

RAD est l'acronyme de Rapid Application Development. C'est une méthode de développement de logiciels dont le cycle de développement est très court. Elle a été développée par James Martin dans les années 1980.

La réalisation et les tests de l'application sont effectués par morceau à intervalles réguliers. Dans cette méthode, des environnements graphiques sont utilisés afin d'obtenir rapidement des prototypes.

Crystal clear

Crystal clear est une méthode de gestion de projet crée par Alistair Cockburn. Cette méthode est adaptable aux spécificités des projets. Elle repose sur un certain nombre de principes auquel doit adhérer l'ensemble de l'équipe :

- La communication a une place importante dans le projet afin de faire collaborer les différents acteurs du projet de manière efficace.
- Le nombre des membres d'une équipe est limité à six personnes pour améliorer la solidarité.
- l'équipe travaille dans une même pièce pour que la proximité facilite la communication.
- Les schémas de modélisation sont effectués en groupe sur tableau blanc pour une meilleure communication et une meilleure collaboration.
- La collaboration avec le client est très importante. Les discussions entre les utilisateurs et les développeurs sont nombreuses.
- Des parties exécutables de l'application sont livrées fréquemment. Le client peut ainsi se rendre compte de l'avancement du projet et faire des retours sur les livraisons.

La méthode Crystal clear reste très souple au niveau des procédures à suivre et des normes à utiliser. La procédure est découpée en différentes étapes :

- Les utilisateurs sont observés dans leur travail afin de mieux comprendre leurs besoins et leur contexte. Les fonctionnalités sont classées par ordre de priorité en collaboration avec les utilisateurs. Les fonctionnalités avec la plus haute priorité sont développées en premier.
- Au début du projet, une ébauche de conception et une ébauche d'architecture sont réalisées.
- On planifie les dates des itérations. Des livrables fonctionnelles sont définies à la fin de chacune d'elles.
- La réalisation proprement dite de l'application se fait durant les itérations.

La méthode Crystal clear présente tous les avantages des méthodes agiles : flexibilité par rapport au changement, rapidité, livraisons fréquentes. Elle convient aux petites structures. Elle est très efficace dans les projets de petite taille mais n'est pas adéquate pour des projets importants.

Scrum

Scrum est une méthode agile pour la gestion de projets. Elle a été conçue pour améliorer la productivité en évitant de paralyser les équipes par l'emploi de méthodologies trop lourdes.

Ken Schwaber et Jeff Sutherland ont mis au point les grands principes de Scrum au début des années 1990.

Le terme *Scrum* est emprunté au rugby et signifie *mêlée*. L'utilisation de ce terme est une analogie au rugby du fait que le processus s'articule autour d'une équipe soudée, qui cherche à atteindre un but.

Dans cette méthode, on focalise l'équipe de façon itérative sur un ensemble de fonctionnalités. L'équipe doit réaliser ces fonctionnalités durant des itérations de l'ordre de 30 jours appelées Sprints.

Un but à atteindre est défini dans chaque Sprint. On détermine un ensemble de fonctionnalités à implémenter. Le sprint aboutit à la livraison d'un produit proposant les nouvelles fonctionnalités.

Un *ScrumMaster* a pour rôle de réduire les perturbations extérieures et de résoudre les problématiques non techniques de l'équipe.

Le client participe de façon active. Il priorise la réalisation des fonctionnalités du logiciel. Il a la possibilité de changer la liste des fonctionnalités désirées à condition qu'elles ne soient pas en cours de réalisation.

Dynamic Systems Development Method

Dynamic Systems Development Method (DSDM) est une méthode de gestion de projet de la catégorie des méthodes agiles. Cette méthode a été développée en Grande-Bretagne à partir de 1994

La méthode DSDM s'appuie sur 9 principes de base :

- Implication des utilisateurs durant tout le cycle de développement. Ils font partie de l'équipe projet.
- Autonomie. L'évolution des besoins peut être influencée par l'équipe projet.
- Visibilité du résultat. Un feed-back rapide est possible par des livraisons fréquentes.
- Adéquation. On cherche à livrer une application en adéquation avec le besoin « métier » du client.
- Développement itératif et incrémental. L'évolution du développement est basée sur le feed-back des utilisateurs.
- Réversibilité. Toute modification effectuée durant le développement doit être réversible.
- Synthèse. Un schéma directeur définit le périmètre et les grandes lignes du projet.
- Tests. Les tests sont effectués en continu, en parallèle du développement.

- Coopération. Il est nécessaire d'être souple par rapport aux modifications des fonctionnalités demandées.

Extreme programming

L'Extreme Programming (XP) est une méthode agile de gestion de projet informatique [BEN05]. Elle est adaptée aux équipes de petite taille qui sont confrontées à des besoins changeants. Des principes simples sont poussés à l'extrême.

L'Extreme Programming a été inventée par Kent Beck, Ward Cunningham et Ron Jeffries pendant un projet "C3" de calcul des rémunérations chez Chrysler entre 1996 et 1999.

Le but principal est de réduire les coûts liés au changement. Dans les méthodes traditionnelles, les besoins sont définis et fixés, au départ du projet. Les coûts ultérieurs de modifications s'en trouvent accrus. Cette méthode fait en sorte de rendre le projet plus flexible et ouvert au changement. Elle introduit des valeurs de base et des principes. Elle repose sur un certain nombre de pratiques.

Elle utilise des principes qui ne sont pas nouveaux mais elle les pousse à l'extrême :

- la revue de code est permanente et effectuée par binôme
- les tests sont systématiques et effectués avant chaque implémentation
- la conception se fait tout au long du projet (refactoring)
- on choisit toujours la solution la plus simple
- l'emploi de métaphores permet d'améliorer la compréhension
- l'intégration des modifications s'effectue plusieurs fois par jour
- pour s'adapter au changement, les cycles de développement sont très rapides.

Les cycles de développement rapides impliquent des itérations très courtes qui peuvent durer moins d'une semaine. Une itération se compose d'un ensemble d'étapes qui permettent de :

- déterminer les scénarios clients de l'itération
- transformer les scénarios en tâches à réaliser et de déterminer les tests fonctionnels correspondants
- réaliser les tâches en binôme
- valider et livrer les fonctionnalités développées au client

Tant que le client fournit des scénarios, le cycle se répète. La première livraison est en générale la plus longue à réaliser mais peut aboutir après quelques itérations. Après avoir mis en production, les itérations peuvent être encore plus fréquentes.

UP

Processus Unifié (PU) ou en anglais Unified Process (UP) est une méthode de prise en charge du cycle de vie des logiciels orientés objets. Elle est générique, itérative et incrémentale, contrairement à Merise ou SADT qui sont des méthodes séquentielles.

RUP, proposé par la société Rational, est l'une des plus célèbres implémentations de la méthode PU. C'est une solution livrée clés en main qui permet d'avoir un cadre de développement logiciel. Cette méthode est guidée par les besoins des utilisateurs et centrée sur l'architecture logicielle.

Les processus unifiés utilisent UML. UML est un langage de modélisation. Mais UML ne prend pas en charge le cycle de vie du logiciel, le processus de création et de conception des modèles. La prise en compte de la diversité des projets, des problématiques, des équipes et des cultures d'entreprise dans une seule et unique méthode n'étant pas aisée, PU tente de couvrir cette lacune. PU est générique et possède de nombreux avatars afin de répondre à la diversité des situations :

- RUP : Rational Unified Process, processus défini par la société Rational Software (IBM)
- EUP : Enterprise Unified Process, processus intégrant les phases de post-implantation et décrivant le cycle de vie du logiciel
- XUP : Extreme Unified Process, processus intégrant UP avec l'extreme programming.
- AUP : Agile Unified Process, processus mettant l'accent sur l'agilité des développements. Il s'appuye plus sur l'optimisation et l'efficience sur le terrain que sur la modélisation
- 2TUP : Two Tracks Unified Process, processus proposé par Valtech prenant en compte les aléas et contraintes liés aux changements perpétuels et rapides des SI des entreprises.
- EssUP : Essential Unified Process, par Ivar Jacobson, à l'initiative d'UML et RUP. Ce processus unifié intègre certains concepts des méthodes Agile.

RUP

RUP repose sur 6 principes :

- développement itératif
- traitement et formalisation des exigences
- architecture à base de composant
- modélisation visuelle
- vérification de la qualité du logiciel
- gestion du changement

RUP est caractérisé par un découpage temporel en phases et itérations, l'utilisation d'un support entièrement informatique et un aspect générique et paramétrable.

UML

UML signifie Langage de Modélisation Unifié (Unified Modeling Language) [ROQ00] [ROQ07]. C'est un langage graphique de modélisation. Il représente une formalisation très aboutie et non propriétaire de la modélisation objet utilisée en génie logiciel. UML n'est pas une méthode.

UML est issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson. Il est né de la fusion des langages de modélisation objet Booch, OMT, OOSE. UML est un standard défini par l'OMG (Object Management Group).

Le modèle UML 2 est composé de 13 types de diagrammes alors que UML 1.3 n'en comportait que 9.

UML se décompose en plusieurs sous-ensembles : les vues, les diagrammes et les modèles d'éléments.

Les vues décrivent le système d'un point de vue donné (organisationnel, dynamique, temporel, architectural, géographique, logique …). Les vues sont des notions abstraites. La combinaison de ces vues définit le système complet.

Les diagrammes décrivent le contenu des vues à l'aide d'éléments graphiques. Les diagrammes peuvent faire partie de plusieurs vues.

Les modèles d'éléments tels que les cas d'utilisation, les classes ou les associations permettent d'élaborer les diagrammes UML.

Une façon de mettre en oeuvre UML est de considérer différentes vues superposables pour collaborer à la définition du système : vue d'implémentation, vue logique, vue des cas d'utilisation, vue de déploiement et vue des processus (Voir *Figure 15*).

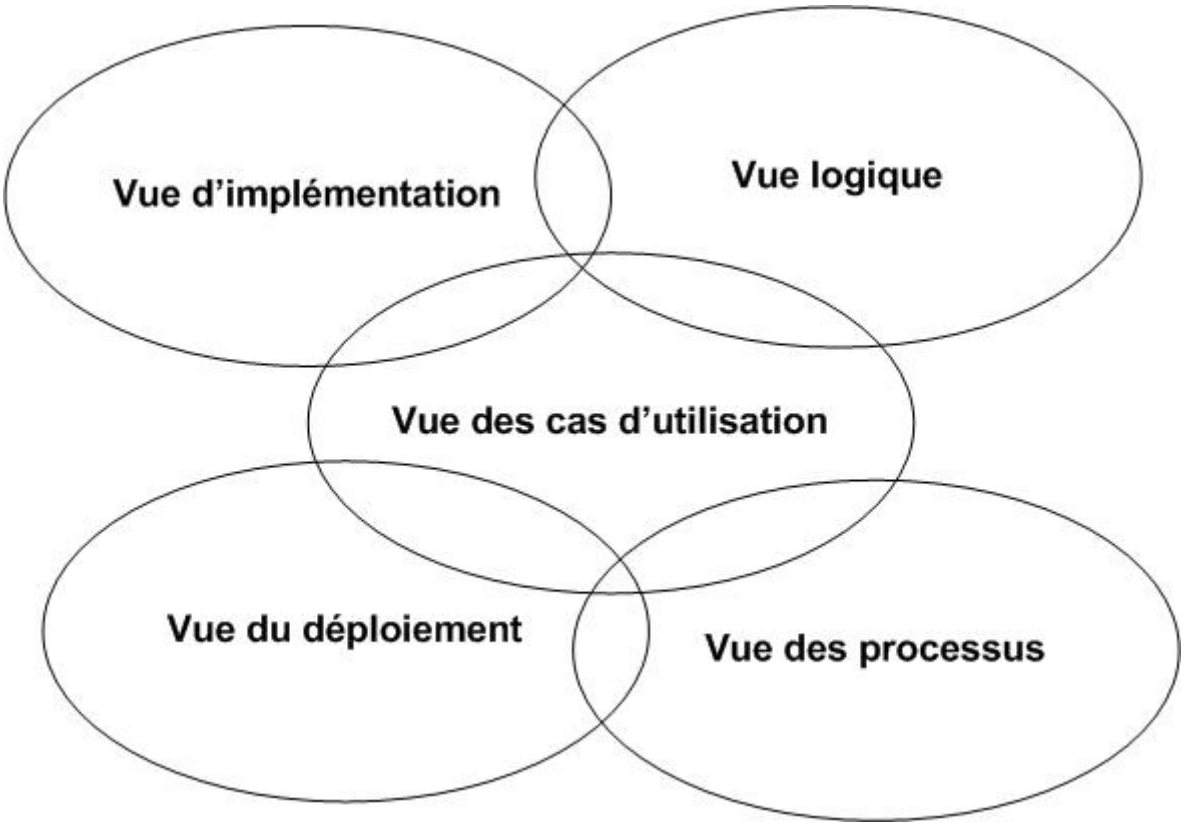


Figure 15 : Les différentes vues dans UML

- Vue des cas d'utilisation :

Elle représente le point de vue des acteurs du système et correspond aux besoins.

- Vue logique :

On définit le système de l'intérieur. Elle explique la manière de satisfaire les besoins utilisateurs.

- Vue d'implémentation :

Elle permet de définir les dépendances entre les modules.

- Vue des processus :

Elle met en œuvre les notions de tâches concurrentes, stimuli, contrôles et synchronisations. Elle couvre des aspects temporels et techniques.

- Vue de déploiement :

Elle décrit la position géographique et l'architecture physique de chacun des éléments du système.

Ces différents points de vues permettent de répondre aux questions suivantes : Quoi ? Qui ? Comment ? Où ? Seul le pourquoi n'est pas défini à l'aide d'UML.

Les 13 diagrammes UML sont dépendants hiérarchiquement et se complètent (voir *Figure 16* et *Tableau 1*). Ci-dessous est présentée la hiérarchie des diagrammes UML 2.0 sous forme d'un diagramme de classes.

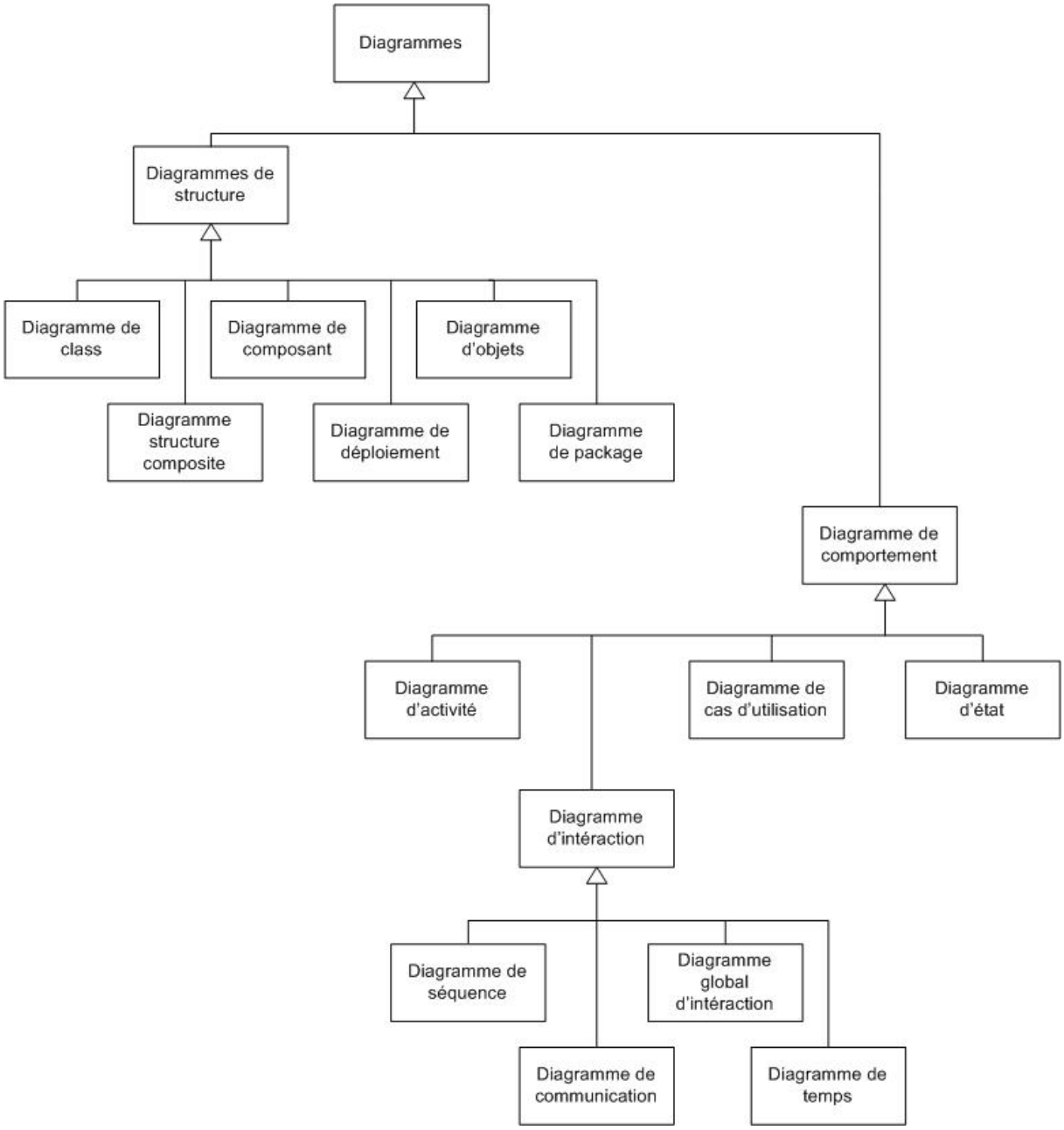


Figure 16 : La hiérarchie des diagrammes UML 2.0

Tableau 1 : Diagrammes UML 2.0

Nom du diagramme	**Description**
Diagrammes Structurels ou Diagrammes statiques (Structure Diagram)	
Diagramme de classes Class diagram	Il représente les classes intervenant dans le système.
Diagramme d'objets Object diagram	Il sert à représenter les objets utilisés dans le système.
Diagramme de composants Component diagram	Il décrit le point de vue physique des composants du système. Il présente, en particulier, leur mise en œuvre (fichiers, bibliothèques, bases de données...)
Diagramme de déploiement Deployment diagram	Il représente les éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...), la répartition des composants du système sur ces éléments matériels et la façon dont ils interagissent.
Diagramme des paquetages Package Diagram	Il permet de représenter les dépendances entre regroupement d'éléments, ainsi qu'entre packages.
Diagramme de structure composite Composite Structure Diagram	Les relations entre composants d'une classe sont décrites sous forme de boîtes blanches.
Diagrammes Comportementaux ou Diagrammes dynamiques (Behavior Diagram)	
Diagramme des cas d'utilisation Use case diagram	Il identifie les possibilités d'interaction entre le système et les acteurs. Les fonctionnalités que doit fournir le système sont alors précisées.
Diagramme états transitions State Machine Diagram	Le comportement du système ou de ses composants est décrit sous forme de machine à états finis.
Diagramme d'activité Activity Diagram	Le comportement du système ou de ses composants est décrit sous forme de flux ou d'enchaînements d'activités.
Diagramme d'interactions (Interaction Diagram)	
Diagramme de séquence Sequence Diagram	Le déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs est représenté de manière séquentielle.
Diagramme de communication Communication Diagram	C'est un diagramme de séquence simplifié qui se concentre sur les échanges de messages entre les objets.
Diagramme global d'interaction Interaction Overview Diagram	Ce diagramme décrit les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences. Un diagramme de séquence est une variante du diagramme d'activité.
Diagramme de temps Timing Diagram	Il décrit les variations d'une donnée au cours du temps.

Ci-dessous un tableau récapitulant les différents diagrammes utilisés en fonction des vues.

Tableau 2 : Diagrammes en fonction des vues

Vues	**diagrammes**
Vue logique	Diagramme de classes et d'objets
Vue d'implémentation	Diagramme de composants
Vue cas d'utilisation	Diagramme de cas d'utilisation
Vue processus	Diagramme d'activité, séquence, collaboration
Vue déploiement	Diagramme de déploiement

Le langage UML 2.0 a été lancé en 2003 [BOR04]. Il présente une avancée très intéressante du support de communication. On peut utiliser UML 2 comme on utilisait UML 1.3.

UML 2 est très influencé par les processus MDA (Model Driven Architecture) et MDD (Model Driven Development). L'un des objectifs d'UML 2 est de parvenir à une automatisation du développement afin de passer directement du modèle au code source et donc de fournir un programme compilé, prêt à être lancé. C'est pourquoi les sémantiques des modèles UML ont été précisées afin d'éviter des problèmes d'interprétation. Selon le niveau d'abstraction, sans pour autant devoir programmer, les modèles se rapprochent des programmes eux-mêmes. Un autre objectif serait de générer également de façon automatique les tests cases qui accompagnent le programme.

Le niveau d'abstraction du langage est donc plus élevé que dans la version précédente. UML 2 bénéficie d'un langage enrichi qui a été modularisé en différents sous langages qui peuvent être combinés.

La nouvelle version d'UML ajoute 4 autres diagrammes : le diagramme des paquetages (package diagram), le diagramme de structure composite (composite structure diagram), le diagramme global d'interaction (interaction overview) et le diagramme de temps (timing diagram).

D'autre part, le diagramme de collaboration d'UML 1.4 est devenu diagramme de communication dans UML 2.0 et la plupart des diagrammes ont été revus pour répondre aux nouveaux besoins d'abstraction et d'automatisation...

Processus 2TUP

2TUP signifie 2 Track Unified Process [ROQ00] [ROQ07]. C'est un processus de développement logiciel qui implémente le Processus Unifié.

Le 2TUP repose sur un cycle de développement en Y (*Figure 17*) permettant de dissocier les aspects techniques des aspects fonctionnels. Il commence par une étude préliminaire. Cette étude préliminaire permet d'identifier les acteurs qui vont interagir avec le système à construire, d'identifier les messages qu'échangent les acteurs et le système, de produire le cahier des charges et de modéliser le contexte. Ensuite, le processus s'articule

autour de 3 phases essentielles : une phase technique (branche de droite), une phase fonctionnelle (branche de gauche) et une phase de réalisation.

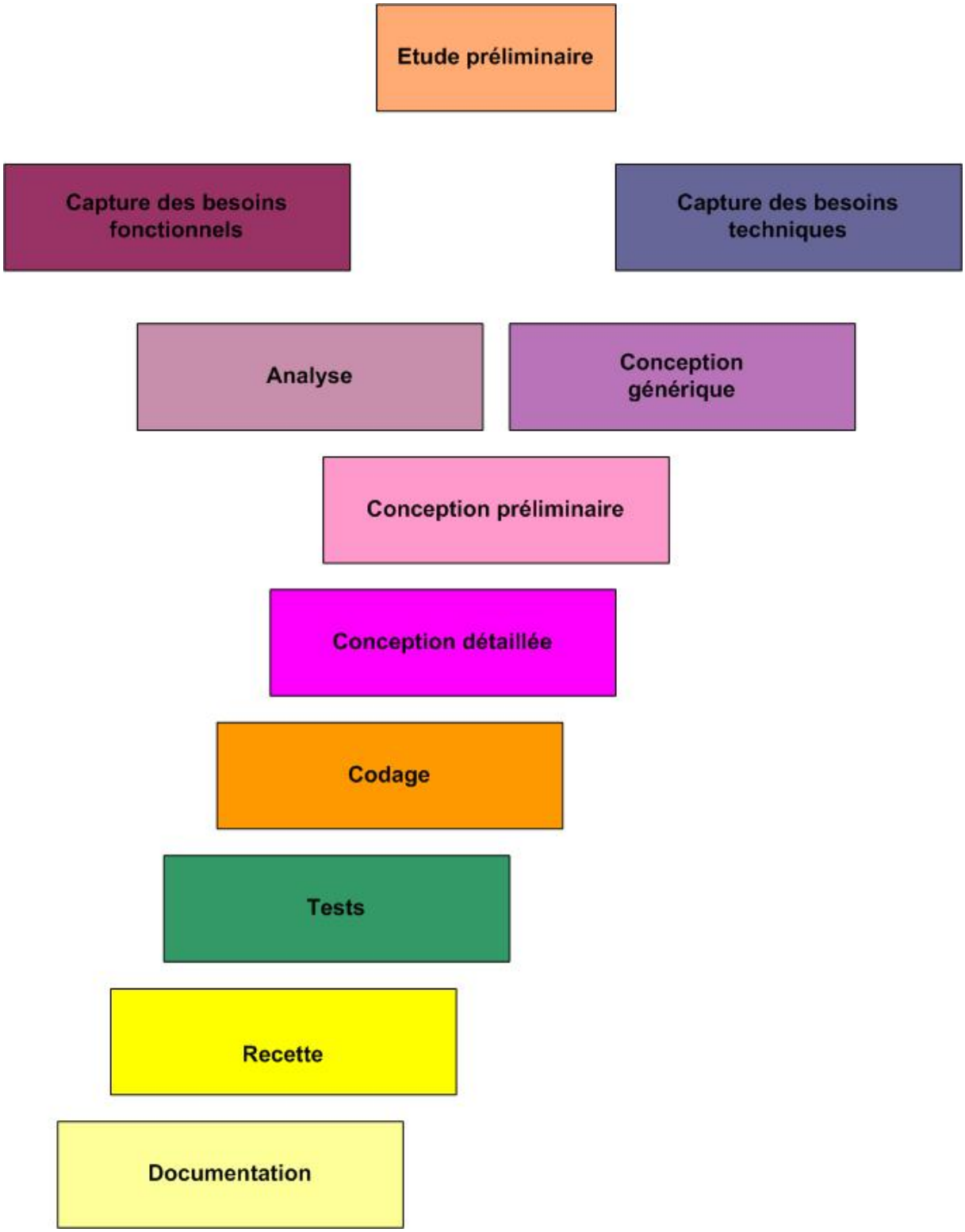


Figure 17 : Schéma en Y

Le processus 2TUP repose sur une démarche itérative et incrémentale pour diminuer les risques et organiser la production des livrables. Il sépare les préoccupations fonctionnelles et techniques.

Le processus 2TUP permet le développement de progiciels objet. C'est pour cette raison que l'analyse est très centrée sur la détermination de classes et que la modélisation objet est très présente dans chacune des étapes du processus (*Tableau 3*).

Tableau 3 : Description des différentes étapes du processus

Phase	**Description**
Etude préliminaire Capture initiale des besoins	Effectuer un recueil initial des besoins fonctionnels et techniques Préciser le contexte, les acteurs, les interactions
Capture des besoins fonctionnels	Compléter le recueil des besoins Rechercher les classes candidates Produire un modèle des besoins Qualifier les risques de non réponse aux besoins
Capture des besoins techniques	Déterminer les contraintes et les choix dimensionnant le système Prendre en compte des contraintes d'intégration Prendre en compte des contraintes techniques et logicielles Déterminer la configuration matérielle Sélectionner les outils et les matériels Définir le style d'architecture en tiers Etablir la spécification logicielle (gestion des erreurs, sécurité, distribution, intégrité, utilisation de l'aide) Définir les couches logicielles (modèle en 5 couches et couche de synchronisation avec le SI)
Analyse	Etudier les spécifications fonctionnelles Veiller à ce que le résultat de l'analyse ne dépende d'aucune technologie Définir le modèle structurel d'analyse : Découper les classes en catégories (cohérence, indépendance) Définir le modèle statique d'analyse : Détailler, compléter, optimiser les diagrammes de classes Définir le modèle dynamique : Déterminer la collaboration entre objets (diagramme interaction et diagramme d'état) Itérer sur les deux derniers modèles
Conception générique	Définir des composants nécessaires à l'architecture technique Uniformiser et réutiliser les mêmes mécanismes pour tout le système Construire le squelette Ecarter les risques Déterminer les Frameworks et les Design patterns Organiser l'architecture technique et les composants Veiller à répondre à des objectifs de réutilisation, de fabrication et de déploiement

	Déterminer les couches, le modèle logique et le modèle d'exploitation Valider avec un prototype
Conception préliminaire	Intégration de l'analyse et de la conception générique Cartographie des composants Déploiement du poste et des composants d'exploitation (composants distribués, applications, base de données) Interface EAI, intégration de progiciels IHM : vision précise des couches présentation et application Compléter la configuration logicielle Déterminer les modules
Conception détaillée	Etudier comment réaliser chaque composant Concevoir et documenter le code qui va être produit Processus de construction itératif qui s'applique successivement aux couches logicielles
Codage	Produire les composants
Tests	Tester les unités de code Effectuer des tests d'intégration pour tester les IHM
Recette	Effectuer des tests de recette pour valider les fonctions du système développé
Documentation	Réaliser la documentation à destination des différents acteurs : utilisateurs, administrateurs et personnes en charge du suivi d'exploitation…

Diagramme UML dans 2TUP

UML est utilisé pour modéliser le système. Ci-dessous, les diagrammes utilisés durant les différentes phases du processus.

Tableau 4 : Les diagrammes utilisés durant les différentes phases du processus.

Phases Diagrammes	Capture initiale des besoins	Capture des besoins techniques	Capture des besoins fonctionnels	Analyse	Conception générique	Conception préliminaire	Conception détaillée
Classes		X	X	**X**	X	X	X
Package		X	X		**X**	X	X
Objets						**X**	**X**
Structure composite						X	X
Cas d'utilisation		**X**	**X**	X	X		
Séquence		X	X	**X**		**X**	**X**
Collaboration	X	X	X			X	X
Etats				X		X	X
De temps						X	X
D'activité		**X**	**X**			X	X
Global d'interaction		X	X	X		X	X
De composants		X			**X**		X
De déploiement		X			**X**		

XP

L'eXtreme Programming est une alternative à des méthodes plus lourdes dans le cas de projets de taille moyenne [BEN05]. Elle regroupe des bonnes pratiques de développement qui visent à améliorer la qualité des logiciels. Elle repose sur un processus projet en continu. Les phases de conception, de validation et d'intégration sont effectuées en continu. Des itérations sont effectuées au niveau du développement et des livraisons. Le code est sans cesse amélioré par la réécriture. Elle repose sur une rétroaction constante, le pilotage par les tests, une planification par les scénarios clients et l'intégration du client. Elle recherche une conception simple, s'appuie sur des conventions d'écriture et le code produit est une copropriété des membres de l'équipe. Les tests fonctionnels expriment les besoins et permettent d'effectuer la recette de l'application. Cette méthode recherche une meilleure maîtrise de l'environnement en s'inspirant du concept de cible mouvante (moving target).

L'eXtreme Programming repose sur un ensemble de pratiques permettant de réaliser un logiciel, de la programmation à la planification. Cette méthode permet d'organiser l'équipe et de gérer les relations avec le client.

Elle est basée sur des principes qui ne sont pas nouveaux : livraisons fréquentes, relecture de code, tests automatiques.

Mais dans cette méthode, ces pratiques sont poussées à l'extrême.

L'équipe se focalise sur la réalisation et ne perd pas son temps dans des activités imposées consommatrices de temps et de ressources comme la production de documents non exploités.

Le contact humain est positionné au premier rang, dans l'équipe et avec le client.

Il n'existe que très peu de projets pour lesquels les spécifications sont complètes et ne changent pas. Dans cette méthode, on n'attend pas d'avoir l'exhaustivité des spécifications pour commencer à réaliser l'application.

Les modifications des spécifications du client sont acceptées et prises en compte tout au long du projet afin de répondre à ses besoins.

XP définit 13 pratiques classées en 3 catégories :

- les pratiques de programmation
  - conception simple (simple design)
  - remaniement (refactoring)
  - développement piloté par les tests unitaires (unit tests)
  - tests de recette (customer tests)
- les pratiques de collaboration
  - programmation en binômes (pair programming)
  - responsabilité collective du code (collective code)
  - règles de codage (coding standards, ownership)
  - métaphore (metaphor)
  - intégration continue (continuous integration)
- les pratiques de gestion de projet
  - livraisons fréquentes (frequent releases)
  - planification itérative (planning game : planification effectuée par le client et l'équipe au cours de séances tout au long du projet)
  - client sur site (on-site customer, whole team)
  - rythme durable (sustainable pace)

Ces pratiques, plutôt d'ordre technique, sont destinées à mettre en place un environnement de travail basé sur les valeurs suivantes :

- la communication pour une meilleure visibilité
  la communication directe est favorisée par rapport à l'écrit car elle permet un échange d'informations très important en un minimum de temps.
  La documentation n'est cependant pas supprimée, elle est gérée comme un besoin comme un autre. Les différents problèmes rencontrés sont résolus par l'ensemble de l'équipe client et fournisseur.
- la simplicité comme garantie de productivité
- le feedback comme outil de réduction des risques
  L'objectif est d'améliorer continuellement le pilotage du projet.
- le courage de prendre de bonnes décisions

Une équipe XP doit remettre sans cesse en question ses méthodes de travail plutôt que de suivre des instructions à la lettre. Les pratiques XP doivent permettre l'amélioration de l'efficacité de l'équipe dans le respect des 4 valeurs d'XP.

Compromis coût, délais, qualité, contenu.

Un projet dépend de 4 variables interdépendantes : coût, délais, qualité, contenu [BEN05].

- Coût

Le fait de changer l'équipe ou l'environnement de travail peut à court terme ralentir l'équipe. La plupart du temps le coût d'un projet est fixé préalablement au lancement du projet, lors de la demande des budgets. Si le coût fixé est sous dimensionné, il y a de fortes chances que la qualité du logiciel en pâtisse. Les coûts de maintenance et d'indisponibilité, quant à eux, risquent d'exploser.

Dans la plupart des cas, la sous estimation du coût d'un projet peut conduire à une dérive conséquente des coûts. La détermination de coûts plus élevés dès le départ permet de limiter l'ampleur des dérapages.

A l'opposé, si le coût du projet est sur dimensionné, on peut être amené à mettre en place une équipe trop importante qui sera difficile à coordonner. Et contrairement à ce que l'on espérait, la qualité de services sera dégradée et la durée du projet allongée.

- Délais

Le fait de repousser les dates de livraison peut induire des coûts de synchronisation et la confiance du client et de l'équipe peut s'en trouver diminuée.

D'autre part, imposer des délais trop courts a pour conséquence de mettre la pression à l'équipe. Le risque est alors un manque de recul des membres de l'équipe et une baisse de leur niveau d'exigence.

- Qualité

Une diminution du niveau de la qualité peut provoquer une démotivation de l'équipe ou une perte de la confiance du client qui finira par s'apercevoir de la dégradation du niveau

de qualité. La démotivation de l'équipe peut avoir pour conséquence une augmentation du turn-over.

- Contenu

Cette dernière variable permet d'être très flexible. Il est possible d'adapter le contenu d'un projet en jouant sur le périmètre fonctionnel. La réduction du périmètre fonctionnel permet de conduire à une réduction des coûts, de la durée tout en préservant un niveau de qualité acceptable. C'est sur cette dernière variable que joue la méthode XP.

Equilibre client et fournisseur

Un équilibre doit s'instaurer entre les développeurs et le client.

Le client ne doit pas avoir tous les pouvoirs sinon on tombe dans un projet « marche à mort » (deathmarch projet).

L'équipe ne doit pas avoir trop de pouvoir sinon le projet risque de ne pas répondre aux besoins du client en cherchant à utiliser à tout prix les dernières technologies à la mode.

Dans une organisation XP, les responsabilités sont clairement séparées :

- le client définit les fonctionnalités et leur ordre d'implémentation
- les développeurs estiment les coûts et prennent en charge la réalisation

La mise en œuvre de la méthode XP peut être de chercher à résoudre les problèmes les plus sérieux de manière itérative et de traiter les problèmes restants jusqu'à parvenir à une situation stable.

Influence de la culture d'entreprise

La culture générale de l'entreprise est un facteur important de la réussite d'un projet XP.

Il est difficile de tirer les bénéfices de la méthode XP dans le cas :

- où le mérite se mesure aux heures supplémentaires et où on considère qu'une équipe ne fonctionne à plein rendement que sous forte pression
- d'une culture centrée sur le jeu politique, où les relations de type gagnant / gagnant ne sont pas de mise
- d'une culture attachée aux démarches linéaires où la qualité du logiciel s'exprime en nombre de documents produits.

XP et le cycle en V

Un comparatif simple entre le cycle en V et XP est effectué dans le tableau ci-dessous (*Tableau 5*).

Tableau 5 : Comparatif simple entre V et XP

Points	**V**	**XP**
Traitement du problème	De front	En tranche
Spécification	Complète	Partielle. Ajustement des spécifications à partir du feedback du client.
Début de réalisation	A la fin de la conception	Tranche par tranche avec prise en compte du changement
Résultat de la conception	Document	Code
Fabrication du logiciel	Codage	Compilation
Organisation	Spécialisation des individus (spécification, conception, codage et tests)	Polyvalence des individus. Chaque membre de l'équipe est responsable de l'ensemble de l'activité.
Moyen de communication privilégié	Documentation	Communication directe

XP et RUP

Un comparatif simple entre RUP et XP est effectué dans le tableau ci-dessous (*Tableau 6*).

Tableau 6 : Comparatif simple entre XP et RUP

La méthode	**XP**	**RUP**
Est basé sur	Des valeurs et principes	Des best practices
Met en avant	Les personnes	Un outil
Repose sur un cycle itératif	Itération sur des livraisons d'un logiciel incomplet	Itération sur des processus linéaires
Produit	Du code	Un modèle en faisant l'abstraction du code

Compléments de conception

Lors de la réalisation du projet, un certain nombre de compléments de conception peuvent être employés. On peut citer par exemple : les modèles de données Merise, les mappings ORM, la conception des écrans d'IHM, l'emploi de Design Patterns, l'utilisation de frameworks et la réutilisation de ce qui a déjà été mis en place.

Merise

Merise est une méthode d'analyse, de conception et de gestion de projet. Son principal atout est d'être une méthode complètement intégrée. Elle fournit un cadre méthodologique et un langage. Elle est née dans les années 1970, à la demande du ministère de l'industrie. C'est une méthode spécifiquement française.

Elle préconise une analyse séparée des données et des traitements à plusieurs niveaux : conceptuel, logique et physique. Elle impose de vérifier la cohérence entre les deux analyses avant la validation et le passage au niveau suivant.

Elle est articulée simultanément selon 3 axes.

Le projet suit :

- un cycle de vie composé de phases de conception, de réalisation et de maintenance. Ensuite, un nouveau cycle de projet commence.
- un cycle de décision comportant des choix : GO et NO GO.
- Un cycle d'abstraction en trois parties : conceptuel, logique et physique. On prend d'abord les grandes décisions « métier » avant d'entrer dans les aspects techniques.

Dans cette méthode, on effectue une analyse critique de l'existant (démarche bottom-up), puis on décline la solution retenue (démarche top-down).

Ce recensement de l'existant allonge considérablement la durée du projet. Dans d'autres démarches reposant sur des méthodes itératives de type RAD ou sur l'adoption systématique des best practices observées dans d'autres entreprises, cette analyse préalable n'est pas effectuée.

Au niveau conceptuel, on s'attache aux invariants de l'entreprise. On veut décrire, après l'abstraction, le modèle de l'entreprise ou de l'organisme.

On utilise alors deux types de schéma. Des schémas représentant la structure du Système d'Information du point de vue des données, les Modèles Conceptuels des Données (MCD) et des schémas représentant les traitements, les Modèles Conceptuels des Traitements (MCT).

Le MCD repose sur les notions d'entités et d'associations et sur les notions de relations. L'entité est définie comme un objet de gestion. L'association ou relation est un lien sémantique entre une ou plusieurs entités.

Le MCT repose sur les notions d'événements, d'opérations et de processus. L'événement est assimilable à un message. Un événement peut déclencher une opération ou être le résultat d'une opération. L'opération se déclenche sur un ou plusieurs évènements synchronisés. Elle est constituée d'un ensemble d'actions. Le processus correspond à un enchaînement d'opérations.

Au niveau organisationnel, on précise comment on organise les données de l'entreprise et les tâches ou procédures.

On construit :

- le Modèle Logique des Données (MLD) qui reprend le contenu du MCD mais précise la volumétrie, la structure et l'organisation des données. A ce stade, il est par exemple possible de connaître la liste exhaustive des tables de la base de données relationnelle.
- un Modèle Logique des Traitements (MLT) qui précise les acteurs et les moyens qui seront mis en œuvre. Les traitements sont découpés en procédures fonctionnelles.

Au niveau physique, on établit la manière concrète de mettre le système en place au travers de deux modèles :

- le Modèle Physique des Données (MPD) permet de préciser les systèmes de stockage employés. On implémente le MLD dans le SGBD retenu.

- le Modèle Opérationnel des Traitements (MOT) spécifie les fonctions telles qu'elles seront réalisées.

SADT

SADT est l'acronyme de Structured Analysis and Design Technique. Cette méthode est également appelée IDEF0 (Integration DEFinition for transitions modeling). Elle est d'origine américaine. Elle a été développée par Doug Ross en 1977 pour la société Softech et était répandue dans la fin des années 1980. C'était l'un des standards de description graphique d'un système complexe par analyse fonctionnelle descendante. Dans cette méthode, l'analyse chemine du général vers le particulier et le détaillé. SADT est une démarche systémique de modélisation d'un système complexe ou d'un processus opératoire. Le principal avantage est de proposer une structure hiérarchisée par niveau permettant une clarification et une décomposition analytique de la complexité d'un système. Le principal inconvénient est l'absence de représentation séquentielle.

Classification des méthodologies

Ci-dessous, une classification des méthodologies par rapport à leurs niveaux de formalisme et les natures des cycles suivis.

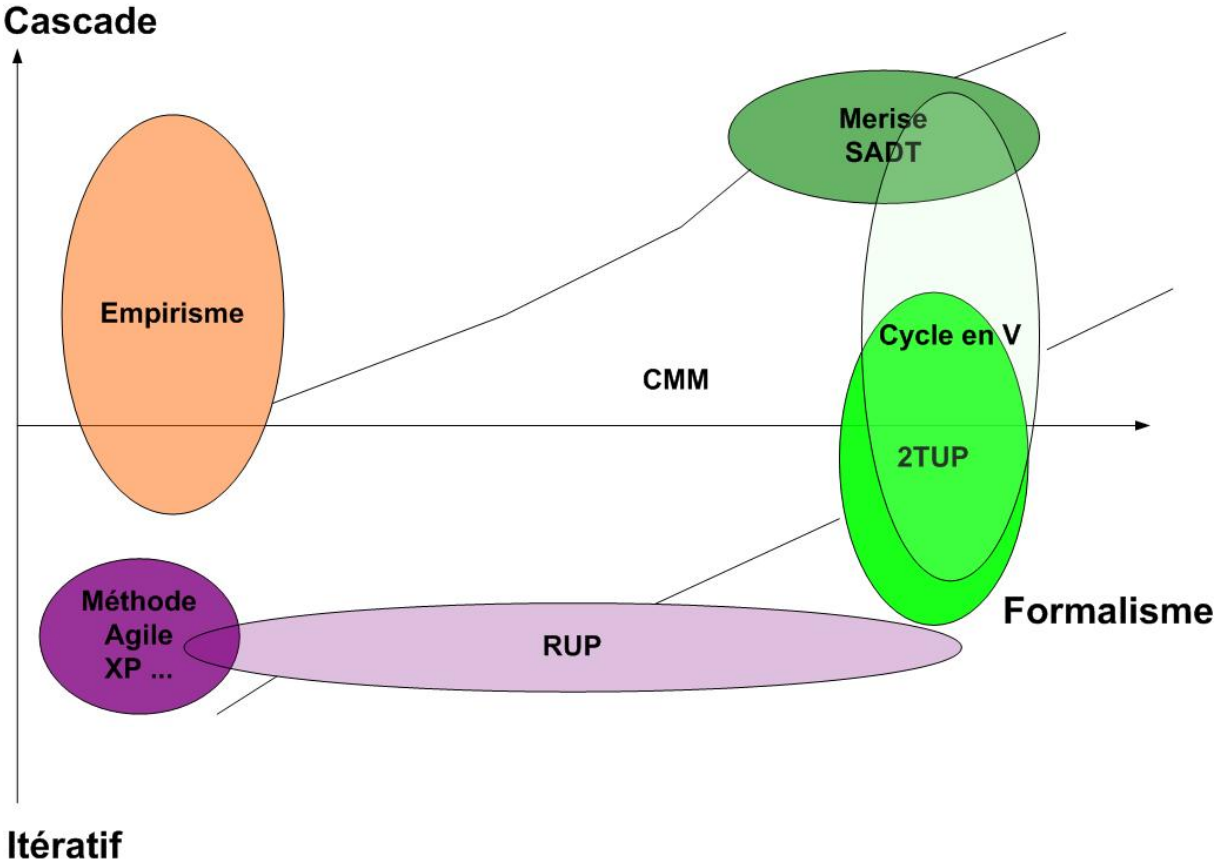


Figure 18 : Classification des méthodologies

CMM

CMM signifie Capability Maturity Model [LAB00]. C'est un système qualité qui vise à améliorer le processus de développement logiciel apparu en 1987. On peut ainsi évaluer la capacité de développement logiciel d'une organisation et la faire évoluer par pallier. Ce modèle se base sur une grille de maturité hiérarchisée. Une organisation d'évolution de processus logiciel en "échafaudage" a été inventée à partir des travaux de Watts Humphrey. Les améliorations réalisées à chaque étage fournissent la fondation de l'étape suivante. Cinq niveaux évolutifs ont été définis. Cette méthode permet d'atteindre des objectifs de coûts, de délais et de qualité. Contrairement à la norme ISO 9000, CMM a été conçu spécifiquement pour le développement logiciel. Ce modèle offre à une organisation bien définie proposant une structure d'évaluation de son niveau de maturité, un ensemble de procédures documentées pour améliorer son niveau de maturité et un ensemble de processus de contrôle pour valider les étapes de cette progression.

Cette méthode commence par la détermination du niveau de maturité actuel de l'organisation. Ensuite l'organisation dispose d'une liste détaillée des actions à entreprendre afin d'atteindre le niveau de maturité recherché. Les niveaux de maturité sont :

- initial

L'organisation n'est pas dans un environnement stable pour l'évaluation, le développement et la maintenance de logiciels.

- reproductible

La gestion de nouveaux projets est basée sur l'expérience d'anciens projets similaires. C'est l'engagement permanent des ressources humaines qui garantit une pérennité du savoir-faire.

- défini

Les directives de gestion de projet sont établies. Le processus standard de développement et d'évolution du logiciel a été documenté.

- maîtrisé

Des objectifs qualitatifs et quantitatifs sont fixés. La productivité et la qualité sont mesurées.

- optimisé

L'organisation toute entière est centrée sur l'amélioration continue des processus.

L'emploi de la méthode CMM ne garantit pas la réussite d'un projet. Des aspects tels que la motivation des équipes, la diminution du turn-over ne sont pas couverts par cette méthode.

2.3. Management des ressources humaines

Le management permet au chef de projet de diriger l'équipe projet afin d'atteindre les objectifs du projet.

Organisation

Les acteurs du projet se répartissent en deux ensembles la MOA et la MOE.

MOA

MOA désigne le Maître d'OuvrAge ou la Maîtrise d'OuvrAge. La maîtrise d'ouvrage est responsable de l'efficacité de l'organisation et des méthodes de travail autour des systèmes d'information. Elle fait appel à un Maître d'OEuvre (MOE) pour obtenir les produits (matériels, logiciels, services et solutions) nécessaires à la réalisation de sa mission. Dans un projet, le maître d'ouvrage décrit les besoins, le cahier des charges, établit le financement et le planning général des projets, fournit au MOE les spécifications fonctionnelles et valide la recette fonctionnelle des produits, assure la responsabilité de pilotage du projet dans ses grandes lignes et adapte le périmètre fonctionnel en cas de retard dans les travaux afin de respecter la date de la livraison finale.

La MOA a également des fonctions d'organisation (conduite du changement, formation des utilisateurs, organisation et modification des processus ou procédures) et doit s'assurer de la bonne adéquation entre la stratégie « métier » de l'entreprise ou de l'organisation (les objectifs, les enjeux...) et le Système d'Information.

MOE

MOE désigne le maître d'œuvre ou l'organisation qui assure la maîtrise d'œuvre garante de la bonne réalisation technique des solutions. Il a un devoir de conseil auprès de la maîtrise d'ouvrage. C'est à la MOE de faire en sorte que le Système d'Information tire le meilleur parti des possibilités techniques et de garantir la qualité technique de la solution.

Equipe projet

Tout projet nécessite la mise en place d'une équipe. Dans une équipe chaque membre de l'équipe a un rôle à tenir et donc la responsabilité d'un certain nombre de tâches à effectuer.

Le chef de projet Informatique a pour responsabilité de planifier les ressources, de former, de développer et de diriger l'équipe projet.

Dans le cas d'une gestion de projet XP, l'équipe comprend 6 membres dont les responsabilités sont indiquées sur la figure ci-dessous [BEN05].

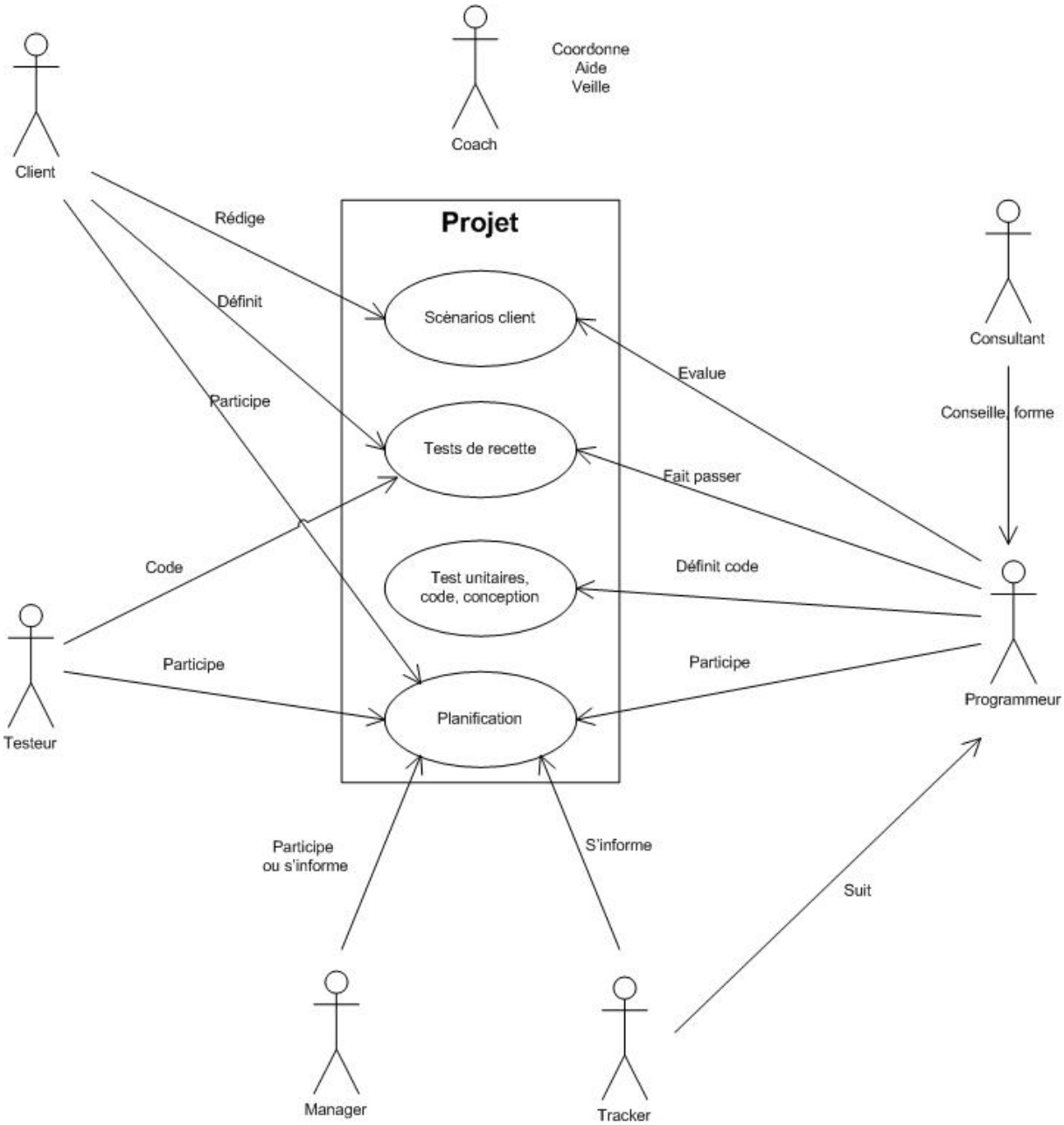


Figure 19 : Organisation d'une équipe XP

Planification et suivi

Le chef de projet doit organiser le suivi du projet et des individus composant l'équipe projet.

Afin de suivre les différentes étapes du projet, il dispose de techniques de planification telles que les réseaux Pert afin de représenter les contraintes d'ordonnancement des tâches du projet, ainsi que les diagrammes de Gantt qui permettent de représenter les contraintes associées aux ressources.

De plus, le chef de projet peut réaliser un tableau de bord afin de rendre compte de l'activité, de l'avancement des lots et de l'évolution des charges restantes.

Il doit diffuser ses rapports d'avancement à sa hiérarchie mais également à l'équipe. La diffusion de l'information au sein de l'équipe est un point crucial. Il est nécessaire de communiquer de façon régulière avec les différents acteurs du projet. La communication humaine est un facteur déterminant pour la bonne réussite d'un projet.

Pour bien conduire le projet, il est nécessaire de contrôler et de réguler le traitement des étapes en fonction des difficultés rencontrées et des modifications des besoins fonctionnels.

Le pilotage d'un projet est cadré par un certain nombre de réunions : comité de direction, comité de pilotage et comité de suivi. Dans les comités de direction, la direction valide les choix stratégiques ou les grandes orientations du projet. Lors des comités de pilotage, les points bloquants sont levés et les décisions importantes sur les orientations de l'application sont prises. Le comité de suivi, quant à lui, permet de suivre la bonne marche du projet.

Les réunions sont nécessaires pour mener à bien un projet mais il est toutefois important de limiter leur coût. Cela peut être fait en ne sollicitant que les personnes concernées. Avant une réunion, on peut également envoyer l'ordre du jour afin que les différents intervenants juge de l'intérêt d'assister ou non à la réunion. Après chaque réunion, un compte rendu doit être fait afin de garder une trace. Avant de diffuser le compte rendu, on peut demander une validation de ce dernier par les différents participants de la réunion.

Il est essentiel que les réunions soient efficaces. Pour qu'elles débouchent sur des résultats, il est nécessaire que les tâches levées lors de ces réunions soient réalisables.

On peut employer des méthodes telles que la méthode Item Action qui consiste à identifier les différents points levés lors de la réunion à des tâches, des décisions à prendre, des recommandations ou des constatations.

Des phrases complètes et compréhensibles par tous doivent être utilisées lors du compte rendu de la réunion. En face d'une action à effectuer sont indiquées son origine, les personnes qui doivent la prendre en charge et les dates d'exécution. Le quoi, le pourquoi, par qui et comment sont ainsi spécifiés. Il ne doit pas y avoir d'ambiguïté lors de la lecture du compte rendu.

L'ensemble des comptes rendus ainsi que la documentation du projet et les livrables doivent être classés de manière ordonnée dans une arborescence de fichiers centralisée.

Gestion du conflit

Toute collaboration humaine débouche sur des points de vue divergents ou des désirs incompatibles. Il est alors nécessaire de gérer les différentes positions et points de vue des parties prenantes tout particulièrement lorsque des conflits apparaissent.

Maîtriser le déroulement

Gestion des délais

Afin d'estimer les délais, il est avant tout nécessaire d'évaluer les charges. La méthode la plus couramment utilisée est de faire une estimation par analogie aux projets que l'on a déjà effectués.

Une fois cette estimation faite, on peut élaborer un échéancier et déterminer le chemin critique du projet.

Il est préférable de conserver quelques marges de manœuvre au niveau de chacune des étapes. On peut alors estimer une durée probable du projet et de chacune des tâches qui le composent. On détermine ainsi une date de livraison et une date de mise en production probable.

Gestion des coûts

Le coût d'un projet peut être décomposé comme suit :
Coûts directs

- Coûts conception / réalisation
  - Coût de main d'œuvre client
  - Coût de main d'œuvre projet
- Coûts logiciels
- Coûts infrastructure
- Coûts de déploiement
  - Coût de main d'œuvre
- Coûts de formation
- Coûts de maintenance évolutive et corrective
  - Coût de main d'oeuvre

Coûts indirects

- Coûts de débordement
  - Coût de main d'œuvre
  - Coût de business du au débordement
- Coûts d'inadéquation aux besoins réels
  - Coût de business dû à l'inadéquation des besoins
- Coûts de changement
  - Coût de main d'œuvre
- Coûts de défaut de qualité
  - Coût de main d'œuvre
  - Coût de business associé à un incident
- Coûts de turn-over
  - Coût de main d'œuvre

Les coûts ont plus ou moins d'importance. Les coûts les plus forts correspondent à la main d'oeuvre informatique. Les coûts logiciel et d'infrastructure peuvent également être très importants.

Les coûts engendrés par l'inactivité des utilisateurs, liés à la non disponibilité de l'application ou à l'inadéquation avec les besoins réels des utilisateurs ne sont pas simples à évaluer.

Des différences de coût peuvent apparaître entre la méthode XP et une méthode basée sur un processus unifié reposant sur l'implémentation d'un modèle. Il n'est pas possible de dire quelle est la méthode la moins coûteuse, cela dépend beaucoup du projet et de son contexte. On peut néanmoins envisager des possibilités de coûts supplémentaires ou des diminutions des coûts sur certains des centres de coûts identifiés précédemment :

- Coûts supplémentaires

La présence du client sur site au sein de l'équipe génère un coût de main d'œuvre client plus important.

Le recourt à la programmation en binôme augmente le coût de la main d'œuvre projet.

Sur les phases de déploiement et de maintenance, l'absence de documentation peut engendrer des surcoûts surtout dans le cas de renouvellement de l'équipe.

La formation continue de l'équipe tout au long du projet génère des coûts de formation conséquents.

- Diminution des coûts

La présence du client sur site au sein de l'équipe permet de minimiser les temps d'attente et les écarts entre les besoins réels et le logiciel obtenu.

Le recours aux outils les plus simples possibles diminue les coûts de licences.

L'intégration continue permet d'anticiper les problèmes de déploiement.

Le client est un membre de l'équipe. Ses besoins de formation sont moins importants que s'il découvre le logiciel lors de sa livraison.

L'utilisation de solutions simples et l'automatisation des tests permettent dans une moindre mesure de diminuer les coûts liés à la maintenance du logiciel.

La méthode XP permet de prévenir les débordements grâce à ses techniques de planification par fonctionnalités.

L'écart entre les fonctionnalités proposées par le logiciel et les besoins du client est minimisé du fait de la forte implication de ce dernier tout au long du projet. Les coûts indirects engendrés par l'inadéquation aux besoins des utilisateurs sont donc fortement diminués.

De part la prise en compte permanente du changement tout au long du projet, il n'y a pas ou peu de coûts indirects liés au changement.

Le recours aux tests automatisés réduit les défauts de qualité.

Les coûts du turn-over sont réduits du fait de l'intérêt porté à chacun dans l'équipe. La dimension humaine a une place importante dans cette méthode. La motivation de chacun des membres est recherchée et favorisée.

Le travail collaboratif diminue l'impact d'un départ.

Gestion des risques

Il n'est pas toujours évident d'identifier les risques. Pourtant, on peut envisager que des tâches reposant sur des domaines fonctionnels ou techniques méconnus représentent des risques. Le fait de travailler pour la première fois avec une personne représente un risque. On peut donc dire que la nouveauté est associée à un risque majeur alors que ce que l'on a l'habitude de faire correspond à un risque moindre.

L'estimation des risques ne peut se faire que sur la base d'une analyse qualitative. On cherche à imaginer les probabilités de risques, leurs impacts et à identifier les facteurs de risques.

L'expérience de la conduite de projet permet de minimiser les risques mais ne les élimine pas pour autant. Les risques ne sont pas toujours prévisibles et peuvent être de nature très diverse telles que la démission d'un membre de l'équipe ou les pannes matérielles.

Gestion de la qualité

La gestion de la qualité est l'ensemble des activités qui permettent d'obtenir de la qualité dans un cadre de production de biens ou de services.

On parle de la qualité totale lorsque l'on cherche à satisfaire les différentes parties prenantes : les fournisseurs, les clients, les actionnaires, les employés et l'État.

La qualité optimale correspond à un équilibre au niveau des besoins de l'ensemble des parties prenantes. Le niveau de qualité optimal ne doit pas produire de coûts inadéquats. On parle alors de sur-qualité. La qualité est censée réduire le coût de la non-qualité. Une entreprise est performante lorsque les ressources qu'elle met en œuvre, sont justifiées et efficaces, et lui permettent de se positionner avantageusement sur un marché avec une marge d'avance sur la concurrence.

Les normes qualité

Les normes internationales de la qualité se sont orientées vers la qualité totale (TQM : Total Quality Management), qui articule stratégie, système, performance, dimension humaine et sociale. En France, le déploiement de la démarche qualité a été tardif : vers 1990. Une version simplifiée de la démarche qualité a alors été élaborée et diffusée sous le nom d'Assurance Qualité, définie dans les normes ISO 9001.

Assurance Qualité

Assurance Qualité ou Quality Assurance (QA) couvre toutes les activités de production depuis la conception, le développement, la production, l'installation, les services et

la documentation. L'objectif étant de bien faire les choses dès la première fois et d'aller droit au but.

De nombreuses méthodes sont utilisées pour améliorer la qualité. Parmi elles on peut citer la roue de Deming, le diagramme de Gantt, l'outil PERT et le brainstorming.

Roue de Deming

La roue de Deming permet de cadrer le pilotage d'une démarche qualité. Elle précise les étapes de mise en place de la maîtrise de la qualité. Elle est aussi désignée par PDCA (Plan - Do - Check - Act : concevoir, mettre en œuvre, contrôler, agir) ou encore par la "roue de la qualité". Cette méthode a été lancée par les qualiticiens JURAN et SHEWART de la société Bell Telephon en 1925.

Diagramme de Gantt

Le diagramme de Gantt permet d'optimiser et de sécuriser un processus. Le diagramme de Gantt est un outil permettant de modéliser la planification de tâches nécessaires à la réalisation d'un projet.

Le diagramme de Gantt a été développé par Henry L. Gantt, ingénieur américain, vers 1910. Il est utilisé en ordonnancement et en gestion de projet. Il permet de visualiser dans le temps les tâches d'un projet. L'avancement du projet est alors représenté graphiquement. Son emploi permet de répondre à des objectifs de planification optimale, de communication sur le planning établi et de justification de choix.

Un diagramme de Gantt permet de déterminer les dates de réalisation d'un projet, d'identifier les marges existantes, de visualiser le retard ou l'avancement des travaux.

De nombreux logiciels de gestion de projet utilisent ces diagrammes. On peut citer par exemple l'offre propriétaire Microsoft Projet ou la solution gratuite GANTT Projet.

Réseau PERT

L'outil PERT est utilisé pour analyser un fonctionnement. C'est une méthode de gestion de projet permettant de définir les tâches et le délai d'un projet et d'en assurer le suivi. Elle a été créée en 1957 par l'US Navy.

PERT signifie Program Evaluation and Review Technique. Il permet de visualiser la dépendance des tâches et de les ordonnancer. Il repose sur un graphe de dépendances. On indique une date de début et de fin au plus tôt et au plus tard de chacune des tâches. On peut en particulier, à l'aide de ce diagramme, déterminer le chemin critique conditionnant la durée minimale du projet. Des tâches critiques sont alors identifiées. Elles correspondent aux tâches qui ne doivent pas subir de retard sinon l'ensemble du projet sera retardé. Cet outil fournit donc une méthode permettant d'optimiser et de planifier l'ordonnancement des tâches du projet.

Le brainstorming

Le brainstorming ou remue-méninges est une méthode de créativité.

ISO 9001

La norme ISO 9001 fait partie des normes ISO 9000 relatives aux systèmes qualité. Elle donne les exigences organisationnelles qui sont requises pour l'existence d'un système de management de la qualité.

Dans la norme ISO 9001 version 2000, les exigences sont relatives à quatre grands domaines :

- Responsabilité de la direction : exigences d'actes de la part de la direction.
- Système qualité : exigences administratives permettant la sauvegarde des acquis. Exigence de prise en compte de la notion de système.
- Processus : identification et gestion des processus contribuant à la satisfaction des parties intéressées.
- Amélioration continue : mesure et enregistrement de la performance, engagement d'actions de progrès efficaces.

La mise en œuvre d'un système de management de la qualité selon les exigences de la norme ISO 9001-Version 2000 consiste à :

- Démontrer l'aptitude à fournir régulièrement un produit conforme aux exigences du client et aux exigences réglementaires applicables.
- Chercher à accroître la satisfaction des clients par l'application efficace du système, et en particulier, mettre en œuvre un processus d'amélioration continue.

Le texte de la norme ISO 9001 aborde les 4 processus principaux :

- La responsabilité de la direction
- Le management des ressources
- La réalisation du produit
- Les processus de mesure, d'analyse et d'amélioration continue

Elle est basée sur 8 principes de management :

- L'écoute client
- Leadership
- L'implication du personnel
- L'approche processus
- Le management par approche système
- L'amélioration continue
- L'approche factuelle pour la prise de décision
- La relation avec les fournisseurs

La version 2000 de la norme ISO 9001 se différencie sur deux points par rapport aux versions précédentes.

Dans les versions précédentes, on définit par écrit ce que l'on doit faire et on fait ce que l'on a écrit. Alors que dans la version 2000, on définit le niveau de qualification (ou de

compétence) nécessaire pour tenir un poste et on s'assure que les personnes tenant ce poste ont la qualification voulue et, si nécessaire, on met en œuvre des formations. On définit un niveau de qualification plutôt que de spécifier un mode opératoire à suivre.

Contrairement aux versions précédentes la satisfaction réelle de l'utilisateur final est plus importante et le client est positionné au sommet de la pyramide.

Les normes ISO 9000 et XP

On peut faire un parallèle entre les principes des normes ISO 9000 avec les principes et valeurs de la méthode XP. Même si la méthode XP ne respecte pas dans les règles de l'art les principes des normes ISO, elle s'en inspire tout de même.

Tableau 7 : Positionnement de XP par rapport aux principes ISO 9000

Principes ISO9000	**Positionnement de la méthode XP**
Orientation client	La méthode XP est orientée client. Le client fait partie de l'équipe. La méthode cherche à comprendre le client et à répondre à son besoin
Leadership	Le bénéfice attendu du leadership est l'établissement de la confiance et l'élimination de la peur, ce qui rappelle les qualités de communication et de courage prônées par XP.
Implication du personnel	La méthode considère que les individus sont plus importants que les technologies et les processus. Cette méthode cherche à mettre en valeur les individus.
Approche processus	XP peut être vu comme un processus avec un formalisme minimal.
Management par approche système	XP s'inspire d'une approche systémique.
Amélioration continue	De fait de l'utilisation d'itérations dans les développements et de la recherche continue de l'amélioration des compétences, XP respecte tout à fait ce principe.
Approche factuelle pour la prise de décision	La méthode XP se veut pragmatique par rapport aux observables.
Relation mutuelle bénéfique pour les fournisseurs	XP est basé sur des relations de type gagnant – gagnant plutôt que sur le respect de clauses d'un contrat initial.

Qualité des données

La qualité des données se réfère à la conformité des données aux usages prévus, dans les modes opératoires, les processus, les prises de décision, et la planification.

La qualité livrée

Pour ce qui est de la qualité, la qualité livrée du logiciel peut être mesurée de différentes manières. En interne, le niveau de qualité est mesuré par le responsable qualité ou les équipes, alors qu'en externe, c'est le client qui mesure le niveau de qualité.

Amélioration de la qualité

Un certain nombre de mesures permettent de contrôler la qualité : la mesure de la conformité, de la satisfaction ou les enquêtes de satisfaction.

La qualité interne peut être améliorée en gérant la qualité de l'architecture et du code par : des lectures croisées, des inspections, des revues de code et d'analyse ou du refactoring.

A force d'implémenter de nouvelles fonctionnalités dans une application, le code source tend à devenir de plus en plus complexe, d'autant plus que l'on emploie des techniques modernes de développement itératif et incrémental.

La refactorisation (refactoring) est une opération de maintenance du code informatique. Elle consiste à retravailler le code source pour améliorer sa lisibilité, simplifier sa maintenance sans pour autant ajouter de nouvelles fonctionnalités.

La qualité interne est également améliorée par l'établissement d'outils tels que les normes de codage, les guides de conception des composants ou les designs patterns.

La qualité externe, quant à elle, peut être améliorée en s'efforçant d'offrir le meilleur service possible à son client.

La qualité est obtenue au travers de

- méthodes structurées
  il est important de déterminer les aspects à traiter, de les ordonner et de gérer le changement par la mise en place de versions de documents et de code.
- la mise en œuvre de moyens et d'outils adaptés
  CVS peut être utilisé pour gérer les différentes versions du code et des documents. Microsoft Office permet d'élaborer de nombreux documents du projet.
- la formation
  le recourt à la formation est un facteur d'amélioration de la qualité et de la productivité du fait de la montée en compétence des acteurs du projet.

L'amélioration de la qualité peut passer par l'utilisation d'UML pour modéliser le système en simplifiant et uniformisant la modélisation, en améliorant la relation entre les différents acteurs et en s'appuyant sur des normes et méthodes.

De plus, il est important d'entretenir une culture de la qualité.

Atteindre des Objectifs

Rentabilité

Tout projet commence en général par l'estimation du ROI (retour sur investissement). C'est un élément important de la prise de décision des dirigeants. Les gains financiers correspondant au projet sont estimés. On évalue également la date de retour sur investissement qui correspond à la date à partir de laquelle le projet permet des gains financiers.

Réutilisation

Il est important de réaliser une phase appelée « mémoire du projet » afin de préciser les aspects réutilisables de la démarche sur des problématiques similaires.

Cette étape permet de tirer parti de ce qui a été fait en évitant de reproduire les mêmes erreurs et en cherchant à réutiliser les points positifs de la démarche projet.

C'est en quelque sorte un bilan du projet.

2.4. ETL et EAI : généralités

ETL GENIO

Présentation

ETL signifie Extract Transform and Load. Un ETL permet de construire de façon simple des processus qui permettent d'extraire des données de systèmes hétérogènes, de les transformer, puis de les charger dans un système cible.

A l'aide d'un ETL, il est possible de connecter diverses sources de données à de multiples systèmes cibles à travers toute l'entreprise, afin d'alimenter un data warehouse, de gérer des méta-données ou d'intégrer des applications d'entreprise.

On peut exploiter de manière plus efficace les données et donc optimiser les processus « métier ». Il est possible de consolider les données afin d'en tirer une information fiable, cohérente et actualisée ou de les échanger entre applications, en ayant au préalable effectué des transformations et nettoyé ou enrichi les données.

Architecture

L'ETL que nous utilisons dans le projet est la version 4 de Genio d'hummingbird.

Son architecture est de type client / serveur, construite sur un modèle Hub and Spoke. Le Hub se compose d'un moteur centralisé et d'un référentiel de données et méta-données. Les sources de données et les cibles entre lesquelles le Hub échange des données correspondent aux Spoke.

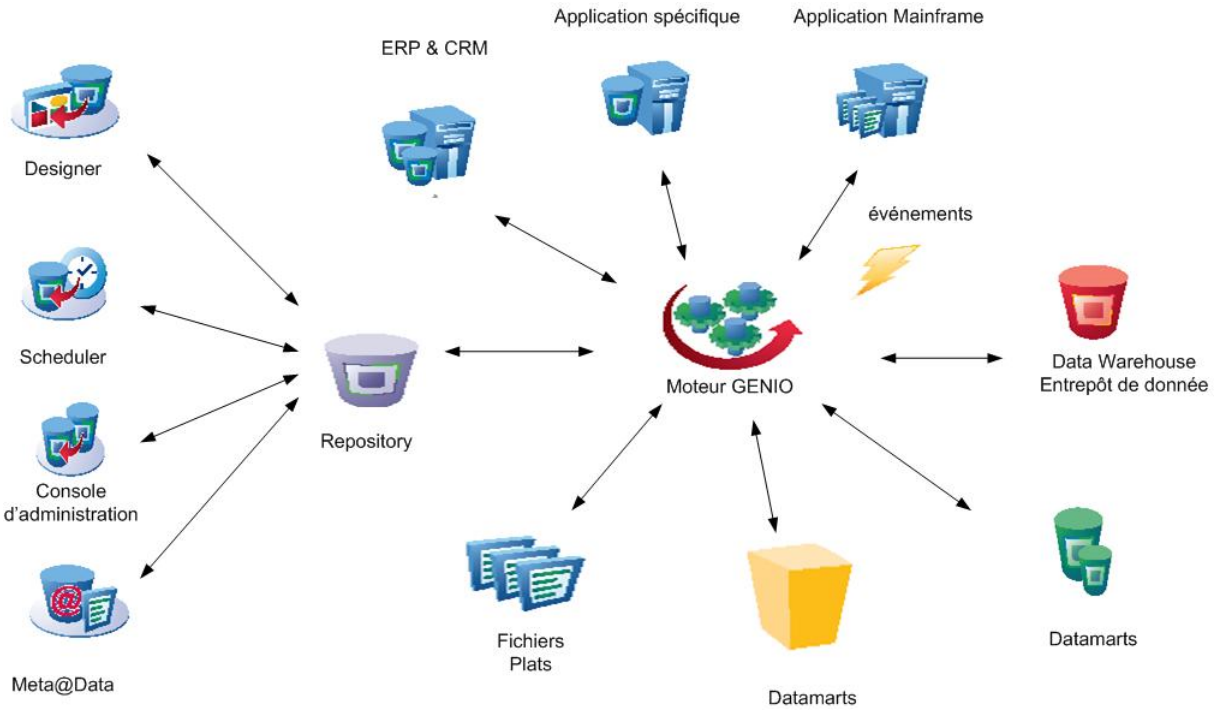


Figure 20 : Architecture de GENIO

GENIO se compose de différents éléments qui permettent de concevoir, de déployer et de maintenir les transformations de données et les processus d'échanges.

Ces composants sont : designer, administration manager, repository, engine, scheduler, metalink et datalink. Nous allons plus nous attacher dans ce qui suit au designer et au scheduler car ce sont les composants que nous utilisons le plus lorsque nous développons.

Engine

L'engine est un moteur de transformation multithread de type broker. Il permet d'exécuter les processus.

Repository

Le repository est un référentiel objet pouvant être hébergé sur les principaux SGBDR du marché. Dans notre cas, nous l'avons hébergé sur une base ORACLE 8. Dans ce référentiel sont stockés et gérés tous les aspects des transformations, les méta-données des processus d'échange, les objets créés au travers du designer, ainsi que les informations relatives aux exécutions des processus.

Administration manager (Console d'administration)

L'interface d'administration permet d'initialiser l'environnement, le repository et de gérer l'administration des utilisateurs et leurs droits.

Scheduler

Le scheduler permet

- la planification temporelle et événementielle. Il est possible de programmer, déclencher et contrôler les processus dont l'exécution peut être planifiée par des événements externes tels que la modification de fichier, le transfert de fichier, à une heure donnée, à partir d'un calendrier, de façon périodique (journalière, hebdomadaire ou mensuelle) ou à la création d'un fichier. Il est un point d'entrée unique pour l'ensemble des planifications de l'architecture.
- le contrôle et la surveillance des exécutions en temps réel au niveau du moteur de l'ETL.
- la visualisation des comptes rendus d'exécution. On peut trier les exécutions par date, par nom de processus, par nombre d'erreurs.
- l'accès à un historique complet

Metalink

Metalink permet de récupérer et de transformer des données et métadonnées provenant d'applications ou de divers ERP.

Datalink

Datalink propose une bibliothèque de liens aux SGBD relationnels ou multidimensionnels et aux systèmes de gestion de fichiers plats.

Designer

Le Designer permet de concevoir les processus exécutables dans le moteur de l'ETL au sein de projets. Il est possible de créer de nombreux objets qui vont permettre de concevoir les différents processus.

Ses caractéristiques

Gain de productivité

Le Designer est une interface graphique qui simplifie les opérations de mapping des données sources avec les structures cibles. Le temps d'apprentissage et les ressources nécessaires afin d'être opérationnel sont minimes par rapport à l'apprentissage d'un langage de programmation tel que Java. C'est un environnement multi utilisateurs qui permet d'élaborer des transformations de données, ainsi que des processus d'échange.

Développement simple

Le designer permet de manipuler de nombreux objets simples qui vont permettre de concevoir les différents processus. Voici une liste non exhaustive d'objets les plus couramment utilisés :

- **Connexion**

C'est un lien vers une base de donnée ou un fichier

- **Table**

Elles sont liées à une connexion et sont le reflet des tables d'une base de donnée ou de la structure d'un fichier.

- **DataSet**

C'est un ensemble de tables d'une connexion avec la possibilité de faire des jointures. Dans le cas des bases de données, on peut utiliser des fonctions telles que somme, max, min … Les fonctions proposées restent très basiques. On peut néanmoins faire des jointures, des agrégations, des distinctions, des filtres ou des tris.

- **Module**

Il accepte une table ou un dataset en entrée et plusieurs sources de données en sortie. Il permet la lecture d'informations présentes dans une base de données ou un fichier, de transformer ces informations ou de les mapper et de les écrire dans un ou plusieurs autres systèmes cibles (bases de données ou fichiers).

C'est au niveau du module qu'est offerte la possibilité d'utiliser un langage simple. On dispose ainsi de fonctions analogues aux langages de programmation : fonctions mathématiques, de manipulation de chaîne ou de conversion.

- **Processus**

Un processus se compose de modules, de déclenchement d'exécutables, d'envois d'email ou d'événements …

Les exécutables peuvent être des batchs externes ou des scripts Shell dans le cas où l'on aurait des besoins de transformations spécifiques.

Les notions de transaction, de commit et de rollback à l'intérieur de ces programmes sont prises en charge.

Possibilité de mise en œuvre de transformation complexe

Des possibilités de scripting disponibles au niveau des modules permettent de mettre en œuvre des processus de transformation complexe de données. Ce langage simple permet d'utiliser des boucles, des tests, des variables et des instructions. On peut définir des variables ou utiliser des instructions logiques telles que *for* ou *if*. Un ensemble complet de fonctions de transformation permet de créer des programmes de transformation au même titre qu'un langage de programmation. Il offre de nombreuses fonctions utilisées pour construire des expressions avec des sources de données et des variables. Ces fonctions couvrent le spectre complet de la manipulation des chaînes, dates, nombres, et booléens. Des clauses complexes telles que des IF, THEN, ELSE et CASE peuvent également être utilisées dans les expressions. Les utilisateurs peuvent ainsi créer leurs propres macros afin de décrire leurs règles « métier ». Les objets créés sont stockés dans le repository afin d'être réutilisés.

Analyse d'impact

Une analyse d'impact des modifications effectuées sur les objets et les objets associés est réalisée par le designer. Si un changement touche l'intégrité d'un objet, l'objet devient

invalide. Les modifications faites au niveau des structures de données sources et cibles sont également détectées. Ceci permet d'éliminer les risques de corruption de données ou d'échec pendant les processus de transformations.

Transparence des implémentations techniques

GENIO utilise les mêmes fonctions lorsqu'il manipule des fichiers ou des sources de données. Il ne fait pas de distinction sur le fait que la source ou la cible soient des fichiers « texte » ou des tables d'une base de données. L'utilisateur n'a pas à se soucier des implémentations techniques des fonctions qu'il utilise. C'est le moteur qui gère cet aspect et qui va utiliser la grammaire SQL native correspondant à une base de données précise afin d'exploiter les possibilités propres de la base de données pour accéder aux seules données que l'on souhaite transformer. Cette approche présente l'avantage de minimiser le trafic réseau.

Gestion des erreurs

Il est possible de lever des exceptions et de gérer des erreurs. Un module peut retourner un code erreur permettant, au sein d'un processus, de conditionner l'exécution des modules qui le suivent.

A l'aide de la commande *write*, utilisable dans les modules, il est possible d'écrire des messages dans le compte rendu d'exécution. De plus, le compte rendu d'exécution propose par défaut la visualisation de nombreuses autres informations qui facilitent la détection des erreurs : l'ensemble des ordres SQL effectués sur les différentes bases de données, des compteurs correspondant aux enregistrements lus, écrits, mis à jour …

Implémentations possibles

Il existe différents types d'implémentation. Soit l'essentiel des manipulations de données se fait au sein du module, soit on déporte une partie des transformations au niveau des bases de données sources ou cibles. Ou encore, on exécute des ordres SQL. Dans ce dernier cas, c'est le moteur de base de données qui fait tout le travail, et il n'y a pas de transfert d'information sur le réseau.

EAI / VBIS

Présentation

L'EAI (Entreprise Application Intégration) désigne des processus et des progiciels permettant d'automatiser les échanges entre différentes applications d'une même entreprise ou entre les systèmes d'information d'entreprises différentes. Les applications et les entreprises peuvent ainsi s'échanger des données, des messages et collaborer à l'aide de processus communs.

Une plate-forme EAI relie les applications hétérogènes du Système d'Information autour d'un bus logiciel commun, chargé du transport des données.

Deux facteurs principaux poussent au développement de la mise en place de l'EAI dans les entreprises :

Le premier est lié à la composition très diversifiée du Système d'Information de l'entreprise. L'EAI permet d'exploiter les données et le potentiel de ces différents systèmes, et également de faciliter la maintenance des passerelles existant entre ces systèmes.

Le second facteur de développement de l'EAI est lié aux événements ou aux changements structurels qui poussent les entreprises à augmenter la flexibilité et la réactivité de leur Système d'Information, d'où la généralisation des besoins d'intégration entre les applications de production (ERP), de logistique (SCM), de relation client (CRM)… La chaîne de valeur se conçoit dans sa globalité.

Deux grandes Familles

On distingue deux grandes familles de technologies EAI : les EAI tactiques et les EAI d'infrastructure.

L'EAI tactique intervient dans un périmètre limité, généralement comme une brique d'un projet fonctionnel de l'entreprise. C'est un outil, un composant qui va permettre d'automatiser un ensemble de traitements localisés, sans impact sur le Système d'Information global de l'entreprise.

L'EAI d'infrastructure implique une prise en compte et une évolution éventuelle du système global de l'entreprise et de ses processus. On parle alors de Business Process Management. Ce sont des solutions qui se justifient dans des environnements applicatifs complexes ou des environnements critiques, de grandes entreprises. L'EAI devient alors l'un des piliers de l'architecture du Système d'Information.

VBIS

Nous allons présenter dans cette partie une solution logicielle de la société Vignette qui permet d'automatiser un ensemble de traitements localisés. Si on va sur le site de l'éditeur, ce logiciel est présenté comme étant un ETL. Mais comme nous le verrons dans la suite de part ses caractéristiques il s'assimile davantage à un EAI tactique. La frontière entre les ETL et les EAI n'est pas très nette et nombre d'ETL et d'EAI présentent les mêmes

fonctionnalités. Une chose est sûre : il est préférable, pour des raisons marketing, de mettre en avant les fonctionnalités ETL de ces produits, les ETL ayant la réputation d'être beaucoup plus stables que les EAI.

Cet outil se compose d'un environnement de développement VBIS (Vignette Business Intégration Studio).

Il permet d'intégrer des applications d'entreprise, des procédures et de l'information de façon rapide. Il offre des possibilités d'intégration et d'unification des systèmes d'information d'entreprise.

Il est possible d'intégrer de l'information à partir - ou vers - des applications, des entrepôts de données ou d'autres sources de données présentes à l'intérieur ou à l'extérieur de l'entreprise.

Caractéristiques

A l'aide d'une interface graphique simple, il permet de réduire les temps de développement, de recette et de déploiement et donc de réduire les coûts liés à la mise en œuvre et à la maintenance des applications.

L'environnement graphique permet le développement de programmes à l'aide d'adaptateurs sans la moindre ligne de code.

Les adaptateurs sont associés aux diagrammes de flux. Les diagrammes de flux qui forment les processus d'adaptation sont exécutés comme une séquence d'étapes. Il existe donc un ordre d'exécution entre eux. Les adaptateurs correspondent aux étapes, et la direction des connexions entre les adaptateurs détermine de quelle façon les étapes sont exécutées. L'exécution des diagrammes de flux est contrôlée à l'aide de ports de contrôle tels que l'exécution port qui prend la valeur *true* ou *false*. Si une valeur false est levée, les adaptateurs qui suivent ne sont pas exécutés.

Un diagramme de flux particulier est associé à l'exécutable pour indiquer le premier diagramme de flux à exécuter. L'ensemble des diagrammes de flux est regroupé au sein d'un projet.

L'environnement se compose :

- d'une fenêtre des librairies d'adaptateur
- de fenêtres permettant l'édition des diagrammes de flux
- d'une fenêtre projet

Ces adaptateurs peuvent être testés à l'aide de VBIS. Le débuggeur permet de gérer des points d'arrêt, de suivre pas à pas les opérations, les valeurs, de visualiser et de se déplacer entre différents niveaux des diagrammes de flux.

Librairies des Adaptateurs

Les adaptateurs se présentent sous forme de modules. Ils permettent de se connecter à des applications d'entreprise spécifiques et à diverses sources d'information. Il existe des adaptateurs pour de nombreuses applications et plateformes. Ils permettent d'intégrer diverses applications d'entreprise (ERP, CRM, système back office), applications gros système, bases de données et autres entrepôts de données internes ou externes à l'entreprise. Il existe également des adaptateurs spécialisés pour de nombreux standards : COM, EDI, EJB, FTP,

Flat Files, HTTP, CORBA (Iona Orbix), JavaBean, JDBC, JMS, LDAP, ODBC, SNMP et XML *(Figure 21)*.

Un ensemble d'adaptateurs est disponible afin de contrôler les flux et la logique, de surveiller et de gérer les erreurs des processus d'intégration et des applications. Ce qui permet d'implémenter la logique « métier » de façon graphique.

Il est possible de réutiliser les processus d'intégration en tant qu'adaptateurs à l'intérieur d'autres processus d'intégration. On peut ainsi imbriquer les procédures « métier ».

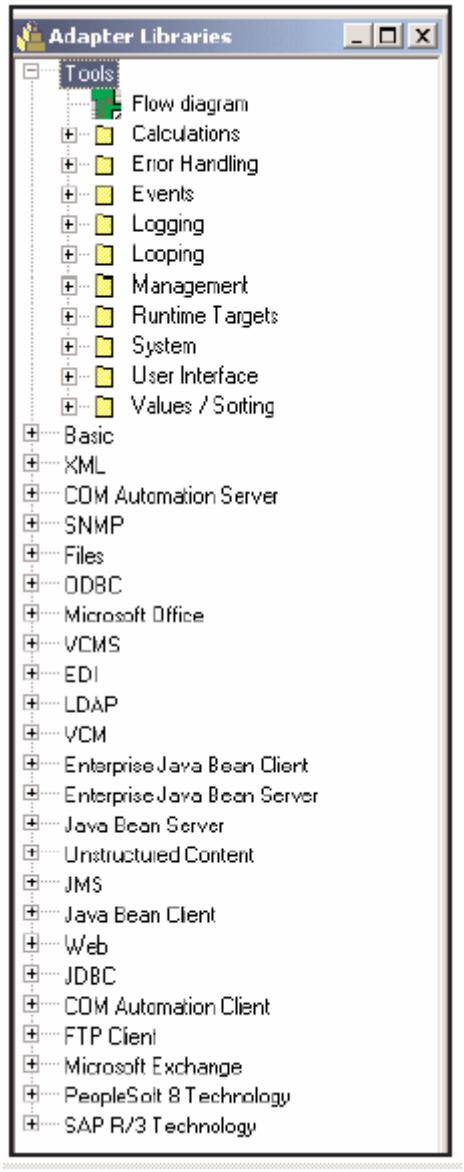


Figure 21 : Différentes librairies d'adaptateurs existants

Ces adaptateurs permettent de connecter un large panel de composants, de systèmes et d'applications. Il est donc possible de gérer des flux de données et de les contrôler au travers d'applications et de sources de données.

Les Processus

Les processus sont élaborés à l'aide de diagrammes de flux. Les diagrammes de flux sont des programmes composés d'éléments visuels (adaptateurs et autres diagrammes de flux) connectés les uns aux autres pour tenir compte de flux de données et de contrôles au travers des applications et des sources de données. Les diagrammes de flux fournissent une méthode simple et rapide d'accéder à l'information existante et de la transformer. VBIS se compose d'un ensemble d'éléments visuels qui correspondent à des données, des applications et des opérations.

Les éléments fournissent des informations sur la façon dont ils sont exécutés. Il est possible de surveiller les adaptateurs, de déterminer si un adaptateur s'exécute, de jauger la progression par rapport à un but connu, de détecter les erreurs, de mesurer la performance, de récolter des statistiques et d'effectuer des audits.

Ils permettent également de visualiser la façon dont se propage l'information entre les applications.

Les adaptateurs sont les éléments fondamentaux des diagrammes de flux, on peut sélectionner des méthodes sur ces adaptateurs afin d'effectuer des opérations spécifiques. Les adaptateurs peuvent avoir plusieurs méthodes mais on ne peut en sélectionner qu'une seule à la fois.

Les adaptateurs sont connectés à l'aide de ports. Un port peut recevoir des entrées d'un autre adaptateur et envoyer des sorties vers d'autres ports. Dans le cas de l'adaptateur d'accès aux bases de données via ODBC, les ports correspondent aux colonnes d'une table relationnelle.

Un port peut être obligatoire ou optionnel, recevoir ou envoyer une ou plusieurs valeurs à la fois. Les connexions entre les ports sont représentées par des flèches.

Flexibilité de déploiement

Le déploiement des projets, une fois réalisés, est très souple. Il est possible de déployer les processus d'intégration de contenu sous la forme de programmes Java standards (EJB, JavaBean,.jar), ou de programmes Windows (COM,.exe), ou de services Web, sur des systèmes d'exploitation variés, sans écrire de code. Un même diagramme de processus d'intégration peut être déployé sous de multiples formes dans de multiples environnements d'exploitation, sans changement. Il n'est pas nécessaire de gérer la complexité de l'implémentation.

Comparatif ETL /EAI

ETL et/ou EAI

Il existe aujourd'hui plusieurs types d'outils sur le marché de l'intégration de données. Les deux principales catégories qui sont utilisées pour implémenter les échanges de données sont les outils ETL et EAI. Les architectures des ETL et des EAI partagent de nombreux concepts communs. Cependant, ils correspondent à des besoins « métier » différents et offrent des fonctionnalités spécifiques. En général, les ETL sont des solutions de traitement automatique d'échange de données axées sur des données techniques et travaillant principalement avec des RDBMS (Relational DataBase Management System).

D'un autre côté, les outils d'EAI sont considérés comme des solutions d'échange de données en temps réel, axées sur des données « métier » et qui travaillent principalement avec des MOM (Message Oriented Middleware).

Cependant, les deux catégories de produits ont souvent affaire aux mêmes données et implémentent les mêmes règles « métier ».

GENIO est un ETL qui offre un certain nombre d'extensions provenant des EAI en proposant un certain nombre d'adaptateurs à diverses applications d'entreprises. Il permet d'éviter dans une certaine mesure de se doter d'une solution ETL et d'une solution EAI distinctes.

Plusieurs cas de figures peuvent faire en sorte que l'on ait recours à un EAI tactique en complément d'un ETL tel que GENIO.

En effet, la bibliothèque d'adaptateurs proposée permet de se connecter à diverses solutions du marché mais est moins riche que celle des EAI. On peut donc être obligé de se doter d'une solution EAI, en plus de cet ETL, afin de bénéficier d'adaptateurs correspondant à des applications qui ne sont pas proposées par l'ETL.

D'autre part, GENIO peut ne pas permettre de gérer pleinement les besoins d'intégration d'applications dans le cas où l'on souhaiterait mettre en œuvre des programmes, sous forme d'exécutables, manipulant des données sans vouloir utiliser de langage de programmation.

ETL

Traditionnellement, les ETL permettaient d'extraire, de transformer et de charger des données en mode batch, afin d'alimenter des entrepôts de données utilisés par des applications décisionnelles.

Par contre, la nécessité d'augmenter les fréquences de rafraîchissement de l'information qui se rapprochent de plus en plus du temps réel, fait que les ETL répondent désormais à des problématiques plus larges d'intégration de données, à la fois batch et en temps réel. Ils se rapprochent ainsi des problématiques traitées par les EAI.

Les ETL sont utilisés pour les mouvements de données en mode batch, principalement pour :

- le chargement des Datawarehouse
- les opérations de réplication et de migration entre bases de données
- la mise à disposition de données auprès de systèmes décisionnels
- l'intégration de données hétérogènes plutôt en mode batch mais parfois en temps réel
- l'analyse préalable des données afin d'obtenir des statistiques sur la fiabilité des données. On peut vouloir déterminer le type des données ou les index utilisés pour y accéder. On caractérise alors les données par un profil.
- gérer la qualité des données. On peut rediriger des erreurs dans des tables d'anomalies en vue de les retraiter.

Même si les ETL et EAI ont tendance à converger sur certains aspects, il existe cependant des différences importantes entre ces deux catégories de produit.

EAI

Les EAI permettent de résoudre l'hétérogénéité du Système d'Information en intégrant et en faisant collaborer les différentes applications ou progiciels.

Il est possible d'intégrer des données en temps réel. La mise en correspondance des formats d'entrée et de sortie se fait à l'aide de graphiques.

Avec un EAI on peut gérer des données de référence de façon cohérente et unifiée, et inclure dans les processus « métier » des validations d'utilisateurs par le biais de workflow.

Certaines solutions EAI permettent également de gérer des cadres d'échange pour chaque type de partenaire de l'entreprise afin de répondre aux besoins d'intégration inter entreprise.

Les EAI proposent trois grandes couches de services :

- le transport et la connectivité : ils proposent une large bibliothèque de connecteurs permettant de relier la plupart des grands progiciels ERP et CRM du marché.
- le routage et la transformation : transformer un format de donnée d'une application source en un format compréhensible par l'application cible.
- BPM (Business Process Management) : permet de se détacher de la technique et de voir les processus sous un angle « métier ».

Tableau comparatif

Les deux familles de produit que sont les EAI et les ETL tentent toutes deux de répondre aux problématiques d'intégration des données hétérogènes en temps réel et aux problématiques de gestion des données de références MDM (Master Data Manager). Ce sont les deux problématiques sur lesquelles ces produits se recouvrent.

L'ETL, initialement mis en œuvre pour extraire des données de production, les transformer et les charger dans un Dataware house en mode batch, peut aujourd'hui récupérer en instantané les données dans les systèmes de production (ERP, CRM) ou être utilisé dans des problématiques de migration d'applications, d'urbanisation ou d'homogénéisation du Système d'Information.

De plus, les Datawarehouse deviennent une application du Système d'Information comme une autre à partir de laquelle on peut répercuter des changements dans le système de production et ainsi modifier les données de départ.

Les EAI, quant à eux, évoluent pour devenir la plate forme constitutive de nouvelles architectures orientée service (SOA) auprès des annuaires de services, avec l'intégration dans un portail et via l'orchestration des services grâce aux couches de BPM.

Tableau 8 : Comparatif des caractéristiques EAI/ETL d'un point de vue général

Caractéristiques	EAI	ETL
Orientation	Orienté Processus	Orienté Données
Principe	Basé sur la mise en œuvre de règles « métier »	Basé sur la mise en correspondance de schémas de données différentes
Volumétrie gérée à l'aide de l'outil	Volumétrie faible de l'ordre de 100 000 enregistrements dans le cas de VBIS afin d'obtenir des temps d'exécution raisonnables	Capable de gérer des volumétries importantes de données. De l'ordre de plusieurs millions d'enregistrements dans le cas de GENIO.
Possibilité d'analyse des données	les EAI étant basés sur une approche orientée « métier », ils ne permettent pas d'analyser les types des données et les index utilisés	Reposant sur la mise en correspondance de schémas de données, des possibilités d'analyse des données sont offertes par les ETL.
Gestion de la qualité des données	Certaines possibilités existent mais sont très limitées	Il est possible de nettoyer les données et de mettre en place des mécanismes de gestion des erreurs.
Possibilité de transformation des données	Il existe des fonctions simples permettant de transformer de l'information. Les possibilités d'agrégation des données sont limitées	Les fonctions de transformation et d'agrégation des données sont plus nombreuses et plus élaborées.
Possibilité d'utilisation de format PIVOT	Possible	Possible
Gestion des Méta-données	Possible	Possible
Gestion des données de références	Pas encore couvert	Pas encore couvert
Connecteurs progiciels	Il existe de nombreux connecteurs	Quelques connecteurs mais moins nombreux que dans le cas des EAI
Bus d'échange et de transformation	Possibilité	Possibilité
Gestion des processus « Métier » BPM	Possible dans certains EAI mais pas dans le cas de VBIS	Non significatif
Intégration inter entreprise	Possible	Non significatif
Rafraîchissement	Temps réel asynchrone	Essentiellement batch

2.5. Organiser une application J2EE

Comment organiser une application WEB ?

J2EE

J2EE est une plateforme, c'est à dire une base générique qui fournit un ensemble de fonctionnalités utilisables dans une majorité d'applications. Les développeurs ont à leur disposition un ensemble d'outils qu'ils n'ont pas besoin de concevoir eux même.

J2EE, qui signifie Java 2 Enterprise Edition, est une norme proposée par la société Sun. Elle a été adoptée par de grands groupes mondiaux tels que : IBM, Oracle, BEA …

Existant depuis 1998, elle est stable et fiable. Elle évolue constamment. Sun a mis en place un système de spécifications permettant d'être à l'écoute des besoins, des propositions ou critiques des développeurs.

L'utilisation de Java permet de rendre les applications totalement indépendantes de la plateforme système utilisée. Une même application peut aussi bien tourner sous Windows que sous Unix.

De plus, les développements ont l'avantage d'être plus facilement maintenables que d'autres langages tels que le C par exemple.

J2EE comprend les spécifications du serveur d'application [JOU05b] (son environnement d'exécution).

Services

La plateforme propose des services offrant un certain nombre de fonctionnalités au travers d'API. Les API présentent l'avantage d'être faciles à prendre en main. Elles permettent de cacher la complexité d'accès aux ressources et donc de gagner considérablement du temps. Les développeurs peuvent ainsi consacrer plus de temps aux aspects « métier ».

Il existe deux types de services : des services d'infrastructure et des services de communication. Les deux tableaux suivants décrivent les différentes API de chacun de ces types.

Services d'infrastructure

Tableau 9 : Services d'infrastructure

Nom de l'API	Description
JDBC - Java Database Connectivity	API d'accès aux bases de données. Son utilisation diminue le nombre de lignes de code à écrire. De plus, les accès peuvent être optimisés à l'aide des pools de connexions fournis par les serveurs d'application.
JNDI	API d'accès aux services de nommage et aux annuaires d'entreprises (DNS, NIS, LDAP, …).
JTA / JTS - Java Transaction Api / Java Transaction Services	API définissant des interfaces standards avec un gestionnaire de transactions.
JCA (J2EE Connector Architecture)	API de connexion au Système d'Information de l'entreprise (ERP…).
JMX (Java Management eXtension)	API permettant de développer des applications WEB de supervision d'applications.

Services de communication

Tableau 10 : Services de communication

Nom de l'API	Description
JAAS (Java Authentification and Authorization Service)	API de gestion de l'authentification et des droits d'accès.
RMI (Remote Method Invocation)	API permettant la communication synchrone entre objets.
Web Services	permettent de « partager » un ensemble de méthodes qui pourront être appelées à distance. Cette technologie utilise XML, ce qui permet de l'employer avec n'importe quel langage et n'importe quelle plateforme.
JMS (Java Message Service)	API fournit des fonctionnalités de communication asynchrone (appelées MOM pour Middleware Object Message) entre applications.
JavaMail	API permettant l'envoi de courrier électronique.

Composants

J2EE repose sur des composants. Ce qui permet :

- d'étendre l'architecture de façon simple
- d'obtenir une bonne qualité de service par la mise en place de mécanismes de haute disponibilité
- de faciliter la maintenance des applications

On peut distinguer 2 catégories de composants : les composants Web et les composants « métier ».

Composants Web

Les composants Web correspondent à la partie présentation d'une application. Ils permettent d'implémenter des Interfaces Homme Machine (IHM) et les traitements qui leur correspondent.

Pour permettre d'avoir un code performant, robuste et maintenable, différentes technologies sont utilisées telles que les JSP et les servlets.

JSP signifie Java Server Page. Ces pages servent à générer le code HTML de l'IHM. On peut utiliser dans ces pages des scriplets Java ou des balises personnalisées.

Les Servlets sont des classes Java qui permettent le traitement des requêtes utilisateurs.

Composants Métier

Ils sont chargés des traitements des données propres à un secteur d'activité. Ils doivent également assurer l'interfaçage avec les bases de données. Les EJB (Entreprise JavaBean) sont un exemple de composant « métier ».

Les composants standards que nous venons de voir peuvent parfois être assez lourds à utiliser. Ils sont conçus pour de grandes architectures et peuvent être mal adaptés aux applications moins importantes.

On peut alors utiliser des frameworks servant à simplifier l'utilisation des technologies standards. Ils sont souvent plus limités, mais leur utilisation est plus simple.

Architecture basique d'une application

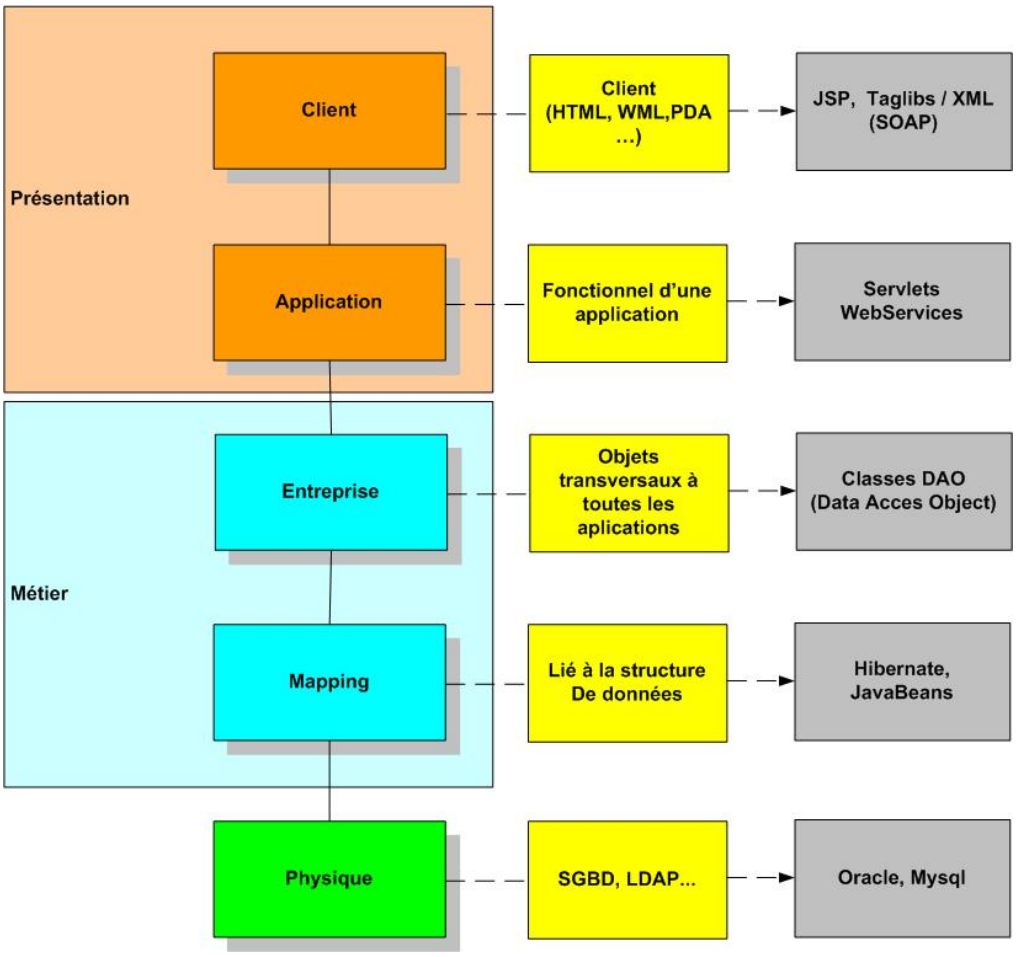


Figure 22 : Exemple d'architecture basique d'application

Cette architecture permet de décomposer le système en sous-systèmes. Elle se compose de 5 couches :

- Client

C'est l'interface utilisateur (HTML, WML, PDA…). Du fait qu'elle ne gère pas les règles fonctionnelles, on peut la faire évoluer facilement.

- Application

Elle permet de traiter le fonctionnel de l'application. Elle utilise les services de la couche Entreprise. La couche Application permet de gérer le fonctionnel spécifique à partir de vues d'objets « métier ». Elle peut être implémentée à l'aide de Servlets ou de Web Services.

- Entreprise

Elle correspond aux objets structurants de l'entreprise. Ces objets sont transversaux à toute l'entreprise. Elle propose des services d'accès aux objets au travers de méthodes de création, de modification, de suppression et de recherche. On peut utiliser des classes DAO ou EJB.

- Mapping

Cette couche est liée à la structure des données. Dans le cas d'un SGBDR, on utilise un outil de mapping relationnel objet tel que Hibernate.

- Physique

Cette couche correspond à la structure physique des données : SGBD tel que ORACLE, CICS, LDAP…

Le schéma ci-dessus *(Figure 22)* est un exemple d'architecture d'application basique. Cette architecture n'est pas générique. Elle est adaptable en fonction du projet que l'on a à mener. Les noms utilisés pour les différentes « couches » peuvent varier suivant les personnes. On peut trouver le terme de couche de Présentation pour désigner la couche gérant l'interface homme / machine. La couche « métier » est également appelée « couche de Domaine ».

L'architecture que nous utiliserons *(Figure 23)* est simplifiée par rapport à cette version basique.

Elle comporte :

- une couche Présentation
- une couche DAO
- une couche Physique

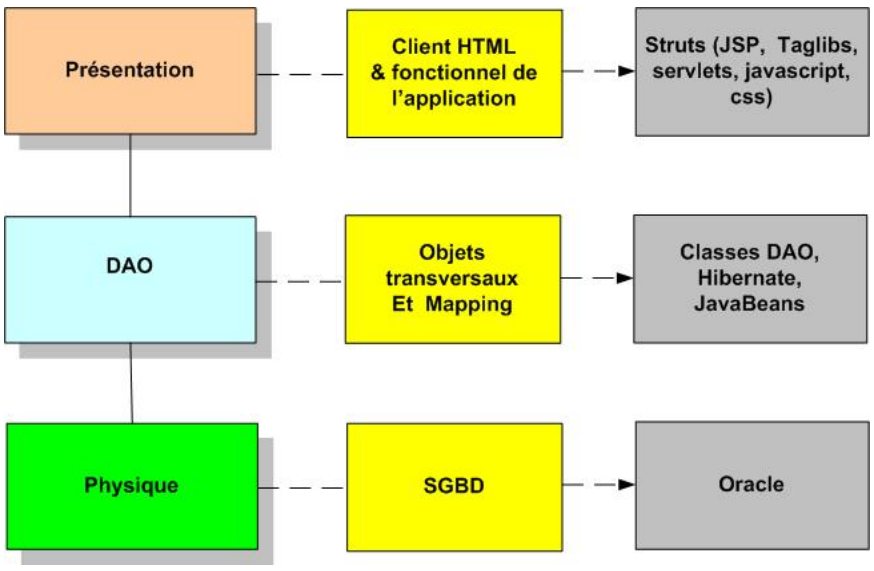


Figure 23 : Architecture utilisée

Dans les sections suivantes, nous allons détailler la couche Application & Présentation et la couche « métier ».

Couche Application & Présentation

Cette couche est très liée au type de client utilisé.

Lorsque l'application n'a pas besoin d'utiliser des composants très spécifiques, on met en place des applications WEB avec un client léger pour les utilisateurs. Il est en effet très simple de générer les interfaces de façon rapide à l'aide d'HTML.

L'inconvénient de l'utilisation de client léger peut être que, dans certains cas, les flux réseaux sont beaucoup plus importants qu'avec une solution basée sur un client riche. Il est alors nécessaire de palier à cet inconvénient en recourant à un réseau possédant des débits plus importants.

La couche Application ne constitue pas toujours une couche indépendante. Elle sert à assurer la communication entre la couche Présentation et la couche « métier ». Elle permet également de contrôler l'enchaînement des différentes tâches. Elle a la charge de connaître l'état des sessions des clients. Dans notre cas nous parlerons uniquement de couche Présentation pour désigner la couche prenant en charge les interfaces utilisateurs.

Pour mettre en œuvre cette couche, il existe deux possibilités :

- Utiliser les Servlets et les JSP

Ceux sont des composants de bas niveau qui n'offrent que des fonctionnalités de base. Les Servlets serviront de « Contrôleur ». Ils devront exécuter les traitements liés aux requêtes utilisateurs, ainsi que les appels aux modèles « métier ».
Les JSP permettront de gérer la partie présentation en générant du HTML à envoyer au client. Elles ne devront avoir qu'un minimum de code Java afin d'être optimales.

- Utiliser un framework

On peut par exemple utiliser le framework Struts qui repose sur une architecture MVC (Model View Controller) [FEL06]. Ce framework est basé sur les Servlet / JSP. A l'aide de cette architecture MVC, le travail du développeur est simplifié.

Couche Métier

Dans le cas d'applications importantes et si l'on doit utiliser un ensemble de clients très différents tels que des clients WEB, clients riches ou des clients programmés avec des langages divers et variés, l'utilisation des Entreprise JavaBean (EJB) peut être utile.
Cependant, si l'application WEB n'a besoin que d'un seul type de client, on pourra alors utiliser des solutions plus légères telles que Hibernate.

Design Patterns

Les patterns - ou modèles de conception - permettent de répondre à un problème particulier [FRE06]. Le principal intérêt des patterns est de permettre la manipulation d'artefacts plus élaborés que les objets et les classes. Ils augmentent la force d'expression des langages de modélisation.

Le concepteur d'un pattern propose une solution générique pour un problème donné, qui est souvent rencontré dans divers développements. Cette solution est présentée sous une forme indépendante d'un langage de programmation particulier, ce qui oblige de l'implémenter pour une application donnée.

Il existe un nombre important de patterns. Ils apportent généralement des solutions aux problèmes critiques. Mais une bonne compréhension de ces derniers est nécessaire afin de ne pas les utiliser dans de mauvaises conditions.

Ils sont généralement représentés sous forme de diagramme UML [ROQ06].

Lors de la mise en place d'une application, on est amené à utiliser un certain nombre de patterns. Nous allons maintenant voir les patterns qui ont été employés dans le cadre de la mise en place de l'application merchandising :

- MVC 2
- DAO
- Abstract factory
- Singleton

MVC 2

MVC est l'abréviation de **M**odel **V**iew **C**ontroller. Le modèle MVC est plutôt un Architectural Pattern qu'un Design Pattern. Il concerne l'architecture globale d'une application et non pas une de ses fonctionnalités.

MVC organise une application interactive en trois modules séparés. Un premier module permet de gérer le modèle de l'application, la représentation des données et la logique du métier. Un second module permet de gérer les vues qui présentent les données à l'utilisateur et recueillent ses saisies. Le dernier module est un contrôleur qui achemine les requêtes et les flux de contrôle.

En utilisant dans nos développements le Framework Struts, nous nous intéresserons à la version 2 de ce modèle *(Figure 24)* qui a été proposée pour la première fois par Sun Microsystem : MVC 2. La différence entre les deux versions est essentiellement due au fait que dans la seconde version le contrôleur est unique, alors que dans la première il existait une multitude de contrôleurs.

Dans l'approche MVC 2,

Le **M**odèle correspond à des JavaBean

Les **V**ues sont des pages JSP

Le Contrôleur est un servlet unique

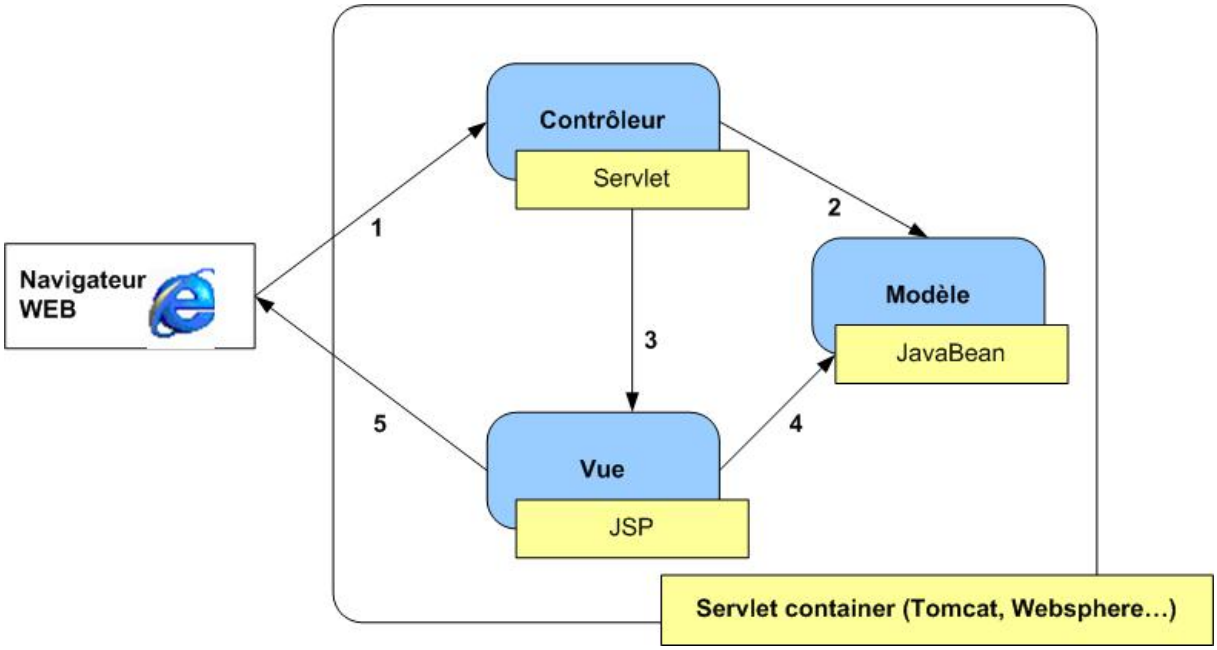


Figure 24 : MVC version 2 adapté au développement Web

1. L'utilisateur envoie une requête à l'application à l'aide de son navigateur. Il déclenche une action.
2. Le **C**ontrôleur reçoit la requête, l'analyse et effectue l'opération sur le **M**odèle.
3. Il fournit ce **M**odèle à la **V**ue.
4. La **V**ue se met à jour en utilisant le **M**odèle.
5. La **V**ue est envoyée à l'utilisateur en réponse à sa demande initiale.

DAO (Data Access Object)

Le pattern DAO permet de regrouper les différents modes d'accès aux données persistantes : stockées en base de données, dans des fichiers, dans des annuaires … Ces modes d'accès sont gérés dans des endroits bien distincts de l'architecture et non pas dispersés dans toute l'application.

De cette manière, le changement du mode de stockage ne remet pas en cause le reste de l'architecture. Mais cette souplesse a pour inconvénient de rendre la mise en oeuvre des appels plus complexe.

Le schéma suivant présente le pattern *(Figure 25)*.

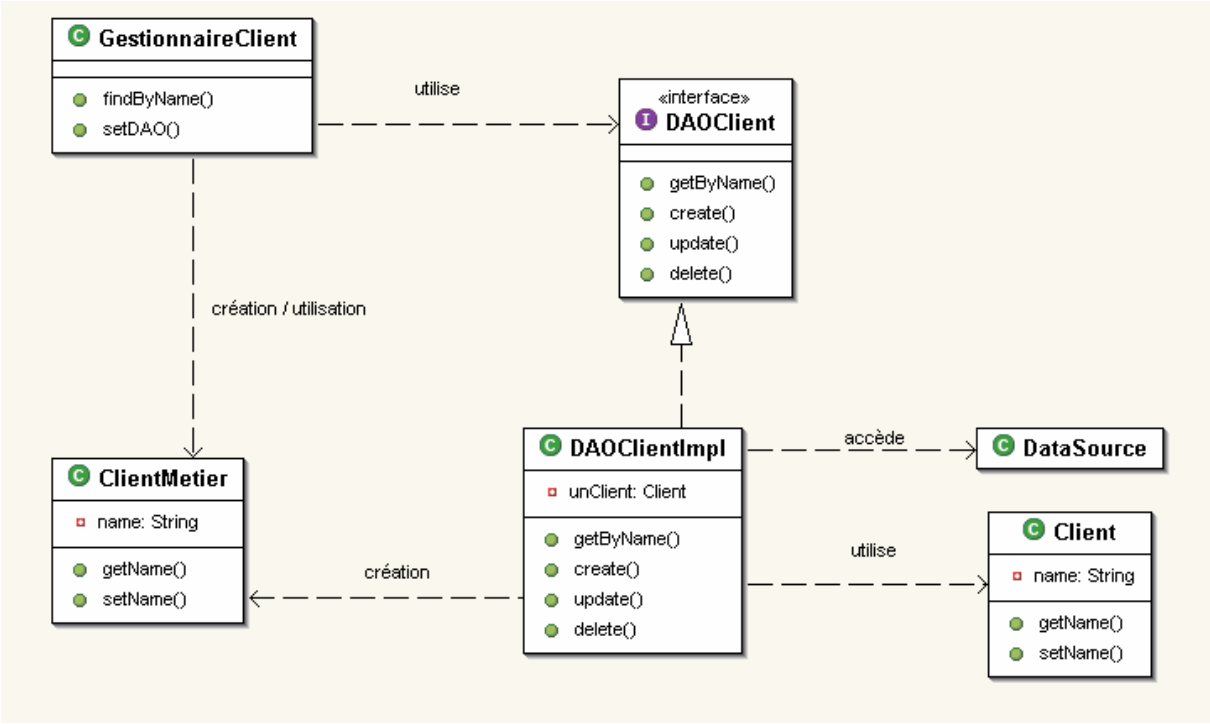


Figure 25 : Design pattern DAO (Data Access Object)

Abstract Factory (Fabrique Abstraite)

Le pattern Fabrique Abstraite fournit une interface pour créer des familles d'objets apparentés ou dépendants sans avoir à spécifier leurs classes concrètes *(Figure 26)*.

Le client est découplé de tous les détails des produits concrets.

Les fabriques concrètes implémentent les différentes familles de produits.

Pour créer un produit, le client utilise l'une de ces fabriques.

Il n'a donc jamais besoin d'instancier un objet produit.

Les méthodes de la fabrique permettant de récupérer un produit doivent renvoyer le type d'interface et pas le type de la classe d'implémentation.

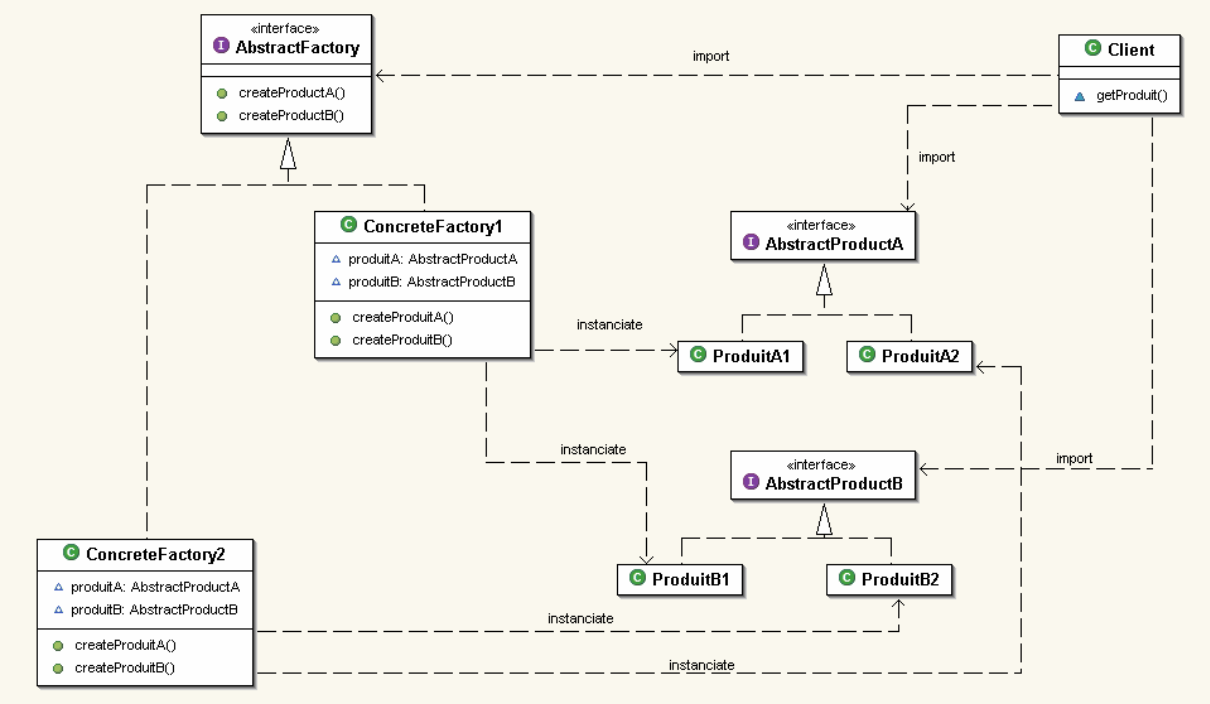


Figure 26 : Design Pattern fabrique abstraite

Singleton

Le singleton est une classe pour laquelle on ne souhaite créer qu'une seule et unique instance pendant toute la durée de l'exécution de l'application. Pour cela, on limite les accès au constructeur qui est déclaré comme privé (private). On implémente une méthode static nommée par convention getInstance() pour obtenir une instance de la classe.

Frameworks

Un framework est une ossature générale pour le développement d'une application dans un domaine particulier.

Les frameworks facilitent le travail du développement en fournissant un squelette d'application qu'il suffit de remplir pour l'adapter à ses besoins. Un framework est une surcouche de tous les besoins génériques. On est obligé de supporter un grand nombre de choses même si l'utilisation que l'on en fait est réduite. De plus, du fait de l'ajout de cette surcouche, le temps pris par le développeur pour monter en compétence sur l'environnement sera plus important.

Struts

Struts est un projet Open Source très populaire. De nombreuses entreprises l'utilisent. Il existe énormément de livres, articles, tutoriaux traitants de Struts. Les développeurs l'utilisant bénéficient donc d'une base d'information importante.

Struts peut être assimilé à une surcouche de Servlet/JSP. Il comble une bonne partie des lacunes du modèle Servlet/JSP.

Il s'appuie sur MVC version 2. Il met à disposition un ensemble d'outils destinés à organiser de façon simple le modèle MVC au sein d'une application WEB. Il permet de centraliser la navigation du site à un seul endroit. Le comportement de la totalité de l'application WEB est donc configuré dans un seul fichier. Ce fichier, nommé **struts-config.xml**, est au format XML et est placé dans le répertoire WEB-INF de l'application.

Ce fichier manipule 3 types d'éléments:

- **Les ActionForms :**

  Elles étendent la classe **org.apache.struts.action.ActionForm.** Elles servent essentiellement de contenant et sont utilisées par le **C**ontrôleur et les **V**ues. On peut les considérer comme étant des "JavaBeans" spécialisés.

- Les **Actions** :

  Elles étendent la classe **org.apache.struts.action.Action**. Elles héritent d'une méthode **execute(...)** dans laquelle est placé le code à exécuter. Il existe des actions pré-définies pour effectuer des traitements courants tels que les uploads de fichier.

- Les **Forwards** :

  Ils étendent la classe **org.apache.struts.action.ActionForward**. Lorsque l'action a terminé ses traitements, ils permettent de déterminer la page vers laquelle l'utilisateur doit être redirigé.

Cycle requête /réponse

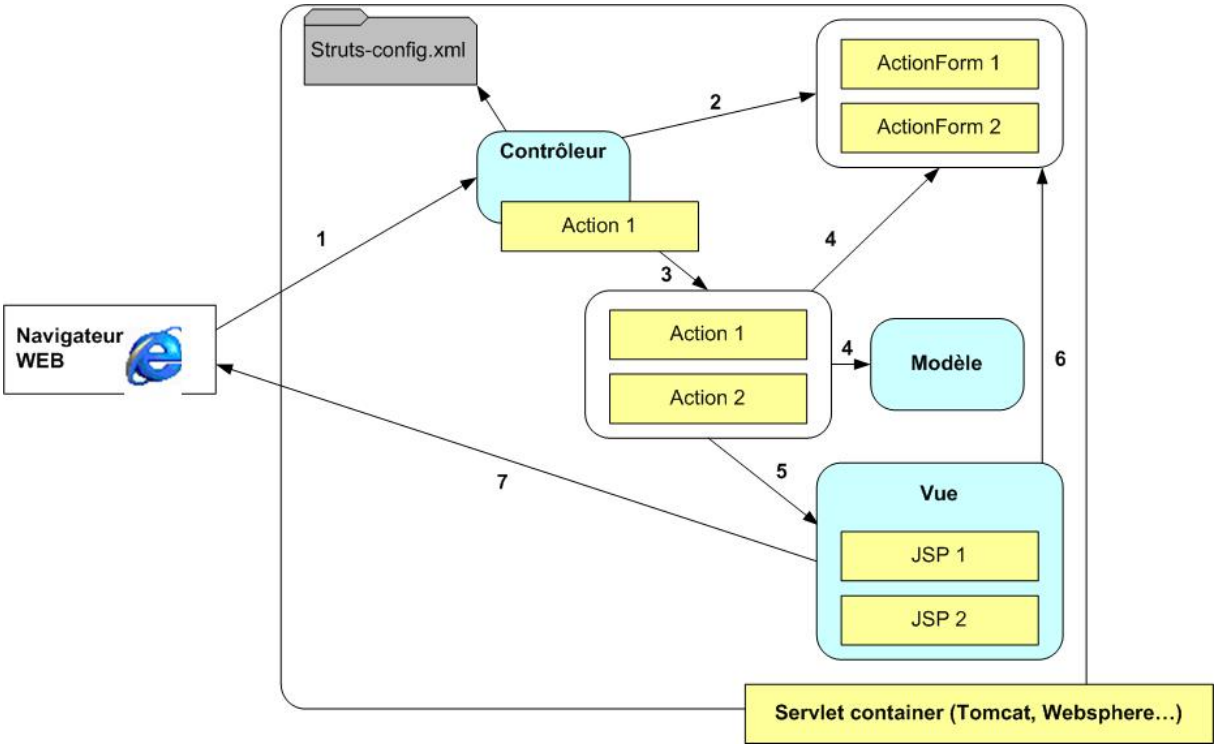


Figure 27 : Cycle requête/réponse classique

Nous allons maintenant voir un cycle requête/réponse classique avec Struts *(Figure 27)*:

1. L'utilisateur envoie une requête HTTP à l'aide de son navigateur. Le contrôleur unique, qui est en fait un servlet, intercepte la requête HTTP.
2. Le contrôleur sélectionne un ActionForm. Les données de la requête HTTP permettent alors de renseigner l'ActionForm.
3. Le contrôleur sélectionne une action à exécuter et lui fournit l'ActionForm.
4. L'action utilise l'ActionForm pour mettre à jour le modèle.
5. En fonction du résultat, l'action sélectionne la page JSP à retourner. Elle fournit ensuite l'ActionForm correspondant à la page JSP.
6. La page JSP se génère en utilisant l'ActionForm.
7. L'utilisateur reçoit le résultat de sa demande.

Les Tiles

Dans une application Web, l'ensemble des pages possède en général un élément central variable entouré d'éléments plus ou moins statiques : barre de navigation, en-tête, colonnes à gauche …

La bibliothèque Tiles est un plug-in intégrable à Struts. Elle permet de créer des modèles de page réutilisables pour l'ensemble de l'application à l'aide d'un document XML central. Cette bibliothèque peut également être utilisée indépendamment de Struts.

Hibernate

Hibernate est un framework Open Source permettant de gérer la couche d'accès aux données [BAU05] [PAT05] [ARN05]. Il est complémentaire au framework Struts qui lui gère l'Interface Homme Machine.

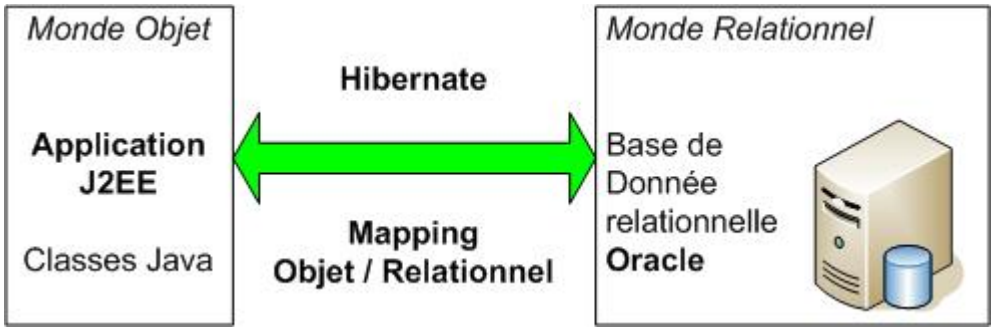


Figure 28 : Schéma simplifié du fonctionnement d'Hibernate

Il facilite le travail entre les deux univers que sont l'orienté objet et la base de données relationnelle *(Figure 28)***.** Il joint ces deux univers, à travers le mapping objet/relationnel. Le mapping objet/relationnel ou ORM sert à faire le lien entre la représentation objet des données et sa représentation relationnelle.

Hibernate s'occupe du transfert des classes Java dans les tables de la base de données. Il permet également de mettre en correspondance des types de données Java avec les types de données de la base de données relationnelle. De plus, il propose des moyens de requêter et de récupérer les données.

Hibernate est assimilable à une surcouche de JDBC qui permet d'ajouter une dimension objet.

Outils de développement

Eclipse

Au sein d'un outil unique, les IDE (Environnements de Développement Intégrés) donnent la possibilité d'écrire le code, de l'exécuter et de le débuguer. Ils peuvent également permettre la conception d'application de façon graphique.

Pour développer en Java, on utilise en générale Eclipse. Eclipse est un projet Open Source. IBM l'a développé pour ses outils de développement et l'a ensuite offert à la communauté. L'objectif était la mise en place d'un outil capable de faire du développement dans de nombreux langages (Java, C++, COBOL), mais aussi de couvrir d'autres activités telles que les développements liés aux bases de données, à la conception UML... Cette polyvalence est possible du fait qu'Eclipse est basé sur une architecture modulaire. Les différents modules ou plug-ins sont réalisés par la communauté ou par des entreprises commerciales. La plate-forme présente ainsi l'avantage d'être facilement évolutive.

Les modules manipulent les fichiers correspondant aux différents projets. Ces fichiers sont regroupés dans l'espace de travail (Workspace). Le Plan de travail (Workbench) permet, quant à lui, de voir des perspectives donnant des visions particulières du projet. Dans chaque perspective, des vues et des éditeurs permettent de travailler.

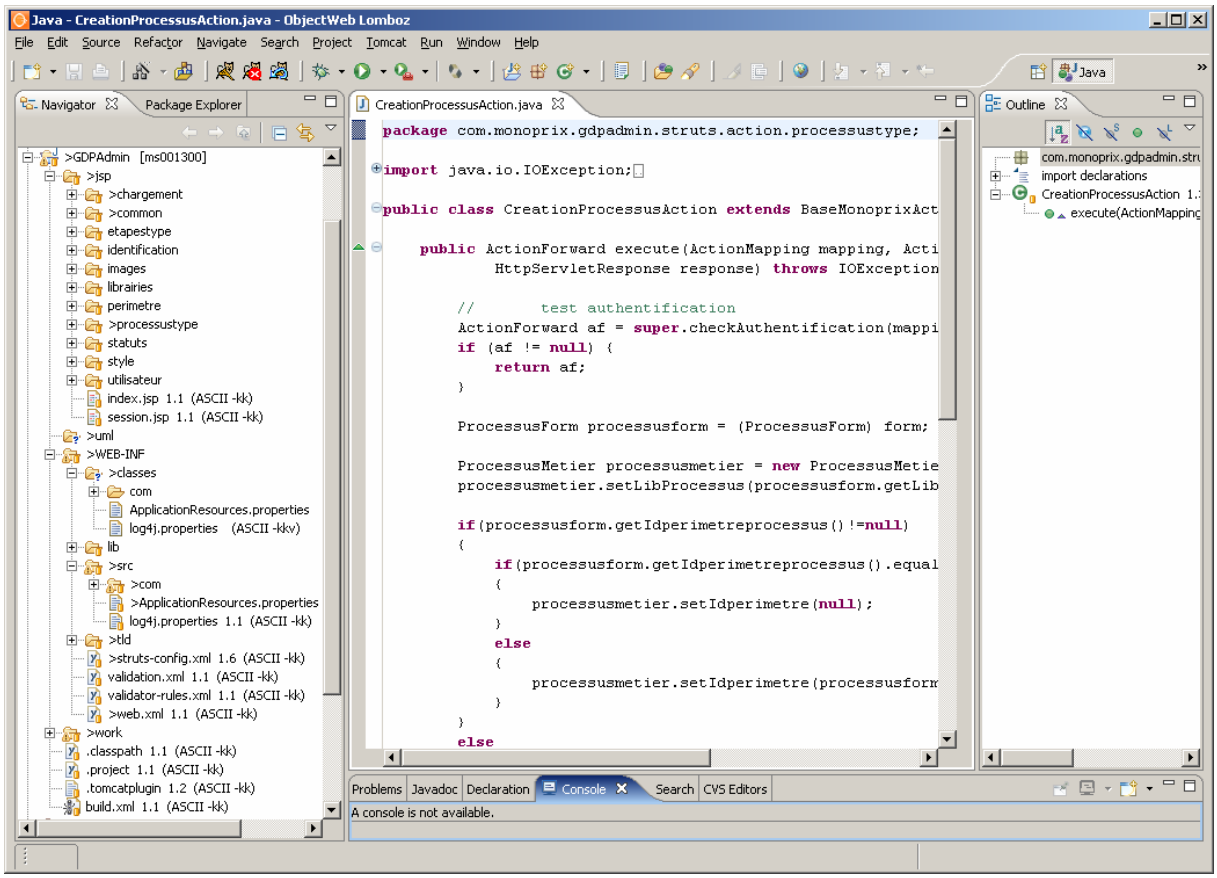


Figure 29 : Exemple de perspective : perspective Java

Pour le développement, Eclipse propose de nombreuses fonctionnalités telles qu'un « refactoring » très puissant, la « complétion » de code, des assistants...

Son ergonomie est configurable : différentes perspectives sont proposées selon les activités à réaliser.

Les exécutions des applications se font dans une JVM dédiée que l'on peut choisir. Il est possible d'utiliser un débogueur permettant de positionner des points d'arrêts conditionnels, de visualiser et de modifier des variables, de changer le code à chaud …

Des outils Open Source tel que CVS, Ant, Junit peuvent être utilisés avec Eclipse.

Cet IDE est très consommateur de ressources. Il est donc nécessaire d'avoir une machine possédant un processeur rapide et beaucoup de RAM.

Les modules

Nous allons voir quelques-uns des plug-in que nous utilisons couramment :

- Tomcat Sysdeo

Ce plug-in est proposé par la société Sysdeo. Il facilite le développement d'applications utilisant Tomcat. Il permet le démarrage et l'arrêt de Tomcat, le packaging d'une application sous la forme d'un fichier .war et son déploiement dans Tomcat.

- Lomboz

C'est un plug-in Open Source. Il facilite le développement d'applications J2EE : applications WEB à base de servlets et JSP, d'EJB et de services WEB.

Il utilise plusieurs outils Open Source pour mener à bien différentes tâches : Ant, Xdoclet, Axis... Il couvre ainsi le cycle de développement des applications J2EE : rédaction, génération du code, déploiement et déboguage.

- Eclipse UML

Eclipse UML permet d'utiliser UML dans Eclipse. Il est proposé par la société Omondo.

- Hibernate synchroniser

Ce plug-in permet de générer les objets et la structure de la base de données automatiquement à l'aide des fichiers de mapping Hibernate.

Les outils

Nous utilisons avec Eclipse un certain nombre d'outils tels que :

- Ant

Ant est un projet du groupe Apache-Jakarta. C'est un outil écrit en Java permettant de construire des applications [REN05]. On peut ainsi effectuer la compilation de code en ayant la possibilité d'exécuter des tâches avant et après la compilation.

On peut ainsi automatiser les opérations répétitives présentes tout au long du cycle de développement d'une application : développements, tests, recettes, mises en production.

Ant est indépendant de toute plate-forme. Il est aussi très efficace pour des développements simples. Il repose sur un fichier de configuration XML. C'est dans ce fichier que l'on décrit les différentes tâches à exécuter.

Un certain nombre de tâches courantes sont disponibles. En respectant certaines spécifications, on peut créer ses propres tâches à l'aide d'objets Java.

Le fichier de configuration *(Figure 30)* contient un ensemble de cibles (targets). Chaque cible contient une ou plusieurs tâches. Il est possible de mettre en place des dépendances d'exécution entre les cibles.

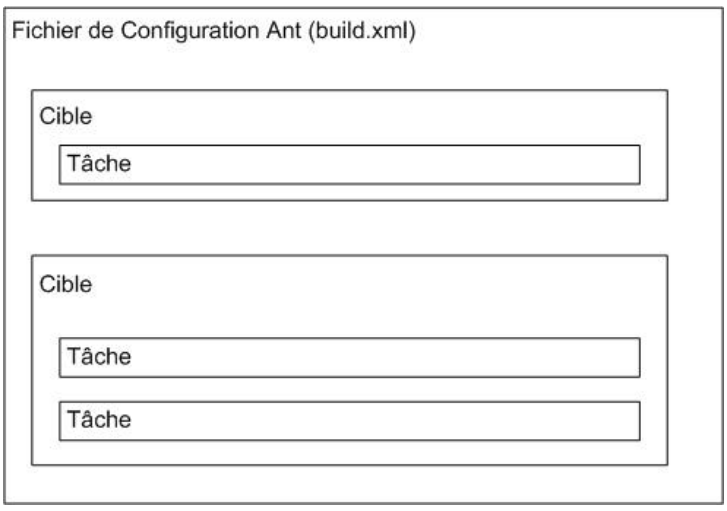


Figure 30 : Structure du fichier de configuration de Ant

Ant est donc un outil multi-plateforme, configurable grâce à son fichier de configuration XML, Open Source et extensible.

- Cvs

CVS signifie Concurrent Versions System. C'est un outil libre de gestion de versions. Initialement développé pour Unix, il est aujourd'hui disponible sous Windows. Il utilise un

référentiel (repository) afin de stocker les données. Une version contient un ensemble de ressources ayant une révision particulière.

CVS ne verrouille pas ces ressources si deux développeurs mettent à jour une même ressource, la fusion des deux versions doit être effectuée par l'un des deux développeurs.

Eclipse propose une perspective pour utiliser CVS.

- Junit

Junit est un Framework de test. Il permet l'automatisation de la réalisation des tests unitaires sur du code Java. Ainsi, le code est de meilleure qualité et plus fiable. Du fait de l'automatisation, les tests de non-régression peuvent être faits de façon plus systématique.

L'utilisation de Junit n'est pas intrusive. Le code à tester n'est pas modifié. Les traitements représentés et leurs tests sont bien séparés.

L'unité de test est une classe dédiée qui regroupe des cas de tests. Ces cas de tests créent un objet de la classe à tester et tout autre objet nécessaire aux tests. Ensuite, il appelle une méthode de l'objet avec les arguments correspondant au cas du test. Le résultat obtenu est comparé au résultat attendu. En cas d'erreur, une exception est levée.

- Log4j

Avoir des fichiers de Log est très important dans l'environnement de développement, pour faciliter le déboguage, et dans l'environnement de production pour avoir des traces utilisables en cas de problèmes d'exploitation.

Pour cela on peut utiliser Log4j. C'est un utilitaire de gestion des journaux qui s'intègre aux programmes. Il appartient au projet Jakarta de la fondation Apache.

Présentation du projet BIRT

BIRT signifie Business Intelligence and Reporting Tools. Ce projet open-source très actif et libre d'utilisation permet la création de rapports Web. Il a été initié par la société Actuate. Il se compose essentiellement d'un outil de conception de rapports basé sur Eclipse et d'un moteur d'exécution qui s'installe dans un serveur d'applications J2EE (*Figure 31*).

L'outil de conception permet de construire graphiquement les rapports. Un fichier au format XML décrit chaque rapport.

Les rapports sont construits par l'assemblage de différents composants graphiques tels que des textes, images, tableaux, listes, graphiques. Ils peuvent être pré-visualisés au format HTML ou PDF.

Les données, permettant de construire les rapports, peuvent être statiques ou dynamiques. Dans le dernier cas, elles sont extraites de « jeux de données » ou calculées à l'aide de formules et de scripts JavaScript.

Afin d'extraire des données à partir de « jeux de données », une vue « Explorateur de données » permet de déclarer des sources de données. Ensuite, les « jeux de données » sont constitués grâce à une requête SQL.

La mise à disposition d'un rapport se fait en déployant le rapport sur un serveur d'applications J2EE dans lequel le moteur d'exécution BIRT est installé.

Des URL permettent alors d'accéder aux rapports. Les paramètres fournis dans les URL servent à préciser le format de génération (PDF, HTML) ou à alimenter les paramètres utilisés dans les requêtes SQL.

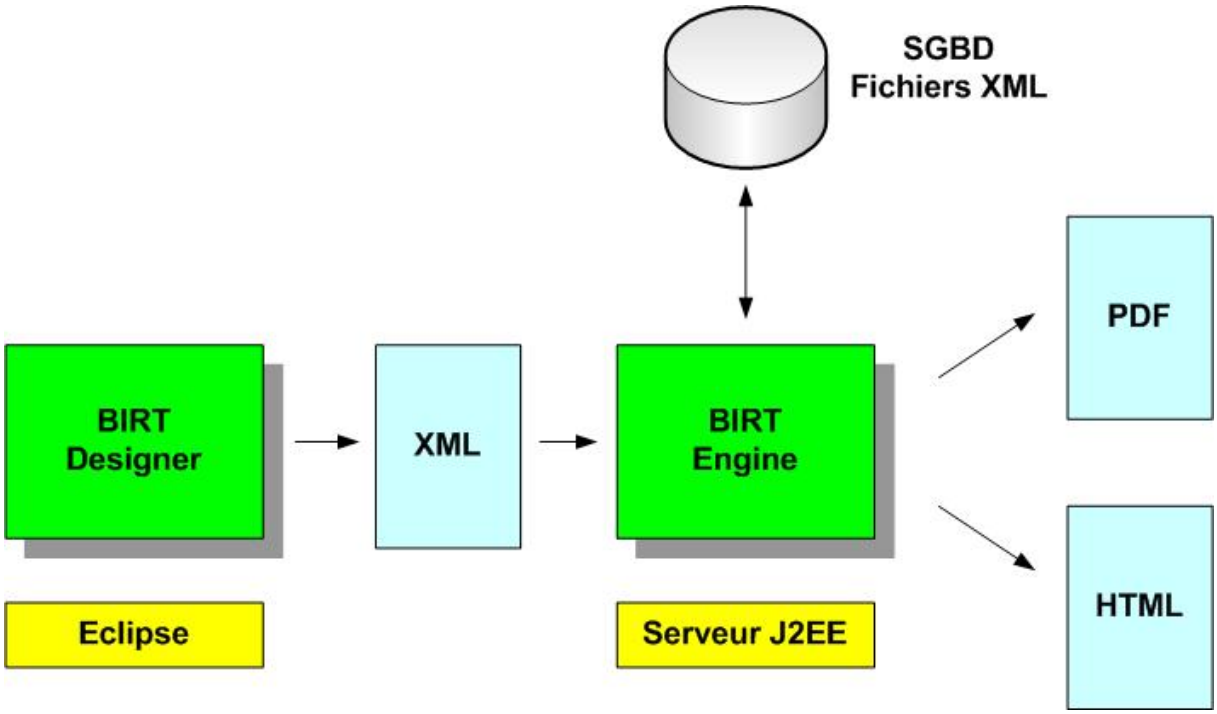


Figure 31 : Architecture de BIRT

2.6. Ruby on Rails

Ruby est un langage objet interprété et typé dynamiquement. Ruby utilise un gestionnaire de package appelé gems.

Rails est un framework basé sur Ruby. Les développements Rails reposent sur des principes tel que le fait de ne pas se répéter. De plus, Rails utilise des conventions plutôt que des fichiers de configuration.

On utilise souvent l'abréviation RoR pour Ruby on Rails.

Architecture et fonctionnement

L'architecture d'une application Ruby on Rails est simple et fixe (*Figure 32, Figure 33, Figure 34*).

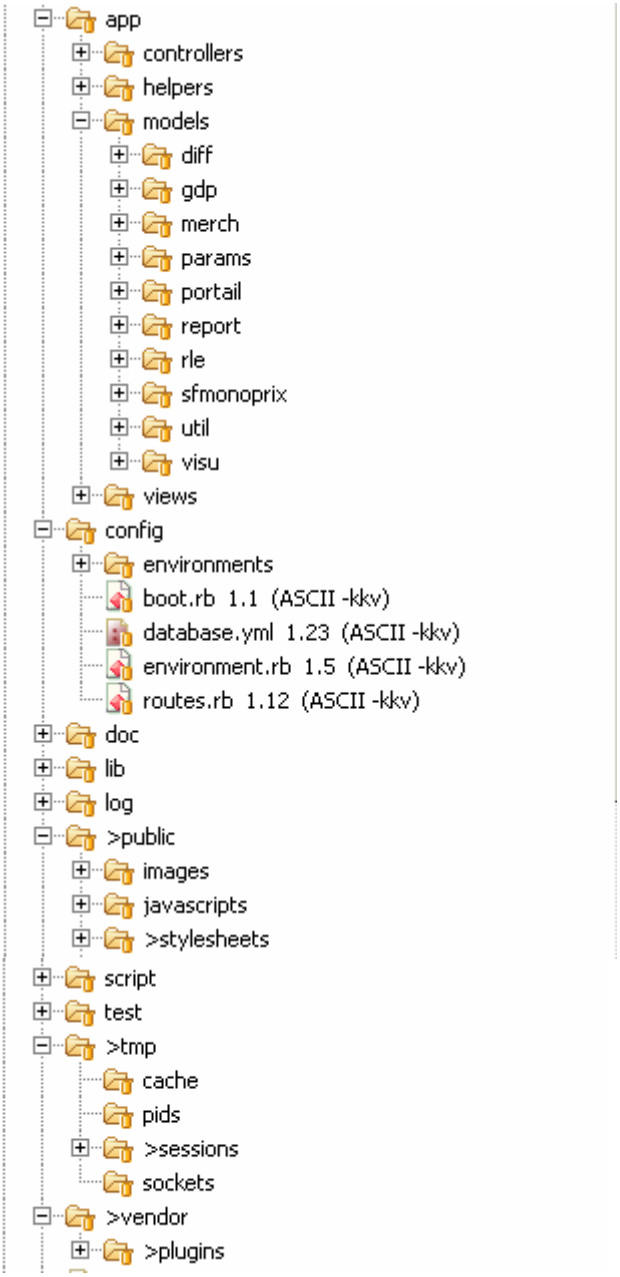


Figure 32 : Architecture d'une application RoR

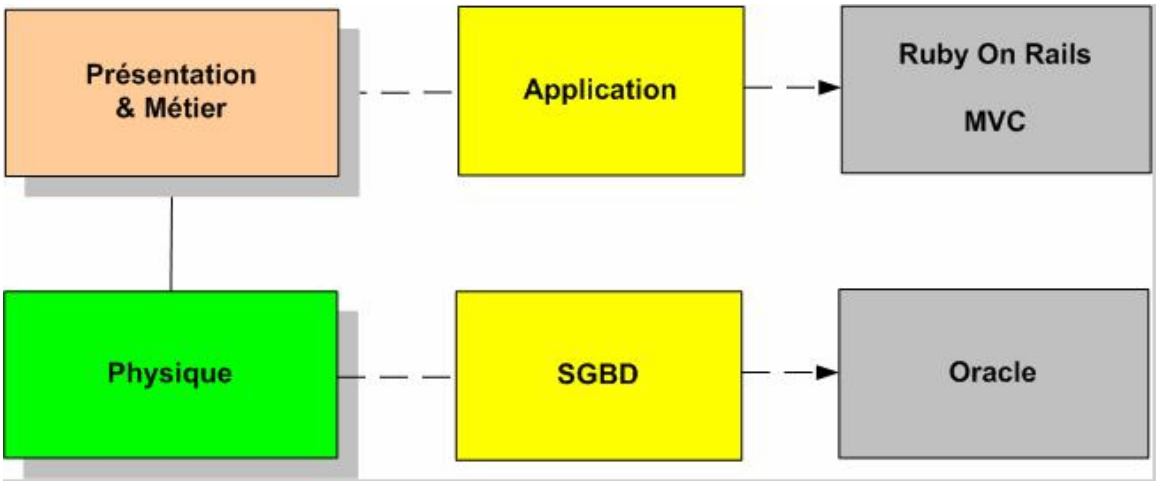


Figure 33 : Architecture en deux couches - RoR et la base de données

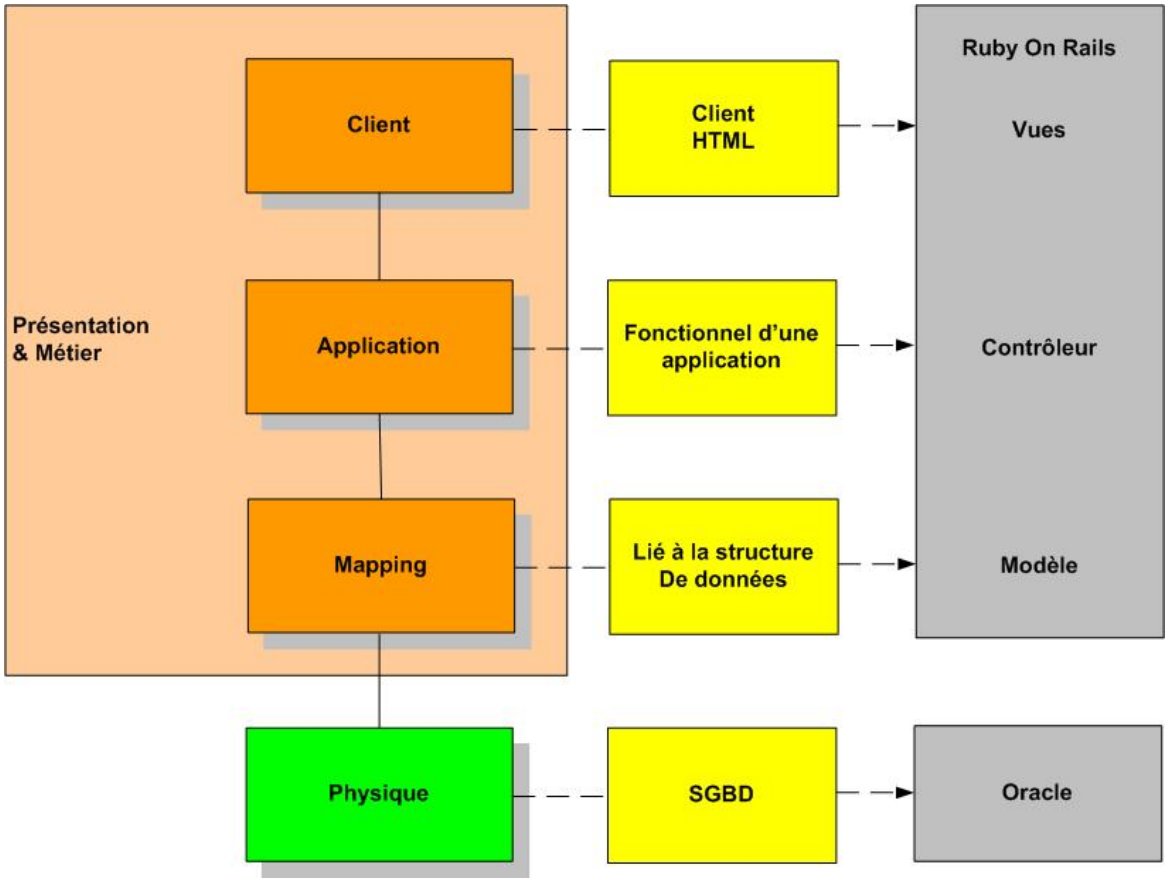


Figure 34 : Architecture en deux couches – détail de la couche applicative

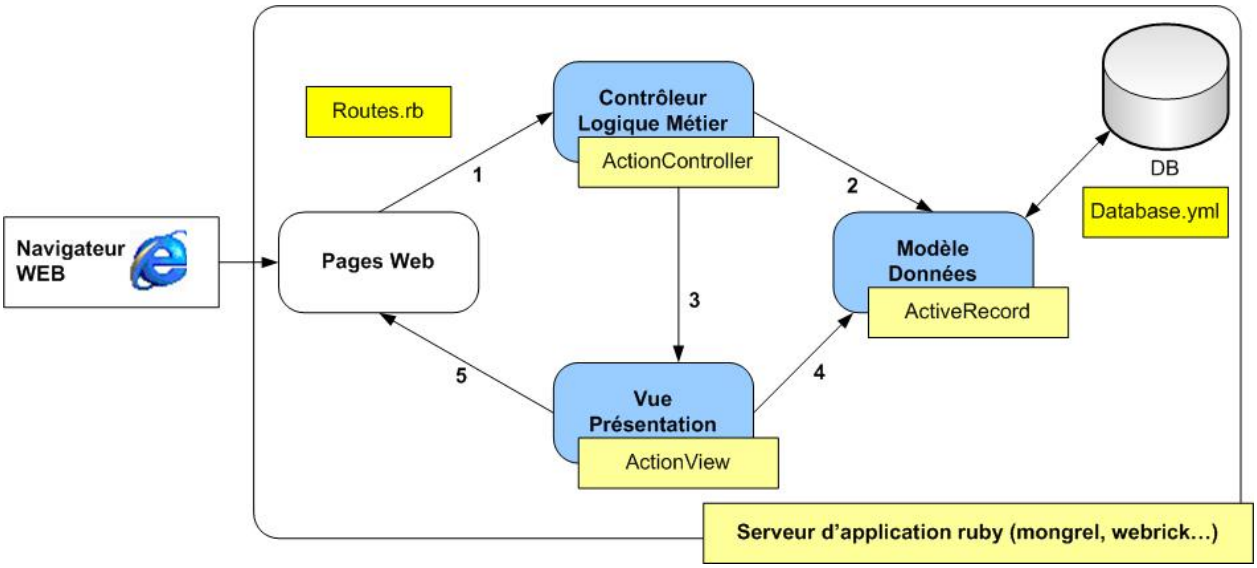


Figure 35 : Architecture Rails

Cette architecture est basée sur modèle MVC (*Figure 35*). Des classes ActiveRecord, ActionController et ActionView permettent respectivement d'implémenter les modèles, contrôleurs et vues.

Les principales étapes du cycle requête réponse sont :

1. suite à la requête http de l'utilisateur, le contrôleur est sollicité
2. le contrôleur accède ensuite au modèle afin de récupérer les informations dont il a besoin
3. une vue est rendue
4. la vue peut accéder directement au modèle si nécessaire
5. la réponse HTML est ensuite retournée au navigateur

Un fichier de routage routes.rb permet de faire la correspondance entre une URL et un contrôleur.

L'URL est terminée par le nom du contrôleur, le nom de l'action et un paramètre qui identifie par exemple l'enregistrement à mettre à jour.

Une URL a la forme suivante :

http://localhost:3000/nom_du_controller/nom_action/id.

Rails repose sur une architecture de transitions.

REST consiste à mener une politique simple mais rigoureuse au niveau des URIs en se reposant sur HTTP. Les ressources sont alors classées. C'est une architecture orientée ressource. Une technologie est dite RESTFULL lorsque l'on fait du CRUD sur une base de données en utilisant les méthodes HTML post, get, put et delete.

Si on définit des appels spécifiques, différents de ceux d'HTTP, on s'oriente vers une architecture de type RPC. Ce protocole permet de faire des appels de procédures sur un ordinateur distant. Les architectures orientées services reposent sur ce concept.

L'architecture propose par défaut trois environnements : développement, test et production.

Un fichier database.yml au format YAML permet de configurer les accès aux bases de données.

Dans cette architecture, il est possible d'utiliser les serveurs d'applications Ruby tels que Webrick et Mongrel. Webrick est utilisable dans un environnement de développement, alors que Mongrel est utilisé en production.

Framework et plug-in

Le framework Rails repose sur l'emploi de conventions qui permettent d'économiser du code.

Modèle

La gestion du mapping ORM est complètement transparente. Il est possible, en complément du mapping proposé par défaut, de déclarer des relations entre les tables (one to one, one to many ou many to many)

Par défaut, de nombreuses validations sont fournies afin par exemple de tester l'unicité d'une clef dans une table au niveau des SGBD, de tester la présence, la taille, le format, le type d'un champ ou la validation d'une checkbox.

Afin d'effectuer des recherches sur le modèle, on utilise la fonction « find » qui permet de faire une requête et de récupérer un résultat.

Vue

Il existe trois types de vues :

- Les Layout qui permettent de configurer l'emplacement des différents objets graphiques
- Les vues qui correspondent à une page Web
- Les vues partielles qui correspondent à des portions de page Web

Le code HTML peut être généré et le code de plusieurs vues peut être factorisé à l'aide de classes spécifiques appelées Helpers.

Les vues sont implémentées à l'aide de fichiers ayant l'extension rhtml. Ces fichiers se composent de code HTML et Ruby.

Il est possible d'insérer le code à l'aide des balises <% code %>.

Des structures de contrôle « if » ou « for » sont également disponibles.

Une variable peut être affichée à l'aide de la syntaxe suivante : <% @var %>.

De nombreux tags sont utilisables afin de gérer :

- les méta-données (javascript_includ_tag, stylesheet_link_tag)
- le HTML (link_to, image_tag, form_tag)
- les formulaires (text_field, select_tag, submit_button)

Contrôleur

Les contrôleurs héritent tous d'un contrôleur père : l'ApplicationController. Tout comme pour les vues, il est possible de mutualiser du code dans des classes Helpers.

De nombreux types de variables sont gérés : session, params, flash, request, response.

Le type flash permet de stocker l'information le temps d'un rendu (une étape response) pour envoyer des messages à l'IHM.

Des filtres peuvent être positionnés avant ou après les actions. Ils permettent d'effectuer des restrictions ou d'exclure des actions. Les filtres sont ordonnancés d'une façon séquentielle.

Par défaut, à la fin d'une action, on affiche la vue portant le même nom que l'action. Si on le désire, il est possible de spécifier une autre vue.

Mise en œuvre simplifiée

Grâce aux outils mis à disposition dans Ruby on Rails, il est possible d'implémenter de façon simple des Web Services, des fonctionnalités AJAX ou d'envoyer des emails.

Autres outils

Afin de générer plus rapidement le html et le css, on peut utiliser des outils tels que Haml et Sass.

Pour mettre en place l'internationalisation, il existe des plug-in tels que Globalize ou simple localization.

Les développements Ruby on Rails peuvent être effectués à l'aide d'Eclipse. Il existe de nombreux plug-in à cet effet.

2.7. L'architecture Web

Les serveurs d'applications sont nécessaires à l'implémentation d'une architecture J2EE. Ils constituent des environnements permettant l'exécution côté serveur d'applications. Ils fournissent un ensemble de fonctionnalités pour que des clients puissent utiliser une seule et même application.

Ils gèrent les sessions correspondant aux différents utilisateurs. Ainsi, des informations propres aux utilisateurs, tel qu'un panier de commandes dans un site de vente en ligne, sont conservées par les serveurs d'applications. En générale, un identifiant unique est géré pour chaque client. Lors des échanges HTTP, l'identifiant est transmit dans les URL à l'aide de variables cachées ou de cookies.

Ils offrent des possibilités de montées en charge et de reprise sur incident par le biais de cluster.

Ils accèdent aux nombreuses sources de données et fournissent des mécanismes de gestion des pools de connexions, afin d'optimiser les performances de ces accès.

Les serveurs d'applications évitent de devoir tout re-développer contrairement à certaines solutions tels que les CGI.

Présentation générale

Nous allons voir les concepts et les notions liés à la mise en place de clusters.

La mise en place d'applications robustes et adaptables en cas d'augmentation du nombre d'utilisateurs nécessite l'utilisation d'un certain nombre de concepts.

Parmi eux nous pouvons citer le Clustering [PEN4a][PEN4b], le Load balancing [VIV05] qui permet de déployer une application sur un ensemble de serveurs, de façon transparente pour l'utilisateur et de manière progressive. On parle également en anglais de la notion de scalability qui correspond à l'aptitude qu'à un système de pouvoir évoluer en plusieurs paliers afin de répondre à un nombre d'utilisateurs croissant. Cela peut être traduit par : « aptitude à monter en puissance par palier ».

Nous pouvons également citer le concept de gestion de persistance des sessions [PEN4c] [PEN4d] qui permet, à l'aide d'un certain nombre de mécanismes, d'effectuer des reprises sur erreur en cas de défaillance du système.

Scalability : Système à grande échelle

Les entreprises ont aujourd'hui besoin d'applications WEB adaptables (scalable) et offrant une haute disponibilité, afin de proposer à des centaines d'utilisateurs, connectés sur un même site, un service de qualité. L'adaptabilité d'une application WEB (scalibility**)** correspond à l'aptitude à supporter un nombre croissant d'utilisateurs en ajoutant des serveurs supplémentaires à un groupe existant de serveurs (cluster).

Offrir un service à un grand nombre d'utilisateurs est une chose, mais ce service doit avant tout être fiable. C'est pour cela que sont mis en place des mécanismes permettant la haute disponibilité (HA : High Availability). Ces mécanismes fournissent de la redondance dans le système. Si un membre d'un groupe de serveurs (cluster) est indisponible pour une raison quelconque, un autre membre du groupe peut traiter les requêtes HTTP entrantes à sa place de façon transparente.

La mise en place d'un cluster adaptable et proposant des mécanismes de haute disponibilité permet d'améliorer la fiabilité du système global.

Pour échelonner un système, il faut donc pouvoir mettre en place un groupe de machines afin de minimiser les temps d'indisponibilité et de s'assurer que tous les composants du groupe sont redondants.

Clustering

Dans le cas des applications J2EE, un cluster est un groupe de serveurs d'applications installé sur une ou plusieurs machines qui exécutent une application comme si elles étaient une seule et même entité. Les serveurs d'applications représentent les membres du cluster. Il existe deux méthodes de clustering :

- **vertical scaling**

Le vertical scaling correspond à une augmentation du nombre de serveurs d'applications qui tournent sur une même machine.

- **horizontal scaling**.

L'horizontal scaling consiste pour sa part à augmenter le nombre de machines dans le cluster. L'utilisation d'un groupe de machines physiques permet de gérer un nombre croissant de requêtes. Du fait de l'utilisation de différentes machines, l'horizontal scaling est plus fiable que le vertical scaling. Dans cette méthode, même si un serveur est indisponible, un autre serveur peut continuer de faire tourner l'application.

Le clustering sert avant tout à supprimer les points de défaillance uniques (SpoF Single Point of Failure) et ainsi d'éviter les défaillances du système liées aux goulots d'étranglement.

Il existe différents types de configuration. Nous pouvons citer les trois approches suivantes:

- Indépendante

  Chacun des serveurs d'applications a ses propres fichiers systèmes et sa propre copie des fichiers de l'application. C'est la configuration que nous avons utilisée dans la solution retenue.

- Fichier système partagé

  Tous les serveurs d'applications utilisent un seul et même système de fichier afin d'obtenir une copie de l'application.

- Gérée par un serveur d'administration

  Un serveur d'administration contrôle l'accès au contenu de l'application. Il assure que tous les serveurs d'applications ont le contenu de l'application disponible. Quand une application est déployée, il met à jour le contenu sur l'ensemble des serveurs d'applications gérés. De même que lorsqu'une application est supprimée, il supprime l'application correspondante de l'ensemble des serveurs d'applications.

Nous pouvons remarquer au passage que le déploiement d'une application Web sur un ensemble de serveurs d'applications est lié au concept de Farming. Le Farming permet le déploiement à chaud d'applications Web au sein d'un cluster. Dans une Ferme, une application Web est déployée en copiant un fichier War sur un nœud unique du cluster. Le farming prend en charge le déploiement de l'application sur la totalité du cluster. De même, la suppression de l'archive se fait sur un nœud unique et le Farming doit prendre en compte la suppression de l'application sur l'ensemble des nœuds.

Design patterns

Nous allons voir deux modèles de conception liés aux systèmes basiques. Nous utiliserons ces modèles par la suite.

Le modèle topologique serveur Maître – serveur de secours : Master–backup topologie pattern

Dans ce modèle, au moins deux machines, interconnectées dans un LAN, sont configurées de la même façon, aussi bien au niveau matériel, système d'exploitation ou logiciel. Une machine est le serveur Maître. Il traite les requêtes entrantes. L'autre ou les autres machines sont des serveurs de secours. Lorsque le serveur Maître est indisponible, l'un des serveurs de secours devient Maître et continue de traiter les requêtes entrantes.

Le modèle comportemental de contournement des erreurs : Fail-over behaviorel pattern

Ce second modèle comportemental permet de passer au-dessus des erreurs. Il fait référence aux moyens avec lesquels le serveur Maître est remplacé par le serveur de secours. Le pattern master-backup n'est pas suffisant pour garantir une reprise sur erreur transparente.

En effet, le serveur de secours doit être capable de savoir ce que faisait le Maître. Ceci suppose un partage de l'information utilisée par le serveur Maître. Presque toutes les solutions de Fail-over reposent sur un mécanisme permettant de maintenir et de partager ces informations entre le serveur Maître et le serveur de secours avant que le Maître ne soit indisponible. La conservation et le partage de ces informations sont la seule façon d'assurer que le serveur de secours puise remplacer le serveur Maître durant son indisponibilité.

Dans les conteneurs de Servlet et de JSP J2EE, les informations sont généralement gérées à l'aide de sessions côté serveur. Il est alors nécessaire de mettre en place un mécanisme permettant de stocker et partager les sessions.

Tolérance de panne : Fault tolerance

La tolérance de panne (Fault tolérance) est un concept permettant de pallier l'indisponibilité du système sans que l'utilisateur final s'en rende compte. Lorsqu'un serveur d'application est indisponible, les requêtes des utilisateurs sont envoyées et traitées par un autre serveur d'application.

Nous pouvons citer deux types de tolérance de panne :

- **request-level fail over**

Si un membre du cluster tombe, toutes les requêtes correspondantes sont dirigées vers un membre actif du cluster.

- **session-level fail over**

A un client Web correspond une session gérée du côté serveur. Cette session contient des informations propres au client. Si un membre du cluster tombe, un autre membre du cluster doit être capable de continuer la session de l'utilisateur que gérait le serveur qui présente une défaillance et ceci sans discontinuité. Pour cela, il est nécessaire de répliquer les données de sessions au sein du cluster.

La mise en place de tels mécanismes permet la mise en œuvre d'applications proposant une haute disponibilité aux utilisateurs finaux.

Haute disponibilité : HA Hight Availability

La haute disponibilité a pour but d'éviter les situations dans lesquelles un système devient indisponible, suite à des problèmes liés au software ou au hardware.

Habituellement, dans les systèmes classiques, toutes les requêtes en cours d'exécution par le serveur sont perdues lors d'une défaillance. Les utilisateurs attendent le redémarrage du système avant de pouvoir retravailler en recommençant depuis le début. Dans le cas de serveurs d'applications J2EE, toutes les informations contenues dans la session de l'utilisateur sont alors perdues.

La haute disponibilité consiste à mettre en place un système capable de continuer à gérer les requêtes entrantes malgré la défaillance d'un des serveurs d'applications. Dans ce cas, le fait qu'un serveur soit défaillant est complètement transparent pour l'utilisateur final.

L'avantage de ces systèmes que l'on qualifie de systèmes de haute disponibilité est de proposer des durées de disponibilité élevées.

Nous venons de voir les concepts et les notions relatives au clustering, nous allons maintenant aborder les différentes façons de répartir la charge de travail au sein d'un cluster, ainsi que les différentes possibilités permettant le partage de sessions dans des applications J2EE, afin d'offrir une haute disponibilité.

Load Balancing

La mise en place d'un cluster nécessite le partage de la charge de travail sur les différents membres du cluster. C'est ce que permet de faire le mécanisme de Load Balancing ou la répartition de charge. C'est un mécanisme qui équilibre la charge de travail sur plusieurs nœuds d'un cluster. La façon d'effectuer cet équilibrage est généralement indiquée par une politique précise.

Il existe de nombreuses façons de mettre en œuvre la répartition de charge :

- DNS Round robin

La résolution d'un nom de domaine unique permet d'aboutir à un ensemble d'adresses IP. Les clients, faisant une demande de résolution du nom, reçoivent séquentiellement une adresse réseau différente de la part du serveur DNS.

Cette méthode présente l'avantage d'un coût de mise en place réduit. Elle est simple et facile à mettre en œuvre. Mais elle ne permet pas de fournir l'affinité avec un serveur (server affinity) ou de proposer de la haute disponibilité HA. De plus, tous les nœuds du cluster sont visibles sur le réseau, donc exposés.

Un autre inconvénient est lié au fait que pour réduire le trafic réseau et améliorer les temps de réponse, les requêtes DNS sont parfois mises en cache. Ceci a pour conséquence qu'un client peut tenter d'accéder à un serveur qui n'est plus présent dans le cluster. En effet, du fait que les requêtes DNS sont en cache, si la liste de serveur change, il faut un certain temps pour que la modification soit propagée, d'où, ce risque d'erreur.

- Hardware Load balancing

Un boîtier de répartition de charge utilise une seule adresse IP virtuelle. Il montre une adresse IP unique pour le cluster, et fait la correspondance avec les adresses de chacune des machines du cluster. Le boîtier reçoit chacune des requêtes entrantes et réécrit l'en-tête des messages TCP / IP afin de pointer sur une machine du cluster.

Cette méthode de répartition de charge présente l'avantage d'être capable de déterminer un certain nombre de métriques : le nombre total de sessions actives ou connectées à une instance, les temps de réponse, les pics de charge, leur durée …

Mais elle est chère et difficile à mettre en place du fait d'une configuration complexe de la partie Hardware. Tous les nœuds du cluster sont, tout comme dans la méthode précédente, exposés sur le réseau.

Il est par contre possible d'implémenter l'affinité de serveur et d'offrir une haute disponibilité.

Le Hardware load balancing propose également une certaine tolérance de panne. Il permet de contourner les pannes liées au niveau requête (request-level fail over). Si le load balancer détecte qu'un noeud particulier est tombé, il redirige toutes les requêtes à destination du nœud mort vers un autre nœud actif. Cependant, il ne permet pas de contourner les pannes liées à la session (session-level fail over).

- Logiciel de distribution de charge réseau

Dans cette méthode, c'est un logiciel qui prend en charge la répartition du flux entrant vers les différentes machines du cluster.

Ce type de logiciel permet de mettre en place des fonctionnalités d'équilibrage de charge avancées. On peut incorporer dans les règles d'équilibrage des métriques telles que la consommation de CPU et de RAM des différents serveurs utilisés dans le cluster et ceci en temps réel. Néanmoins, ces fonctionnalités sont généralement proposées par des solutions commerciales, telles que EDGE d'IBM, qui représentent un coût financier important en terme de licence. Il est cependant possible de trouver des produits Open Source tel que Balance ou PEN qui proposent des fonctionnalités plus basiques.

Par contre, comme la solution est logicielle et correspond donc à une couche haute du modèle OSI, elle est plus gourmande en ressources qu'une solution HardWare travaillant sur les trames TCP IP, c'est à dire sur les couches basses du modèle OSI.

- Répartition de charge Réseau

Des éditeurs tels que Microsoft proposent des solutions basées sur la répartition de charge réseau. C'est un système logiciel distribué et redondant, permettant de répartir la charge sur une ferme de serveurs. Il ne nécessite pas de répartiteur car l'ensemble des membres de la ferme reçoit les données. Ces fonctionnalités sont présentes dans la version Windows 2003 Entreprise.

Voici donc les différentes méthodes permettant de distribuer la charge de travail aux différents membres d'un cluster. C'est l'une des conditions essentielles à la mise en place d'un cluster. Mais un mécanisme de répartition de charge ne permet pas de mettre en place des applications proposant une haute disponibilité. Pour cela, il est nécessaire d'avoir des mécanismes permettant de partager les informations relatives aux clients sur l'ensemble des membres du cluster. C'est l'objectif des mécanismes de persistance des sessions.

Persistance des états des sessions : Session State Persistance

Le but principal de la réplication de session est de ne pas perdre les détails de la session d'un utilisateur dans divers cas de figure. Dans le cas où un membre du cluster tomberait ou serait stoppé afin de mettre à jour l'application ou afin d'effectuer la maintenance du système, l'utilisateur pourrait récupérer les informations correspondant à sa session sur un autre serveur que celui sur lequel il avait initialement ouvert sa session.

Deux possibilités sont offertes afin de gérer les sessions des utilisateurs:

- l'affinité de session (session affinity ou sticky session)

Un membre n'a pas connaissance des états des sessions des autres membres du cluster. La session utilisateur est gérée par un unique membre du cluster, sélectionné par le mécanisme de load balancing. Il n'y a donc pas de réplication des informations correspondant à la session.

- La réplication de session

Chaque membre du cluster connaît l'état des sessions des autres membres du cluster. L'état des sessions d'un membre est propagé à l'ensemble des autres membres du cluster de façon périodique.

Il existe trois moyens d'implémenter la persistance de session :

- La réplication des sessions en mémoire

Les attributs des sessions sont propagés à l'ensemble des membres du cluster de façon systématique. Cette solution convient pour des clusters de petite taille.

- La persistance des sessions dans un système de fichiers

Les informations de session sont stockées et lues à partir d'un fichier centralisé.

- La persistance des sessions en base de données

  Les informations relatives à la session sont stockées dans une base de données.

Solution Open Source Apache Tomcat 5

Clustering Tomcat

Tomcat est un conteneur de JSP et de Servlet [CHO04]. Il est aujourd'hui couramment utilisé en environnement de production.

Il supporte l'horizontal scalibility qui permet de répondre à un nombre important de requêtes utilisateurs par l'ajout de serveur.

En général, l'échelonnage d'une application correspond à un coût important. Il est nécessaire d'investir dans des systèmes multi CPU et des extensions de mémoire très coûteuses. Cette approche correspond à ce que l'on appelle le Scaling up.

Le clustering Tomcat utilise des interconnexions LAN afin de partager des ressources de plusieurs serveurs. Cette approche correspond au Scaling out ou Horizontal Scaling. Elle se base sur une ferme de serveurs, moins coûteuse qu'un unique serveur multiprocesseur.

Le clustering de serveur Tomcat permet d'éviter de rendre le système indisponible en cas d'une augmentation du nombre de requêtes utilisateurs et de s'assurer que les utilisateurs ne perdent pas leurs données. Les multiples serveurs Tomcat sont vus comme un unique serveur par les utilisateurs.

Voici, ci-dessous *(Figure 36)*, l'architecture d'un cluster Tomcat.

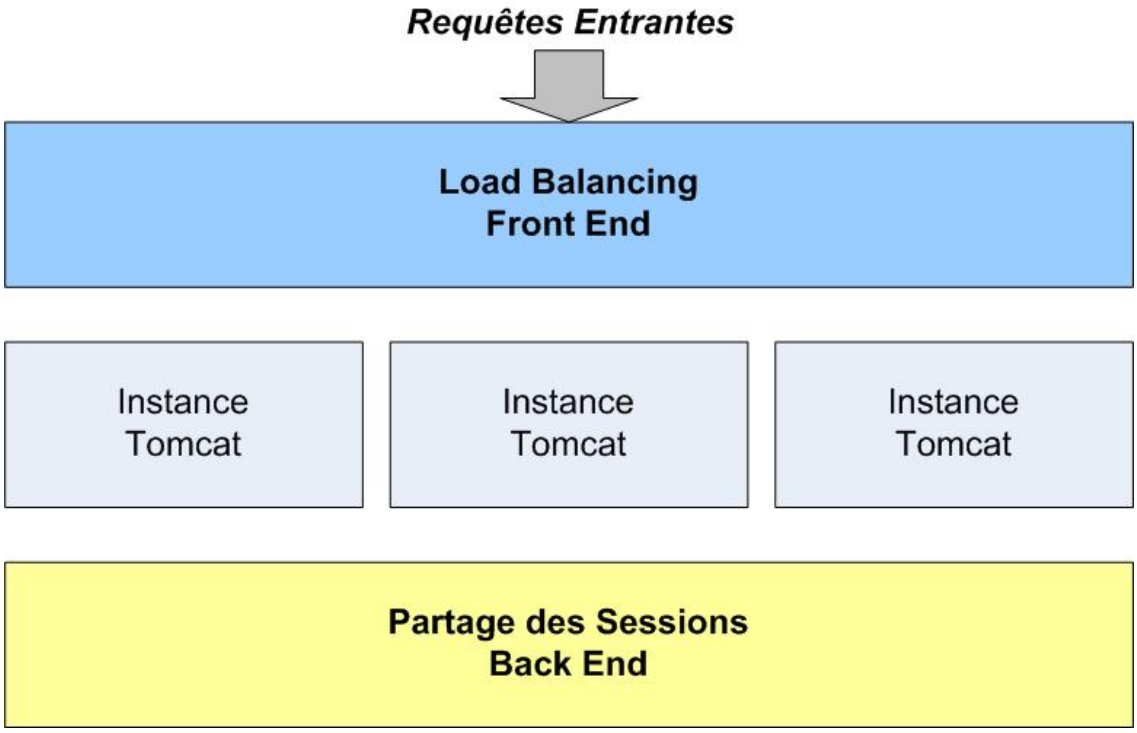


Figure 36 : Architecture d'un cluster Tomcat

Cette architecture se compose :

- d'un front End

Cette couche permet de distribuer les requêtes entrantes sur les différentes instances Tomcat. La distribution du travail aux différents serveurs est assurée par l'équilibrage de charge (Load Balancing).

- d'un ensemble d'instances Tomcat

Cette couche correspond aux instances Tomcat.

- d'un back End

Cette couche s'assure de rendre disponible les données de session aux différentes instances. Ces informations sont synchronisées de façon périodique.

Load balancing

Dans une architecture sans Load balancing où une unique instance de Tomcat est associée à un serveur WEB Apache, en cas d'augmentation de la charge des requêtes entrantes, l'instance Tomcat surchargée va tomber par manque de ressources.

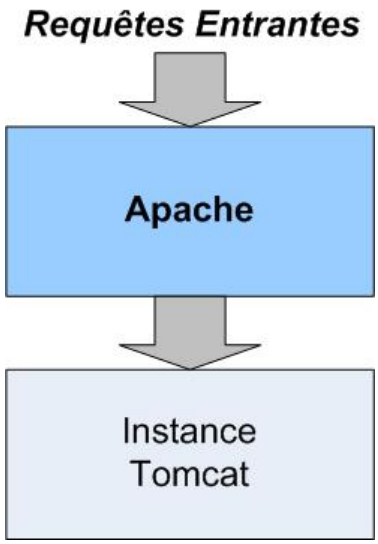


Figure 37 : Architecture avec un serveur Apache et un serveur Tomcat

Lorsque l'on met en place une architecture basée sur un cluster comportant 3 instances Tomcat *(Figure 38)*, le cluster peut répondre à beaucoup plus de requêtes avant de tomber.

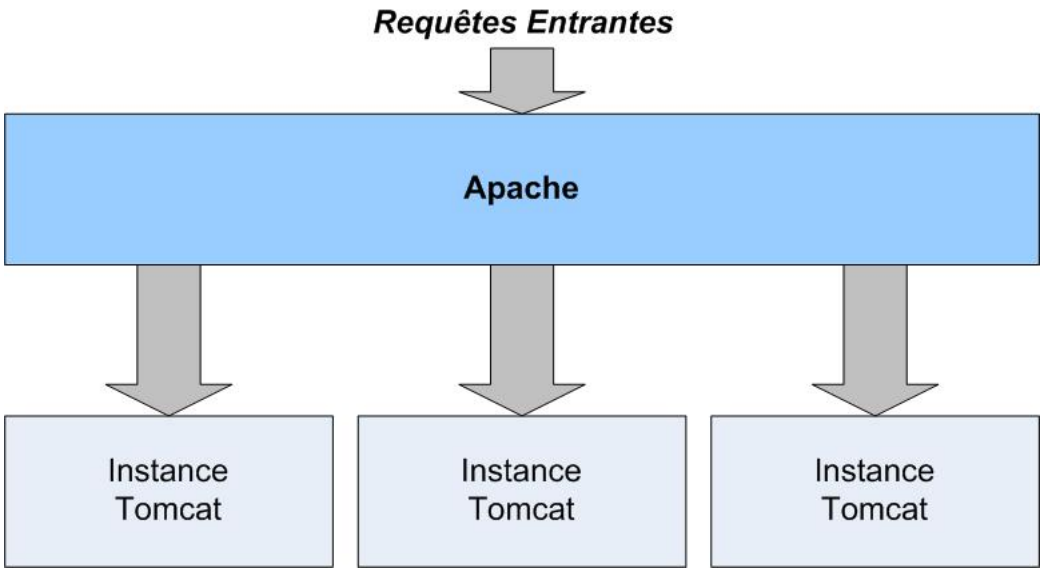


Figure 38 : Architecture avec un serveur Apache et plusieurs serveurs Tomcat

En général, en environnement de production, on utilise avec Tomcat un serveur Apache en frontal. Le serveur Web gère alors tout le contenu statique, alors que le serveur Tomcat gère tout le contenu dynamique. Il y a au moins deux raisons à ceci :

D'une part, un conteneur de servlets tel que Tomcat ne gère pas les contenus statiques tels que les pages HTML statiques, images, feuilles de style, javascript… aussi efficacement qu'un serveur Web tel qu'Apache.

D'autre part, le serveur Web Apache existe depuis beaucoup plus longtemps, il est donc plus stable que Tomcat et présente moins de failles de sécurité. Il présente également l'avantage d'être beaucoup plus configurable et de proposer plus de possibilités que le connecteur HTTP fournit par défaut avec Tomcat.

Dans le cas où un site Web présente un trafic important, c'est une bonne solution de diriger les requêtes entrantes d'un Apache sur de multiples instances Tomcat au lieu d'une seule instance. Les multiples instances de Tomcat seront alors capables de gérer une charge plus importante avant de tomber, qu'une unique instance Tomcat.

Tomcat propose différentes méthodes logicielles afin d'implémenter la répartition de charge :

- apache mod_jk

sauf pour la version 5 de Tomcat

- apache mod_jk2 à partir de la version 5

Il est possible d'affecter des poids aux diverses instances de Tomcat.

- L'application WEB Balancer

Cette application est incluse dans Tomcat 5. Elle se base sur des règles de répartition.

MOD_JK2

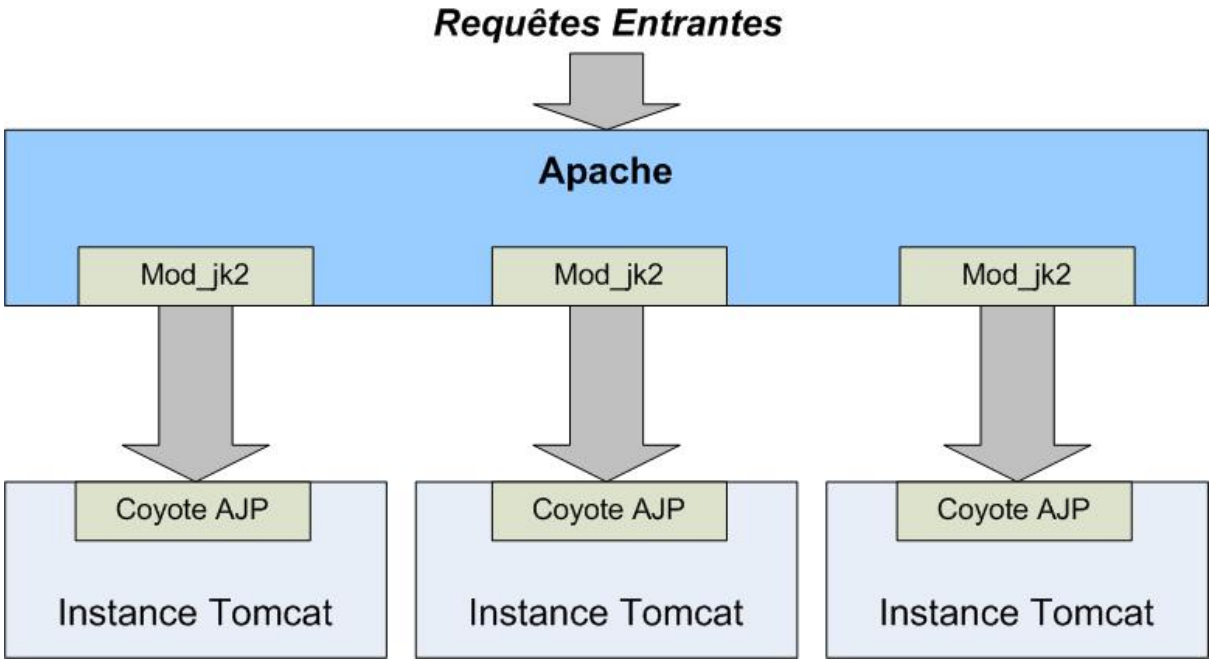


Figure 39 : Tomcat et Apache intégrés à l'aide de JK2

Tomcat est intégrable à Apache à l'aide du connecteur JK2 *(Figure 39)*. Cette intégration est rendue possible par un module Apache appelé mod_jk2 du côté Apache, par un connecteur du côté Tomcat et par un protocole AJP permettant la communication entre les deux.

Toutes les requêtes arrivent au serveur Apache qui prend en charge le contenu statique. Les requêtes, correspondant aux contenus dynamiques tels que les pages jsp et les servlets, sont passées à Tomcat qui les prend en charge. Les réponses sont ensuite envoyées par Tomcat au client via mod_jk2.

Une instance Tomcat est représentée par un worker. Il permet de gérer les requêtes des composants dynamiques. Chaque worker est identifié de façon unique par la combinaison de l'adresse IP ou du nom de la machine sur laquelle tourne l'instance Tomcat et du numéro de port sur lequel Tomcat écoute les requêtes entrantes. Afin d'implémenter du Load balancing ou du site partitionning, on peut définir plusieurs workers.

Il existe plusieurs types de workers :

- ajp13

Les workers de type ajp13 représentent une instance Tomcat. Ils peuvent posséder un identifiant nommé tomcat id, ainsi qu'un canal de communication. Le port utilisé par défaut est le 8009.

- lb

Le second type de workers est lb. Ce type est utilisé pour le Load balancing. Ce worker ne traite aucune requête. Il gère la communication entre le serveur WEB et les autres workers Tomcat définis avec le type ajp13. Ce worker supporte le Round Robin Load balancing: les différentes instances de Tomcat sont choisies de façon séquentielle. Il est possible d'affecter des poids aux différents workers. Il permet un certain niveau de tolérance aux pannes et de gérer les échecs au niveau des requêtes (request based fail over).

La configuration de ces workers est effectuée à l'aide d'un fichier appelé workers2.properties.

En utilisant le connecteur AJP, Apache distribue les requêtes entrantes correspondant aux pages JSP ou aux servlets à l'un des workers Tomcat disponibles.

Afin de dispatcher les requêtes, Apache utilise des règles fournies sous la forme de mapping URI dans le fichier de configuration workers2.properties *(Figure 40)*. Un motif ou pattern est utilisé pour faire correspondre les requêtes aux workers. Il se compose d'un nom de serveur, d'un port et de caractères spéciaux.

```
# déclaration des règles de mapping URI

# toutes les requêtes correspondant au module
# gdp sont envoyées vers un groupe d'instances tomcat particulier

[uri:smbplano2/gdp/*]
group=gdp

# il en va de même pour les modules d'administration

[uri:smbplano2/merchadminV3/*]
group=adminDiffusion

[uri:smbplano2/visuadmin/*]
group=adminDiffusion
```

Figure 40 : Exemple fichier worker2.properties

Le module Apache mod_jk2 supporte le Load balancing avec le concept de seamless session ou session affinity.

Quand un client demande une ressource dynamique pour la première fois, le Load balancer va diriger sa requête à n'importe quelle instance Tomcat disponible. Toutes les requêtes suivantes de la même session du navigateur seront dirigées sur le même serveur Tomcat.

```
# si l'URL contient l'URI ci-dessous
# elle est redirigée vers le groupe diffusion

[uri:smbplano2/diffusionV3/*]
group=diffusion

[uri:smbplano2/visu/*]
group=diffusion


# définition du worker permettant le load balancing

[lb:diffusion]
debug=0

# définition des canaux de communication
# pour chacune des instances tomcat
# port: précise le port sur lequel écoute l'instance
# host: précise le nom du serveur sur lequel est démarrée l'instance
# group: indique le groupe sur lequel va être répartie la charge
```

```
# tomcatId: identifie une instance de tomcat.
# Cet identifiant est utilisé dans la gestion de l'affinité de serveur
# cet identifiant est rappelé au niveau de la balise engine
# dans le fichier de configuration server.xml correspondant
# à l'instance tomcat à l'aide de l'attribut jvmRoute

[channel.socket:smbplano2:13009]
port=13009
host=smbplano2
group=diffusion
tomcatId=tomcat1

[channel.socket:smbplano2:14009]
port=14009
host=smbplano2
group=diffusion
tomcatId=tomcat2

[channel.socket:smbplano2:15009]
port=15009
host=smbplano2
group=diffusion
tomcatId=tomcat3

# définition des workers correspondant aux instances tomcat

[ajp13:smbplano2:13009]
channel=channel.socket:smbplano2:13009

[ajp13:smbplano2:14009]
channel=channel.socket:smbplano2:14009

[ajp13:smbplano2:15009]
channel=channel.socket:smbplano2:15009
```

Figure 41 : Exemple de fichier avec plusieurs instances

L'inconvénient de cette solution est que les sessions ne sont pas sauvegardées. Elles sont volatiles. Si une instance Tomcat présente une défaillance, toutes les sessions correspondantes sont perdues.

L'application WEB balancer

L'application WEB balancer est une alternative à la solution ci-dessus. Elle est fournie avec Tomcat 5 et permet de mettre en place une répartition de charge utilisant des règles de réécriture d'URL. Elle se déploie sur une unique instance de Tomcat, ce qui peut représenter un goulot d'étranglement. Cette application fonctionne correctement quand le nombre de membres présents dans le cluster est faible. Sinon, elle peut rapidement ne pas tenir la charge. Une solution à base du serveur Web Apache est beaucoup plus stable.

Tomcat HA

Tomcat propose également des mécanismes de haute disponibilité HA qui permettent de continuer à répondre aux requêtes utilisateurs même en cas de défaillance du système. Il est donc possible d'offrir un haut niveau de disponibilité.

Dans le cas de Tomcat, une requête destinée à un serveur indisponible est redirigée vers un autre serveur disponible du cluster. La requête entrante est traitée par le serveur qui fonctionne. Le serveur Tomcat en échec est retiré du cluster. Ainsi, aucune autre requête ne lui sera envoyée. A l'aide de certains mécanismes, il est également possible de faire en sorte que, lorsque le serveur en échec est à nouveau disponible, il soit réintégré dans le cluster et traite à nouveau des requêtes entrantes.

La clé de ce scénario est de faire en sorte que les informations correspondant à l'utilisateur dans la session soient conservées par le serveur et mis à disposition du serveur qui fonctionne.

Partage de Session (Session Sharing)

Le partage de session peut être implémenté de plusieurs manières. Chacune de ces méthodes apporte différents niveaux de fonctionnalité et de complexité d'implémentation. Il permet d'assurer un transfert transparent des sessions gérées par le serveur en échec à un serveur actif.

En utilisant mod_jk2, on peut configurer le partage de session de différentes manières avec :

- L'affinité de session sans le partage de session
- L'affinité de session et le gestionnaire de persistance de session permettant le partage de session à l'aide d'un fichier de stockage partagé
- L'affinité de session et le gestionnaire de persistance de session permettant le partage de session dans un RDBMS à l'aide de JDBC
- La réplication des sessions en mémoire

Affinité de session sans le partage de session

Cette solution peut être mise en place dans le cas où les serveurs ne seraient pas souvent indisponibles, et si la perte de session de façon occasionnelle est acceptable. Dans ce cas, on ne parle pas de haute disponibilité.

Affinité de session et sessions stockées dans un fichier

Tomcat 5 fournit un gestionnaire de persistance de session, ce qui permet d'assurer la continuité de la session à la suite d'une défaillance d'un serveur.

En étant rendues persistantes dans un fichier ou une base de données, les sessions peuvent avoir un cycle de vie plus long que le serveur.

De plus, étant donné que les sessions sont stockées, lorsque le système est complètement indisponible, cela permet de redémarrer le système tout entier en récupérant un certain nombre de sessions.

Dans le cas du stockage des sessions dans un fichier, toutes les instances Tomcat utilisent le même répertoire pour stocker les sessions. Etant stockées, les sessions seront donc éventuellement récupérables. Plus la session est longue, plus elle a des chances d'être sauvegardée. Par contre, on n'a pas l'assurance que l'ensemble des sessions pourra être traité par une instance de Tomcat disponible.

Affinité de session et sessions stockées dans une base de donnée

C'est la même chose que la solution précédente, sauf que l'on utilise un RDBMS à la place de fichiers. On a alors de meilleures performances dans le cas où on a besoin de stocker de nombreuses sessions.

Sessions répliquées en mémoire

Les informations correspondant aux sessions sont maintenues en mémoire sur l'ensemble des instances du cluster. On peut ainsi avoir un système qui apporte tous les bénéfices de la haute disponibilité sans perte de sessions.

Tous les membres du cluster communiquent entre eux afin de synchroniser les informations correspondant aux sessions. Les sessions et les changements sont répercutés de façon systématique. Plus il y a de membres dans le cluster et plus le temps de réplication des informations relatives à la session est important. Cette stratégie permet de faire en sorte qu'un membre défaillant puisse être retiré du groupe et qu'un membre de nouveau actif puisse y être directement ré-intégré.

Avantages / inconvénients

Le tableau suivant présente de façon synthétique les avantages et les inconvénients des différentes solutions de gestion des sessions que nous venons de voir *(Tableau 11)*.

Tableau 11 : Les avantages et les inconvénients des différentes solutions de gestion des sessions

	Affinité de session sans partage de session	**Affinité de session et sessions stockées dans un fichier**	**Affinité de session et sessions stockées dans un SGBDR**	**Sessions répliquées en mémoire**
Répartition de charge basée sur du Round Robin Load balancing	+ supporté	+ supporté	+ supporté	+ supporté sans la mise en place de l'affinité de session
Configuration et maintenance	+ simple	+ relativement simple	+ relativement simple	- complexe
Perte des sessions	- si une instance est indisponible, on perd les sessions correspondantes	- certaines sessions peuvent être perdues pendant que le serveur tombe	- certaines sessions peuvent être perdues pendant que le serveur tombe	+ aucune perte de session si un membre du serveur est indisponible - sauf si le cluster tout entier est indisponible, on perd alors l'ensemble des sessions
HA		+ dans la plupart des cas du fait du stockage des sessions dans des fichiers	+ dans la plupart des cas du fait du stockage des sessions dans des fichiers	+ bonne HA sauf si l'ensemble du système est indisponible
Point de contention	- Le nombre de requêtes simultanées pouvant être accepté par le serveur s'élève à 256. Pour gérer plus de clients, il est nécessaire de modifier une constante dans les sources Apache et de le recompiler. Sinon, il faut utiliser plusieurs serveurs Web Apache installés sur plusieurs machines. Sous Windows, l'installation de plusieurs serveurs Apache sur un même serveur physique suppose également la modification du code source d'Apache et sa recompilation	- le trafic correspondant à l'accès du fichier peut être important, constituant ainsi un goulot d'étranglement	+ étant basé sur un RDBMS, les performances sont améliorées dans le cas où le système doit stocker de nombreuses sessions par rapport à la solution utilisant des fichiers	- le trafic au niveau du LAN peut rapidement devenir très lourd, du fait que les modifications des sessions sont synchronisées périodiquement sur l'ensemble des membres du cluster

2.8. Les stress tests

A quoi cela sert-il ?

La mise en place d'un cluster n'assure pas que la solution mise en oeuvre offrira la qualité de service attendu. Les problèmes de performance peuvent avoir d'autres origines.

Les tests de montée en charge permettent de simuler l'utilisation réelle de l'application, afin de pouvoir :

- Anticiper les problèmes

Les tests permettent d'anticiper les problèmes afin que le site ne soit pas indisponible lors de sa mise en production.

Ils aident à identifier de façon rapide les problèmes d'installation, les causes d'erreurs et d'évaluer les éventuels points de contention du système.

- Fiabiliser et stabiliser la plateforme

Après résolution des problèmes et suppression des points de contention, la plateforme est fiabilisée et stabilisée.

- Valider le dimensionnement de la plateforme

En vérifiant la bonne tenue en charge de la plateforme avec un certain nombre d'utilisateurs virtuels, on peut valider le dimensionnement de l'infrastructure technique mise en place.

- Tester la résistance de la plateforme

Des tests de stress plus poussés peuvent être effectués pour déterminer les limites de charges supportées par la plateforme.

- Connaître le niveau de performance

En évaluant et analysant les performances des scénarios « métier », on peut connaître le niveau de performance que propose l'application. Un certain nombre d'indicateurs va permettre de quantifier et de suivre l'évolution des performances offertes.

- Offrir une bonne qualité de service.

L'objectif global est d'offrir une bonne qualité de service aux utilisateurs finaux.

Offres du marché

Les outils de test de montée en charge permettent de simuler des milliers d'utilisateurs virtuels à la place d'utilisateurs réels. Cette charge est mesurable et répétable sur le système à partir d'un point de contrôle unique. Effectuer de tels tests sur un système rend possible l'identification des goulots d'étranglement et la déduction d'axes d'amélioration afin d'obtenir de meilleures performances.

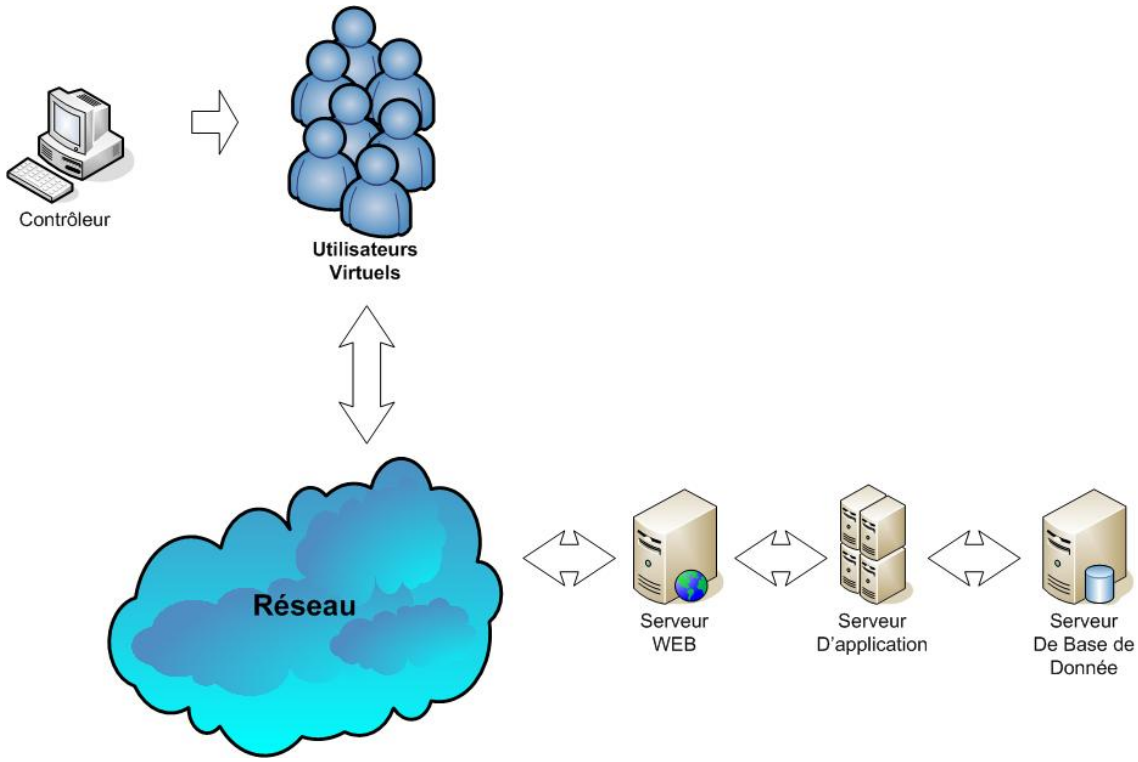


Figure 42 : Outils de test de montée en charge

Nous allons maintenant présenter brièvement différentes offres du marché. Deux solutions commerciales, LoadRunner et Webload, ainsi que la solution Open Source Opensta.

LoadRunner

C'est un outil de la société Mercury Interactive Corporation, qui est le leader du marché. LoadRunner est un produit mature qui permet de simuler une charge correspondante à plusieurs milliers d'utilisateurs, afin de mesurer les temps de réponse d'un serveur « stressé ». Cet outil est capable de collecter des indicateurs à divers endroits du système et de les stocker dans une base de données afin d'être exploités dans des analyses. LoadRunner supporte les EJB, J2EE, .NET, ORACLE, PeopleSoft, SAP et Siebel.

LoadRunner fonctionne conjointement avec d'autres produits proposés par la société Mercury : Tuning, Diagnostics & Monitoring et Global Management.

Tuning permet de surveiller des serveurs, des composants ou des dispositifs matériels et de capturer des métriques appropriées pour chacun d'entre eux.

Diagnostics & Monitoring est un produit identifiant les goulots d'étranglement dans des applications complexes telles que les applications J2EE, .NET, ERP ou CRM. Les développeurs peuvent, à partir d'une transaction, descendre jusqu'au composant, à la méthode ou à l'ordre SQL provoquant la dégradation de performance.

Enfin, Global Management permet de planifier des tests sur des ressources distribuées sur une large variété de machines à travers l'entreprise. Les scripts correspondant aux tests, les paramétrages de configuration, les données et les résultats des tests sont stockés de façon centralisée dans une base de données. Il est possible d'accéder à cette base de données à l'aide d'un simple navigateur Web.

WebLoad

La société Radview propose la suite TestView qui comprend WebFT, WebLoad et TestView Manager, une console d'administration centralisée.

WebFT permet de faire des tests fonctionnels. Il possède une interface très visuelle. Il combine la possibilité de faire des tests de façon rapide et automatique avec de nombreuses options permettant une exploitation simple des résultats. Les scripts peuvent être utilisés par WebLoad.

WebLoad est le produit de Radview réalisant des tests de charge. Il fonctionne avec la plupart des applications serveurs. Il supporte de nombreux standards tels que W3C DOM, HTML, XML, ActiveX, Java, Javascript …

WebLoad Analyzer permet de collecter les informations provenant des différents serveurs et de les examiner en recherchant des motifs révélant des problèmes et leurs causes. Toutes les informations sont automatiquement corrélées et synchronisées en temps réel. Les goulots d'étranglements peuvent être réduits à un conteneur J2EE, un composant, une classe ou une méthode.

Opensta

Opensta signifie Open Systems Testing Architecture. C'est une architecture distribuée, gratuite, permettant de tester des logiciels. Un ensemble d'outils est fourni afin de réaliser des tests de charge. Les scripts sont enregistrés à partir d'un simple navigateur. Ils peuvent être créés à partir de flux HTTP ou HTTPS. Il est possible de les éditer et de les contrôler à l'aide d'un langage de script. Ces scripts peuvent être rejoués afin de simuler de nombreux utilisateurs dans un moteur de génération de charge. On peut mesurer les performances en remontant des indicateurs provenant de plateformes Windows. Il est possible de simuler une charge réaliste correspondant à l'activité de centaines voire de milliers d'utilisateurs virtuels.

Des résultats et des statistiques sont collectés durant l'exécution des tests. Les indicateurs peuvent être des minuteurs, des données SNMP, des statistiques provenant de la fenêtre de performances de Windows … On peut observer ces indicateurs en temps réel pendant l'exécution. Après l'exécution, il est également possible de consulter, de filtrer, d'exporter les informations relatives à ces indicateurs afin de bâtir des rapports plus sophistiqués.

Comparatif

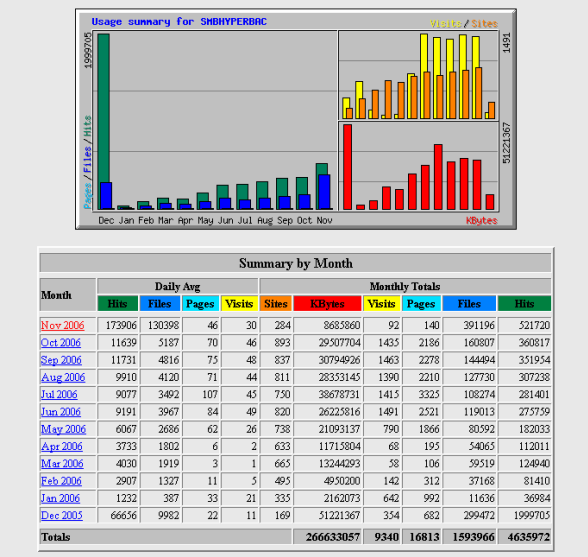
Tableau 12 : Comparatif des différentes offres

		Outils		
Critères	**Description**	**LoadRunner**	**WebLoad**	**Opensta**
Protocole	Liste des protocoles qui peuvent être gérés par l'outil	De nombreux protocoles sont supportés	De nombreux protocoles sont supportés	HTTP 1.0 / 1.1 / HTTPS seulement
Langage de script	Le langage utilisé pour représenter les données du protocole capturé et manipuler les données afin de les rejouer	TSL qui utilise une syntaxe standard du C	Syntaxe javascript	SCL qui est un langage similaire au BASIC Limité en terme de fonctions
Corrélation	Tâches permettant la substitution des valeurs dans les données dynamiques	Possibilité de corrélation automatique	Corrélation facilitée	Corrélation manuelle
Contrôleur	Application permettant de gérer et contrôler les tests	Permet la surveillance en temps réel. Possibilité de génération automatique de scénario	Permet la surveillance en temps réel. Possibilité de modifier le nombre d'utilisateurs virtuels en temps réel	Permet la surveillance en temps réel. Pas de possibilité de génération de scénario en temps réel
Surveillance	Possibilité de capture des informations relatives aux ressources utilisées. Possibilité de les voir pendant l'exécution et de les utiliser afin de construire des rapports	En temps réel via une interface Web. Possibilité graphique importante. Très nombreux systèmes supportés pour des captures	En temps réel via un client lourd. Des possibilités graphiques. Nombreux systèmes supportés pour des captures	Permet de capturer des indicateurs Windows et SNMP en temps réel, des mesures de progression du test
Tests distribués	Possibilité de distribuer la génération de la charge à l'aide de plusieurs machines	Permet de gérer plusieurs injecteurs de charge gérés par un contrôleur unique	Permet de gérer plusieurs injecteurs de charge gérés par un contrôleur unique	Permet de gérer plusieurs injecteurs de charge gérés par un contrôleur unique.
IP Spoofing	La possibilité de simuler le comportement de différentes adresses IP accédant aux systèmes. Particulièrement intéressant avec les systèmes de répartition de charge.	Supporté	Supporté	Non supporté
Emulation de la vitesse des connexions utilisateurs	Possibilité de simulation des différentes vitesses réseaux qui peuvent être utilisées par de vrais utilisateurs	Supporté	Supporté	Non supporté
Rapports et analyses	Possibilité d'examen et de recherche dans les résultats de test. Y compris les indicateurs remontés du système testé. Possibilité de mise en forme graphique.	Possibilité de rapport sophistiqué : de nombreux diagrammes et graphes disponibles. Génération automatique de rapport	Permet la mise en place de rapport évolué. Génération automatique de rapport	Permet de réaliser des diagrammes et des graphes simples suffisants pour analyser les résultats des tests de montée en charge. Possibilité d'export vers EXCEL. Des outils libres sont disponibles afin de mettre en forme ces exports
Scalability	Possibilité de générer un nombre d'utilisateurs virtuels important, ainsi que de gérer les indicateurs de suivie.	Supporté	Supporté	Supporté
Coût	Coût relatif aux licences	En 2005 pour un stress tests de 100 VU, le tarif avoisinait les 40 000 euros pour 2 semaines de test	En 2005 pour un stress tests de 100 VU, le tarif avoisinait les 20 000 euros pour 2 semaines de test	Gratuit
Machines nécessaires	Machines nécessaires à l'installation les outils de stress tests	Configuration hardware requise à priori plus lourde que les autres solutions du fait des nombreuses fonctionnalités disponibles	Configuration hardware relativement moins importante	Configuration hardware plus légère mais besoin en RAM beaucoup plus important que les deux premières solutions

Webalizer

Pour effectuer les tests de montée en charge, nous avons utilisé Opensta. Afin d'avoir une vision plus synthétique des journaux du serveur WEB Apache, nous avons également utilisé l'outil Webalizer.

Webalizer est un programme libre qui permet d'analyser les fichiers de log WEB. Il produit des rapports détaillés facilement configurables au format HTML. Ces rapports sont consultables à l'aide d'un simple navigateur Web *(Figure 43)*.



Summary by Month

Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	KBytes	Visits	Pages	Files	Hits
Nov 2006	173906	130398	46	30	284	8685860	92	140	391196	521720
Oct 2006	11639	5187	70	46	893	29507704	1435	2186	160807	360817
Sep 2006	11731	4816	75	48	837	30794926	1463	2278	144494	351954
Aug 2006	9910	4120	71	44	811	28353145	1390	2210	127730	307238
Jul 2006	9077	3492	107	45	750	38678731	1415	3325	108274	281401
Jun 2006	9191	3967	84	49	820	26225816	1491	2521	119013	275759
May 2006	6067	2686	62	26	738	21093137	790	1866	80592	182033
Apr 2006	3733	1802	6	2	633	11715804	68	195	54065	112011
Mar 2006	4030	1919	3	1	665	13244293	58	106	59519	124940
Feb 2006	2907	1327	11	5	495	4950200	142	312	37168	81410
Jan 2006	1232	387	33	21	335	2162073	642	992	11636	36984
Dec 2005	66656	9982	22	11	169	51221367	354	682	299472	1999705
Totals						266633057	9340	16813	1593966	4635972

Figure 43 : Exemple de rapport Webalizer

Il permet d'avoir de façon graphique et sous forme de tableau :

- La moyenne journalière du nombre de demandes (Hits), de fichiers, de pages et de visites.
- Le total mensuel du nombre de sites, de Kbytes échangés, de visites, de pages, de fichiers et de demandes.
- Les statistiques mensuelles.
- Le nombre de demandes classées par code réponse HTML : de 200 pour OK à 500 pour une erreur système.
- Le détail journalier des statistiques correspondant aux nombres de sites, de Kbytes échangés, de visites, de pages, de fichiers, et de demandes.
- La moyenne par heure sur le mois du nombre de Kbytes échangés, de pages, de fichiers, et de demandes.
- La liste des n URL les plus demandées avec le nombre de Kbytes échangés et le nombre de demandes correspondantes.
- La liste des URL correspondant aux volumes de données échangées les plus importants.
- La liste des URL les plus visitées.
- La liste des pages de sortie.
- La liste des différentes machines (sites) identifiées par leurs noms ou leurs adresses IP, avec le nombre de demandes, de fichiers, de KiloBytes échangés et de visites correspondant à chaque site.
- La liste des différents types de navigateur utilisé.

La démarche

Définition des tests

On spécifie avant tout l'objectif que l'application doit être capable d'atteindre. On fixe pour cela le temps de réponse désiré en fonction d'un nombre d'utilisateurs qui utilisent l'application de façon concurrente.

Lors des tests de montée en charge, on va appliquer une charge à un système à tester.

Il est donc très important de bien déterminer la charge à appliquer et de bien prendre en compte les caractéristiques du système à tester.

Simulation de la charge

Dans un premier temps, il est nécessaire d'identifier les scénarios « métier » utilisés par les utilisateurs. La description de ces scénarios permet de dérouler les transactions « métier » et de traduire ainsi le comportement des utilisateurs.

On décortique les processus suivis par les utilisateurs afin de dégager un certain nombre de scénarios qui seront simulés lors des stress tests. Il est important que ces scénarios soient le plus proche possible des comportements qu'auront réellement les utilisateurs finaux.

Pour représenter les comportements des utilisateurs, on précise les temps d'attente entre chaque action des utilisateurs dans les scénarios. Cette notion est très importante car elle est directement liée à l'évaluation de la charge de l'application. En effet, plus ces temps d'inaction sont conséquents plus la charge supportée par le système sera importante.

Il ne faut pas que les valeurs retenues minimisent la sollicitation du système par les utilisateurs virtuels. On peut estimer le temps d'attente à environ 5 secondes en consultation, et à peu près le double en modification.

A partir de la connaissance du nombre d'utilisateurs par scénario, on va déterminer un modèle de charge. Il est nécessaire de connaître le nombre total d'utilisateurs qui utiliseront l'application, leur répartition sur les différents scénarios identifiés, la façon dont ils accéderont à l'application, au moyen d'un client WEB ou à l'aide d'un client lourd. Le point d'accès utilisé est également très important, car il faut alors vérifier qu'il n'y a pas de goulot d'étranglement entre ce point d'accès et l'application. Au niveau du réseau, on vérifie si la bande passante est suffisante.

Lors de la définition du modèle de charge, le nombre d'utilisateurs à simuler pour chacun des scénarios est donc précisé. Un scénario donné peut n'être utilisé que par un seul type de profil. On répartit donc les différentes populations d'utilisateurs existants sur les scénarios identifiés.

Si l'applicatif est utilisé par 2 administrateurs et 100 utilisateurs finaux, les scénarios des administrateurs seront déroulés pour 2 utilisateurs concurrents alors que les scénarios correspondant aux utilisateurs finaux le seront pour 30 ou 50.

Il est difficile de déterminer exactement le nombre d'utilisateurs concurrents. Cependant, on peut se dire que le nombre d'utilisateurs concurrents doit être d'autant plus élevé que la période accordée aux utilisateurs finaux pour utiliser l'application est courte et que la durée des scénarios est longue.

Analyse du système à tester

Après avoir défini le modèle de charge à appliquer au système à tester, il est nécessaire d'analyser celui-ci. Cette analyse ne se limite pas uniquement à l'architecture de l'application. Il faut également vérifier, au niveau du réseau, le chemin utilisé par les utilisateurs pour solliciter les serveurs de l'application. Ceci dans le but de lever d'éventuels goulots d'étranglement.

Une bonne connaissance de l'architecture va permettre d'identifier un certain nombre d'indicateurs que l'on pourra récupérer lors de l'exécution des tests de montée en charge afin de surveiller le système. On aura ainsi la possibilité d'observer le comportement des composants importants et constitutifs de l'application.

On commence donc par faire un état des lieux de l'application.

On identifie les différents modules qui la composent, les différentes technologies utilisées, le type de répartition de charge employé, la nature du système de répartition utilisé, s'il est logiciel ou hardware, le type de serveur Web, de serveur d'application, la base de données employée…

S'il y a plusieurs serveurs, on localise les différents éléments installés sur chacun des serveurs. On liste les caractéristiques techniques des serveurs : nombre de CPU, RAM, caractéristiques des disques durs, type de contrôleur (RAID 0 à 5) …

Ensuite, on passe en revue le réseau en réalisant une cartographie de celui-ci. On identifie les différentes bandes passantes présentes entre les serveurs et les utilisateurs finaux, les différents débits, les routeurs et les mécanismes de redondance mis en place.

Le schéma ci-dessous *(Figure 44)* présente une cartographie du réseau utilisé dans notre cas.

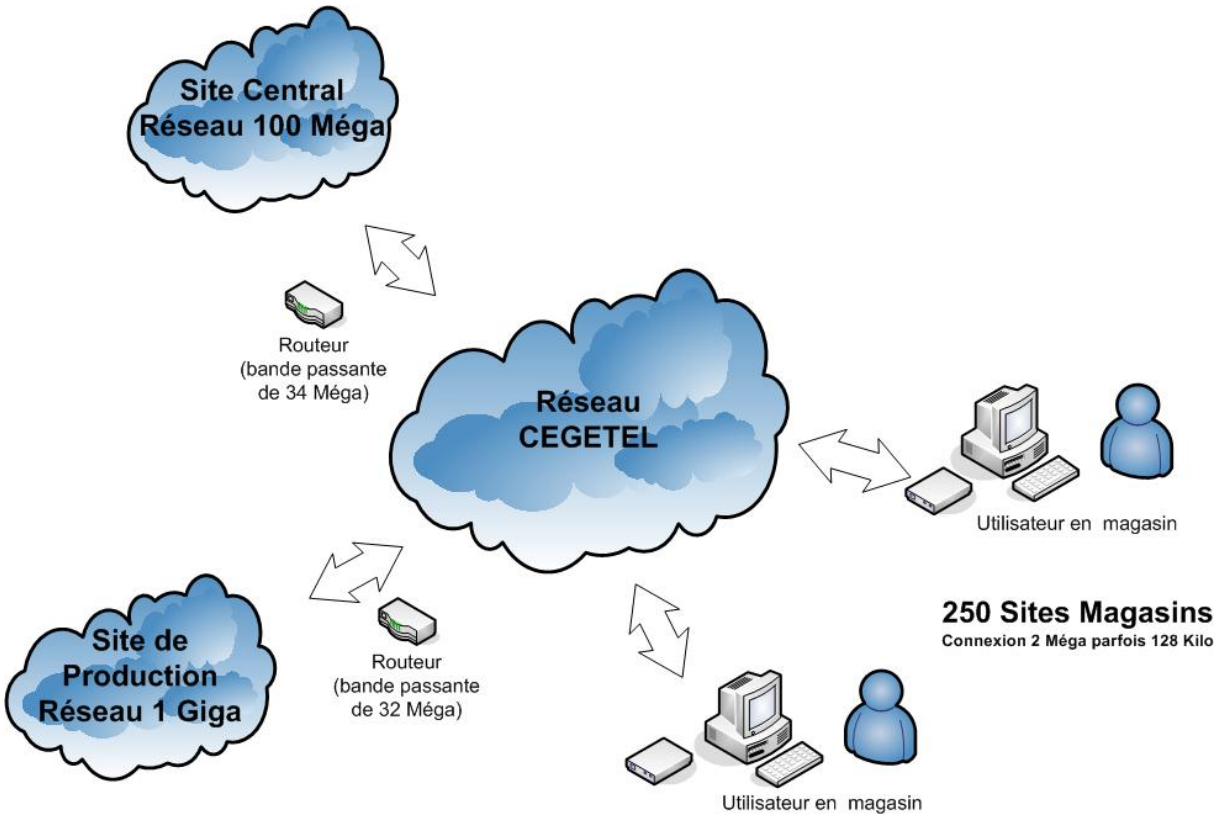


Figure 44 : Exemple du réseau

On identifie également les flux concurrents et si ces flux sont priorisés.

Ce passage en revue permet de décider quels seront les indicateurs à utiliser pour surveiller le système lors des tests de montée en charge. Une fois ce choix effectué, il ne reste plus qu'à installer les différents moniteurs permettant de remonter l'information correspondant aux indicateurs. Par exemple, on peut activer les services SMNP sur le serveur hébergeant une base de données Oracle afin de récupérer le nombre de connexions ouvertes sur la base ou tout autre indicateur.

Choix des indicateurs à récupérer

On liste donc les indicateurs que l'on veut récupérer à partir des serveurs lors des tests de montée en charge.

En général, on surveille dans un premier temps les indicateurs relatifs aux systèmes d'exploitation puis on affine les résultats à l'aide des indicateurs relatifs aux bases de données, aux serveurs Web, aux serveurs d'applications …

Les indicateurs peuvent être pour:

- Windows

Le pourcentage de CPU ou de RAM consommé par le serveur, par les différents processus qu'il exécute tels que les processus Oracle, java, … Dans le cas des processus, de nombreux indicateurs peuvent être surveillés tels que le nombre d'entrées-sorties sur le disque ou tous les indicateurs accessibles dans l'onglet « processus » de la fenêtre de gestionnaire des tâches de Windows.

- Oracle

Pour ce qui est des bases de données relationnelles, on peut surveiller le nombre de connexions, les requêtes SQL les plus coûteuses. Tous les outils de tests de montée en charge ne proposent pas forcément ces options.

- les serveurs d'application et les serveurs WEB

On peut surveiller la taille des JVM (Java Virtual Machine), le pourcentage de CPU consommé, et à l'aide de certains outils, le nombre de sessions, de threads créés ou détruits …

Il est très important de se rappeler que la récupération des indicateurs parasite les tests. La tenue de charge est moins bonne que ce qu'elle est réellement. En effet, une partie des ressources du serveur est utilisée pour remonter les informations permettant de surveiller le système.

Mise en place de l'outil de stress test

Une fois le modèle de charge établi et les différents indicateurs de surveillance choisis, il est possible de définir une plateforme de test.

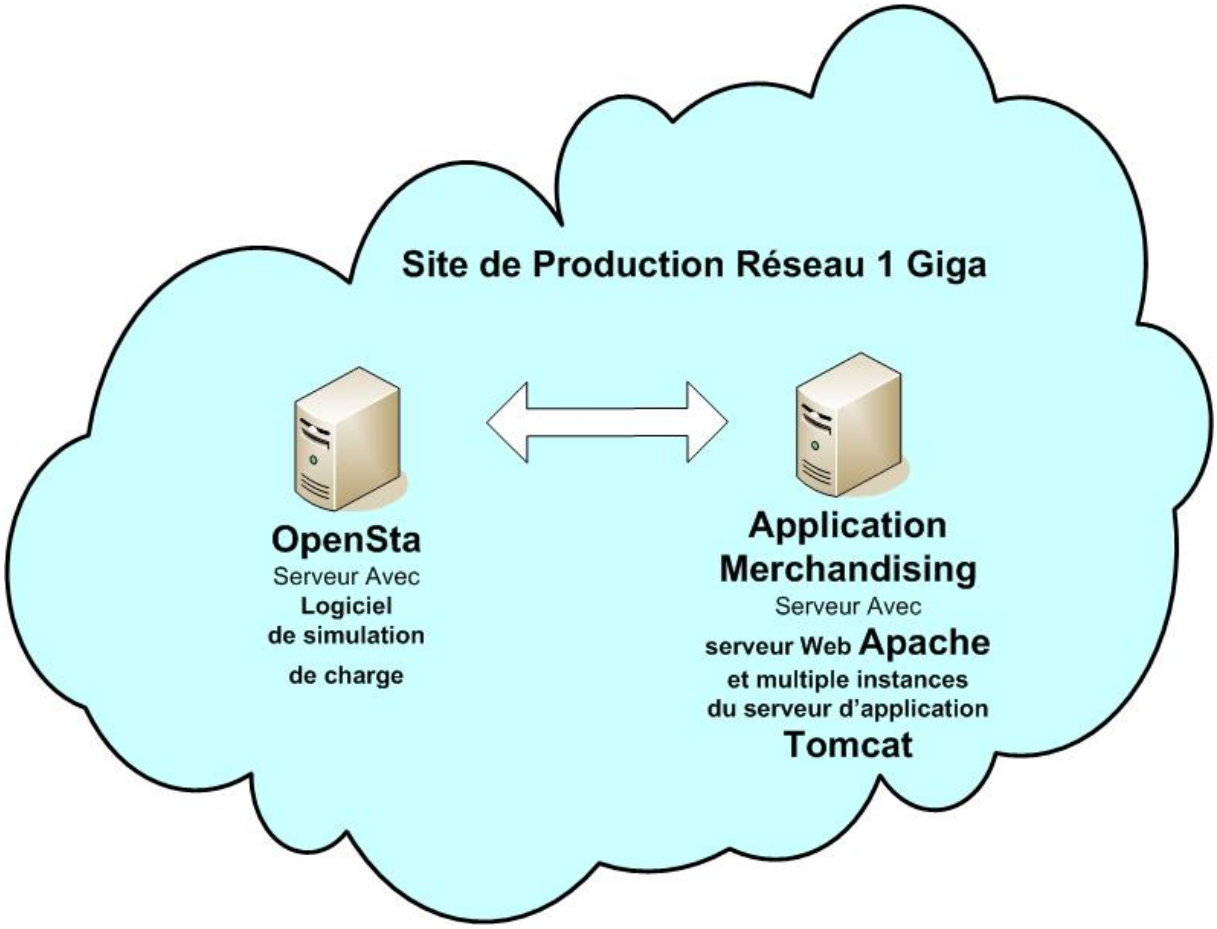


Figure 45 : Plateforme de test

On va choisir la ou les machines qui vont permettre de générer la charge. On spécifie les caractéristiques matérielles requises pour pouvoir simuler le nombre d'utilisateurs virtuels qui a été défini comme objectif dans le test. Voici un tableau résumant les caractéristiques du serveur Opensta *(Tableau 13)*.

Tableau 13: Caractéristique du serveur Opensta

Caractéristiques	**Valeurs**
Micro processeurs	2 cpu 2,4 GHz
Mémoire	3,4 Go de ram
Disque dur	30 Go de disque dur
Système d'exploitation	Windows 2003 server SP 1

On installe le logiciel qui va permettre d'effectuer le test de charge. On définit les configurations relatives à la remontée des indicateurs.

Les injecteurs de charge sont alors mis en place, ainsi que l'application de contrôle qui va permettre de piloter ces injecteurs. Dans notre cas, la mise en place d'un unique injecteur a été suffisante pour effectuer le test de montée en charge.

Pour récupérer certains indicateurs, il est nécessaire d'activer des agents sur les serveurs à tester. On peut activer la surveillance des métriques sur les applications serveurs ou configurer les services SNMP d'ORACLE.

Le logiciel de stress tests pourra alors récupérer les indicateurs choisis précédemment et les corréler avec les résultats récupérés lors des tests de montée en charge.

Ce travail de préparation de l'outil de stress test n'est pas négligeable et peut être très consommateur de temps, d'autant plus si des mécanismes de sécurité ont été mis en place dans le Système d'Information de l'entreprise. Il est alors nécessaire de déclarer des utilisateurs pour les tests de montée en charge avec des droits adéquats.

Enregistrement des scénarios

On enregistre chacun des scénarios retenus. On obtient alors des fichiers de script que l'on industrialise afin de représenter le comportement d'un ensemble d'utilisateurs.

Une fois les captures effectuées, on analyse la façon dont est gérée l'identification pour pouvoir simuler des utilisateurs virtuels avec des noms et des mots de passe différents. On définit alors des variables qui seront alimentées par des fichiers contenant la liste des utilisateurs et leurs mots de passe.

On examine également la façon dont sont gérés les différentes zones présentes dans les formulaires, les choix dans les listes déroulantes, les cases à cocher, afin de pouvoir simuler des comportements distincts pour chacun des utilisateurs lors du déroulement du test de montée en charge.

Il est alors nécessaire de fournir un fichier contenant les valeurs des choix qu'effectuent les différents utilisateurs lorsqu'ils naviguent dans les pages.

De plus, si on utilise un workflow, un utilisateur peut donner la main à son responsable. Il faudra également fournir un moyen d'identifier le supérieur de chacun des utilisateurs.

Il y a donc un travail important en terme de préparation de données afin de mettre en place les scénarios qui vont être utilisés dans les tests de montée en charge. Ces données permettront d'alimenter les variables utilisées dans les scripts et de simuler des utilisateurs virtuels distincts qui font des choix différents lorsqu'ils naviguent sur le site.

Exécution des tests

Les enregistrements des scénarios, une fois paramétrés, peuvent être utilisés dans des tests de montée en charge en indiquant un nombre d'utilisateurs virtuels.

Lors des stress tests, il est possible de récupérer des informations relatives aux scénarios mais également à l'ensemble des URL qui les composent. Si plusieurs pages de l'application ont la même URL avec des paramètres différents, les informations relatives à ces pages seront cumulées.

Pour avoir une vision « métier » des scénarios, il est possible de définir des transactions qui correspondent à des ensembles d'instructions. Cela peut être une étape d'identification, une étape de saisie et d'envoi de données, une demande d'affichage d'une page précise …

Les informations récupérées pour les scénarios, les transactions et les URL sont le temps d'exécution, la bande passante correspondante, le nombre d'erreurs associées …

Plusieurs types de test de montée en charge peuvent être exécutés :

- Test témoin

Un premier test comportant peu d'utilisateurs virtuels est effectué pour chacun des scénarios et va servir de référence pour les temps de réponse. Les temps de réponse obtenus correspondent aux temps de réponse à vide.

- Stress test

On procède à un test de montée en charge par palier. On va, par exemple, simuler la connexion de 10 utilisateurs toutes les 5 secondes pour atteindre un nombre total de 100 utilisateurs simultanés

On surveille alors l'ensemble du système afin d'identifier les points faibles, c'est à dire lorsque l'application répond mal ou provoque de nombreuses erreurs. On suit en particulier les consommations de CPU, de RAM, les entrées-sorties sur les disques, les codes erreurs dans les fichiers journaux (logs).

Ce type de test permet de mettre à jour les goulots d'étranglement que présente le système testé.

- Test de tenue en charge

Une fois les goulots d'étranglement identifiés et éliminés, on peut faire des tests de charge avec un nombre moyen d'utilisateurs sur une durée de 12 à 24 heures. Ceci afin de vérifier que l'application a une bonne tenue en charge et de s'assurer que la mémoire utilisée

pour la gestion des sessions est correctement libérée. On peut, à l'aide de cette méthode, mettre en évidence les débordements de mémoire.

Itération des tests

Pour tirer tous les avantages des tests de montée en charge, il est nécessaire de supprimer les goulots d'étranglement identifiés afin d'effectuer les tests de façon itérative et ainsi lever d'autres goulots d'étranglement.

Généralement, lors des tests, il est nécessaire de faire appel à un ensemble de consultants spécialisés dans chacun des domaines concernés ou de solliciter diverses Hotlines correspondant aux composants logiciels et matériels utilisés.

Les goulots d'étranglement pouvant être de natures diverses et variées, on peut donc être amené à solliciter de nombreux intervenants : des ressources des éditeurs de logiciels (consultants, Hotlines), les responsables systèmes et réseaux, les personnes en charge du développement …

La non-résolution d'un goulot d'étranglement peut engendrer alors un coût important du fait que certains des intervenants devront attendre avant de pouvoir effectuer leurs tâches.

L'idéal est pourtant d'avoir l'ensemble des intervenants présents sur le site pour résoudre les problèmes de façon rapide.

Bien que les stress tests représentent un investissement financier important, ils permettent de garantir un certain niveau de qualité de l'application.

Rapport

Les enregistrements effectués lors des tests vont permettre, après le dépouillement, de réaliser des rapports.

Dans ces rapports, on présente :

- Les scénarios déroulés ainsi que leurs contextes
- Le comportement du système et ses limites, par rapport aux sollicitations des utilisateurs virtuels simulés.
- Les goulots d'étranglement ; ils correspondent aux points faibles de la solution testée.
- Les axes d'améliorations.

Les axes d'améliorations peuvent être de simples optimisations ou la remise en cause de l'architecture. Ce peut être, par exemple, l'installation d'un composant spécifique de l'application sur un serveur dédié afin de pouvoir supporter davantage d'utilisateurs. L'architecture est alors modifiée.

Pour élaborer les rapports, on utilise un certain nombre de graphes et de tableaux afin que l'exploitation des résultats soit plus simple.

Deux points de vue sont traités : le point de vue « métier » et le point de vue technique.

Le point de vue métier est couvert au travers de l'analyse du comportement des scénarios, des transactions et des URL. On peut ainsi avoir une vision à plusieurs niveaux des processus métier suivis par les utilisateurs.

Un scénario se compose d'un ensemble de transactions. Une transaction comporte plusieurs URL. Elle correspond à une étape du scénario telle que l'identification, une demande particulière d'information …

Les différents graphes possibles sont :

Pour les scénarios, les transactions et les URL :

- Les temps de réponse en fonction du nombre d'utilisateurs

Il est également possible d'extrapoler le nombre de transactions :

- tableau extrapolant le nombre de transactions supportées par heure, par jour ou par mois

Le point de vue technique est couvert au travers de l'étude de la performance des machines, des différents processus installés sur les machines. Dans certain cas, si le produit le permet, on peut aller jusqu'à identifier la classe Java ou même la méthode de la classe Java qui pose des problèmes.

Les différents graphes possibles sont :

Pour les performances machines :

- le pourcentage de CPU et la RAM consommée en fonction du temps
- le pourcentage de CPU et la RAM consommée en fonction du nombre d'utilisateurs

Pour les différents processus présents sur les machines :

- le pourcentage de CPU, de RAM consommée et le nombre d'utilisateurs en fonction du temps

Pour les performances réseaux :

- La bande passante exprimée en Mbits par seconde en fonction du nombre d'utilisateurs

Pour l'application

- Les erreurs en fonction du temps

On peut indiquer le nombre d'erreur 500 en fonction du temps afin d'avoir une idée de la disponibilité globale de l'application.

Résolution des problèmes de performance

Les problèmes de performance peuvent être résolus à plusieurs niveaux :

- Technique

Au niveau du réseau, on peut être amené à installer des boîtiers de compression des flux échangés pour diminuer la bande passante consommée. Une autre solution peut consister à mettre en place sur certaines parties du réseau, des technologies proposant des débits plus importants.

Au niveau de l'architecture, on peut chercher à optimiser les différents composants. Dans le cas où cela ne suffirait pas, on peut revoir complètement l'architecture.

Au niveau de l'application, on peut optimiser la configuration de certains composants tels que la base de données Oracle, le système d'exploitation Windows ou les anti-virus utilisés …

Par exemple, les anti-virus font en général des mises à jour de façon automatique ou balaient des répertoires utilisés par l'application de façon périodique. La conséquence est une dégradation des temps d'accès à ces répertoires. Il est alors nécessaire d'exclure ces répertoires du périmètre de l'anti-virus afin qu'il n'y ait pas de dégradation des performances de l'application.

Pour ce qui est de l'optimisation des développements, on peut distinguer deux cas de figure, selon que l'application stressée est un développement spécifique ou un progiciel du marché.

Dans le cas d'une application spécifique, il peut être intéressant d'effectuer une analyse des composants techniques les plus consommateurs de ressources ou qui correspondent aux temps de réponse les plus longs. On peut alors revoir le code correspondant afin de l'optimiser.

Dans le cas d'un progiciel, cette démarche ne présente que peu d'intérêt car les équipes de développement n'ont pas la main sur le code du progiciel.

Par contre, l'identification des tâches « métier » très consommatrices peut permettre dans une moindre mesure d'optimiser le paramétrage correspondant du progiciel.

- Fonctionnel

Lorsque les optimisations techniques ne suffisent pas pour résoudre les problèmes de performance et que l'on ne désire pas investir dans la solution matérielle, il reste encore la possibilité d'apporter une solution fonctionnelle aux problèmes en recherchant un moyen d'arriver au même résultat d'une façon différente. Par exemple, si un rapport prend beaucoup trop de temps à s'exécuter, on peut chercher à le décomposer en plusieurs autres rapports ou le remanier afin qu'il prenne moins de temps à s'exécuter.

- Organisationnel

Si on doit mettre en place des rapports en temps réel, et que l'on se rend compte que la solution ne permet pas de répondre aux utilisateurs de façon fiable et provoque la chute du système de façon systématique, que de plus, aucune solution fonctionnelle n'ait été trouvée afin de palier ce problème, on peut alors envisager une organisation différente. La solution peut consister à mettre en place des rapports journaliers qui se génèrent pendant la nuit.

4. Bilan

4.1. Bilan ETL

Utilisation de GENIO et de VBIS

Dans le fond l'utilisation que nous faisons de ces deux progiciels est très similaire.

- Manipulation de l'information

On extrait de l'information de diverses applications, on la transforme et on la charge dans un système cible.

Les principales opérations appliquées à ces informations sont
- le filtrage à un périmètre précis
- l'agrégation à un niveau exploitable par le système cible
- la mise en correspondance des informations présentes dans les différents systèmes.

On peut être amené à faire correspondre des notions identiques caractérisées par des identifiants différents dans des systèmes distincts.

- Suivi de la bonne exécution des traitements

Les traitements sont suivis à travers la console de GENIO (le scheduler). Si des traitements spécifiques sont utilisés, afin de suivre leur bonne exécution, il est nécessaire de gérer des tables de statuts au niveau d'une base relationnelle. Aucune trace de l'exécution d'un agent distant n'est fournie par défaut par l'ETL.

En cas de problème, des emails sont envoyés aux administrateurs pour les avertir de la mauvaise exécution d'un programme. Dans ces emails, on peut trouver en pièces jointes les journaux correspondant à l'exécution du traitement s'ils ne sont pas trop volumineux.

Les traitements fonctionnent en général en tout ou rien. Après correction de l'erreur, il suffit de relancer le traitement.

Constat positif mais pas parfait

Ces deux outils permettent de mettre en œuvre des traitements répondant à nos attentes. Même s'il ne paraît pas y avoir de réel gain au niveau des développements, il y a un gain réel au niveau de la gestion de la complexité du Système d'Information de l'entreprise. La maintenance et la mise en place des évolutions liées à ces développements sont plus simples à mettre en œuvre. Les intervenants n'ont pas besoin d'être des profils très techniques connaissant des systèmes spécifiques.

Néanmoins, ce gain est limité par les performances de ces produits, dans le cas où les volumes seraient importants ou les règles de transformation seraient complexes, ces progiciels ne tiennent pas la charge. Il est alors nécessaire de faire un pré traitement de l'information à l'aide de langages tels que PACBASE et de bases de données relationnelles telles que DB2 pour transformer l'information à un niveau d'agrégation exploitable par le logiciel. C'est en particulier le cas du traitement des ventes qui nécessite le traitement de plusieurs dizaines de millions de lignes.

Complémentarité de VBIS et GENIO

GENIO et VBIS peuvent être utilisés de façon complémentaire.

On peut choisir d'utiliser VBIS pour :

- la mise en place d'agents sur des serveurs autres que le serveur de l'ETL. Genio ne permet pas la mise en place d'agents sous la forme d'exécutables dans le cas de Windows.
- recourir à un adaptateur qu'il est seul à posséder.
- ne pas recourir à un langage de programmation traditionnel tel que Java. L'utilisation d'un tel langage présente l'avantage de pouvoir implémenter presque tout ce que l'on souhaite mais au prix de lourds investissements en temps. Il est en effet nécessaire de réfléchir à la structuration des développements. Alors que, par défaut, les solutions EAI et ETL proposent un cadre structurant des développements.
- mettre en place des traitements utilisant de nombreux exécutables, fichiers de données, journaux, commandes système sur un même serveur et devant utiliser diverses sources de données et envoyer des emails.

De plus, il est nécessaire de recourir à d'autres outils pour pouvoir lancer l'exécution. En effet, GENIO n'est pas capable de lancer de façon directe un exécutable sur un serveur distant. L'hôte distant doit avoir un service permettant de faire du rexec sous Windows. On a aussi la possibilité d'utiliser des outils libres tels que Pstools qui permettent de lancer un exécutable sur un serveur sans avoir à installer quoi que ce soit sur ce dernier à l'aide de la commande Psexec.

Les avantages

Communs aux deux progiciels

- Interface simple

  Elle permet de mettre en place des traitements plus rapidement qu'un langage traditionnel de programmation.

- Représentation unique :

  Le développeur manipule des objets uniques indépendants de l'implémentation physique. Dans GENIO, les tables peuvent aussi bien être des fichiers que des tables relationnelles Oracle, DB2 …

  Par exemple, c'est le progiciel qui prend en charge la correspondance d'une fonction par rapport à ces implémentations dans les différents SGBDR.

Quelle que soit l'implémentation, les traitements se présentent toujours de la même façon. Ce qui permet de gérer des systèmes hétérogènes de manière homogène.

- Gain au niveau de la maintenance et l'évolution

  Les traitements ayant toujours la même structure quelle que soit l'implémentation, on n'est plus obligé de recourir de façon systématique à des compétences particulières maîtrisant les différents systèmes spécifiques.

- Analyse d'impact

  Si un objet change, une analyse d'impact sur les objets qui l'utilisent est effectuée. Ce qui permet de facilement corriger les traitements en erreur.

- Montée en compétence plus rapide par rapport à un langage de programmation traditionnel

Propres à GENIO

- Ordonnanceur

  Possibilité de déclencher les traitements de nombreuses manières

- Référentiel centralisé

  Tous les objets manipulés à l'aide de GENIO sont centralisés dans un repository et sont réutilisables.

  Les rapports d'exécution sont centralisés à un endroit unique. Il est possible de les consulter au travers d'une console d'administration.

Propres à VBIS

- Richesse de la librairie d'adaptateurs

- Facilité de déploiement

  Il est possible de déployer un même projet sur plusieurs systèmes d'exploitation.

Les inconvénients

Communs aux deux progiciels

- Le prix des licences

  Ces prix avoisinent des dizaines de milliers d'euros voire d'avantage suivant les adaptateurs désirés.

- La représentation visuelle

  Dans certains cas, la représentation visuelle des éléments utilisés pour construire les traitements est difficile à gérer. Si les traitements sont complexes, on peut avoir du mal à s'y retrouver. On est obligé de consacrer du temps à la gestion de cette complexité visuelle. On ne gagne alors pas plus de temps qu'en utilisant un langage de programmation traditionnel.

- La version des adaptateurs et des connecteurs

  Pour utiliser un adaptateur ou un connecteur propre à la version d'une application, on peut être amené à devoir faire une migration de l'EAI ou de l'ETL. Le dit adaptateur ou connecteur n'étant disponible que dans la version supérieure. Ce qui induit des coûts de migration non négligeables.

- Le besoin d'outils complémentaires
  pour lancer des traitements sur des serveurs distants

Propres à GENIO

- coût et durée de migration importants

- pas de possibilité de créer des agents sur des serveurs distants

Propres à VBIS

- Tout doit être construit à partir des adaptateurs.
  Il n'existe pas de gestion centralisée des erreurs, ni de moyen de planifier les tâches

- pas de référentiel centralisé

Les limites

- liées aux fonctionnalités très spécifiques

Ils ne permettent pas d'implémenter des fonctionnalités très spécifiques. On est obligé de rester dans le cadre de ce que proposent les adaptateurs.

L'adaptateur XML de VBIS permet de gérer des documents XML simples avec un niveau hiérarchique mais n'est pas capable de gérer des documents XML comportant de nombreux niveaux hiérarchiques.

Les DataSet de GENIO ne permettent pas de faire des jointures entre des tables appartenant à différentes bases de données.

- liées à la volumétrie :

L'ETL GENIO permet d'extraire de l'information de diverses bases de données ORACLE, DB2 au travers d'une source de donnée ODBC. Cette méthode est envisageable lorsque le nombre d'enregistrements n'est pas trop élevé.

Dans le cas où le nombre d'enregistrements est trop important, il vaut mieux utiliser la méthode suivante :

  - Ecrire dans un fichier les données à extraire
  - Les transférer à l'aide de FTP sur le serveur GENIO
  - Déclencher le processus à l'aide d'un événement
  - Charger des fichiers dans la structure cible (utiliser le cas échéant des outils de chargement spécifiques de la base de données si le nombre d'enregistrements est vraiment très important).

Cette méthode permet de réduire les temps de traitement et minimise les risques d'erreur : des temps de connexion longs favorisent les risques d'erreur liés au réseau. De plus, elle permet d'éviter de faire tomber le moteur de l'ETL en voulant gérer des volumes trop importants par le connecteur ODBC.

Si le nombre d'enregistrements contenus dans le système source est important, de l'ordre de plusieurs millions de lignes, il est préférable d'agréger préalablement l'information avant de l'envoyer dans GENIO. Il faut alimenter l'information au niveau auquel on va l'utiliser dans le projet. Ceci est d'autant plus vrai si les manipulations à effectuer sont complexes.

Nous sommes donc parfois amenés à décomposer les traitements de transformation en opérations élémentaires. Nous préférons donc, dans certains cas, faire au préalable des consolidations au niveau du Mainframe par exemple, avant de l'envoyer dans GENIO.

En général, il est nécessaire de bien gérer les erreurs dès qu'elles apparaissent. Le fait d'avoir trop d'erreurs peut entraîner la chute du moteur d'exécution de GENIO. Ceci peut arriver lorsque l'on demande l'affichage du compte rendu d'exécution du processus en erreur comprenant de nombreux messages d'erreurs.

4.2. Bilan J2EE

La présentation du développement d'une action utilisateur (Annexe C) a permis de voir concrètement la façon dont a été structurée l'application. On peut ainsi de suite s'apercevoir que, malgré le nombre réduit de couches, la mise en place d'une action utilisateur correspond à un volume de code important.

Nous avons choisi d'avoir peu de couches, au total 3 couches, afin de réduire les temps de développement. En effet, plus on ajoute de couches plus la mise en place d'une fonctionnalité est complexe. Il faut travailler sur chacune des couches et effectuer les tests unitaires correspondants.

D'un autre côté, le fait de gérer une couche par préoccupation rend l'application plus flexible par rapport aux différentes évolutions possibles : le changement de la base de données, des clients utilisés, l'implémentation de la couche « métier ». Seule une couche est impactée.

Cependant, la gestion de nombreuses couches a un inconvénient évident qui est la dégradation des performances, surtout si on utilise des frameworks ou des technologies différentes pour les implémenter.

Par contre, la mise en place des couches à l'aide d'autres outils permet de disposer de nouvelles fonctionnalités et de rendre les développements plus structurés et plus maintenables.

Le nombre de couches est donc un compromis à faire entre les temps de mise en œuvre et de réponse de l'application par rapport aux fonctionnalités que l'on peut proposer et le niveau de structuration de l'application.

On peut légitimement se poser les questions suivantes : quelles couches pourrait-on mettre en place ? De quelle façon ? Et pourquoi ?

L'architecture à 5 couches : cliente (HTML, WML, PDA…), application (aspects fonctionnels d'une application), entreprise (objets transversaux à toutes les applications), mapping (lié à la structure des données) et physique (SGBD, CICS, LDAP…). Ce modèle permet de décomposer le système en sous systèmes.

La modélisation des sous systèmes peut être effectuée de façon indépendante par des personnes différentes. Les couches peuvent être testées individuellement. L'objectif de ce modèle est l'industrialisation de la production de logiciels. Ce modèle suppose tout de même des ressources de développement importantes.

En distinguant les préoccupations fonctionnelles et transactionnelles, un haut niveau d'évolutivité est garanti. Du fait de la mise en place de couches par préoccupation, les développements sont plus maintenables. Enfin, le couplage étant faible entre chacune des couches, la réutilisation des développements se trouve améliorée.

Couche Cliente

Aujourd'hui, le terme WEB 2.0 est très à la mode. Ce terme désigne la mise en place d'interfaces utilisateurs plus évoluées que les traditionnelles interfaces clients légers. Pour cela un ensemble de nouvelles technologies telles que AJAX (XMLHttpRequest, javascript, dom, css) est mis en place afin de modifier le contenu des pages de façon dynamique sans effectuer des nouvelles requêtes au serveur [CRA06]. Le cycle classique requête/réponse utilisé par Struts est alors évité. Du code Javascript, présent dans les pages Jsp, sollicite directement des classes Java de l'application pour modifier en temps réel une portion de la page.

Dans l'application GDP, une fonctionnalité de recherche avancée a été implémentée. L'utilisateur tape une lettre par exemple « c », on lui propose la liste des processus contenant la lettre « c ». S'il complète le « c » par les lettres « ho », on lui propose la liste des processus contenant les 3 lettres « cho » comme on peut le voir sur la copie d'écran (*Figure 157*). C'est ce que l'on appelle une recherche avec complexion.

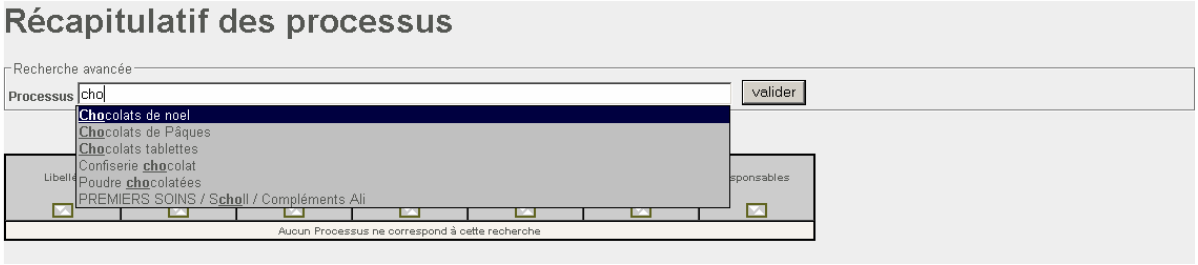


Figure 157 : Fonctionnalité de recherche avancée, implémentée dans GDP

Ensuite, on clique sur un élément de la liste et on obtient le résultat ci-dessous (*Figure 158*).

Figure 158 : Résultat de la recherche

L'utilisation d'AJAX permet donc de mettre en place des fonctionnalités que l'on n'a pas l'habitude de rencontrer dans les interfaces Web traditionnelles. Ceci permet d'améliorer de manière significative les interfaces utilisateurs WEB.

Cependant, l'utilisation directe d'AJAX nécessite un volume important de code Javascript. Il est donc préférable d'utiliser DOJO qui permet d'introduire des concepts WEB 2.0 de façon simple. DOJO est un outil Open Source qui permet de construire des interfaces graphiques DHTML.

Couche application

La mise en place des solutions à base de servlets telles que Struts présente l'inconvénient d'être liée au langage Java. Dans le cas des informations référentielles provenant des systèmes centraux, il est avantageux de pouvoir y accéder à la fois à partir des applications Web basées sur la plateforme J2EE, mais aussi à partir des progiciels ETL et EAI tels que GENIO et Vignette. Les Web Services peuvent permettre de répondre à ce problème.

Ils permettent de développer des applications distribuées accessibles depuis n'importe quel type de client tel que des clients WAP, Web, ou des applicatifs spécifiques et ceci de manière transparente pour l'utilisateur. Ils sont multi-plateformes, multi-langages, disponibles sur Internet ou Intranet avec une information actualisée en temps réel.

Les Web Services correspondent à des applications modulaires et offrent la possibilité d'exécuter des tâches précises en respectant un format spécifique. Les applications qui les utilisent peuvent faire appel à des fonctionnalités à distance. Ils présentent l'avantage de permettre de remplacer des protocoles standards tels que DCOM, RMI, CORBA, DIIOP et RPC.

Les Web Services reposent sur plusieurs standards :

- SOAP (Simple Object Access Protocol)

SOAP est un protocole simple et léger qui permet l'échange d'informations. La structure des messages échangés entre les applications pour dialoguer entre elles est basée sur le langage XML.

- WSDL (Web Service Description Language)

WSDL est une grammaire dérivée de XML qui permet de décrire les Webs Services. Il permet en particulier de décrire les méthodes et les paramètres des composants invocables à distance.

- UDDI (Universal Description, Discovery and Integration)

UDDI est une spécification définissant la manière de publier et de découvrir les Web Services sur un réseau. Tout nouveau service est décrit en utilisant un langage dérivé de XML selon les spécifications de UDDI.

Les Web Services peuvent donc être utilisés afin d'accéder aux référentiels centraux tels que le référentiel produit, à partir de diverses applications. Ce qui évite de développer les fonctionnalités correspondantes dans chacune des applications WEB et chacun des progiciels utilisés dans le projet. La gestion des fonctionnalités proposées est donc centralisée à un endroit unique. Néanmoins, comme ils reposent sur un protocole basé sur XML, ils sont plutôt destinés à gérer des volumétries de données pas trop importantes, ce qui est le cas en général des référentiels.

Framework Web

Le framework Struts que nous avons utilisé se situe principalement dans les couches application et cliente. Il ne permet pas de distinguer franchement ces couches. C'est pour cette raison que nous avons parlé de couche de présentation qui désigne le regroupement de ces deux couches.

L'un des principaux inconvénients de Struts est de ne pas permettre de mettre des composants graphiques réutilisables dans une application ou dans un ensemble d'applications. On est obligé de redonder le code à chaque utilisation.

Un second inconvénient tient au fait que l'on ne peut utiliser qu'un type de vue unique au niveau du contrôleur. L'implémentation des autres vues se fait de façon spécifique.

Il existe d'autres solutions telles que JSF, Tapestry, Spring MVC qui permettent de mieux gérer ces aspects.

Les deux premières solutions sont basées sur une approche « component-based » contrairement à Struts qui est basé sur une approche « request-based ». Cette approche différente rend ces frameworks plus adaptés à la construction d'interfaces utilisateur complexes.

La troisième solution Spring MVC est une sous partie du projet Spring Framework destinée à réaliser des applications Web. Elle permet, entre autres, d'utiliser de nombreux types de vue au niveau du contrôleur, de façon native. Elle fait partie de la solution Spring qui est un framework transversal.

Spring

Spring est un conteneur léger [DUB06]. Il peut remplacer un conteneur lourd utilisé dans les solutions basées sur les EJB. Ce conteneur léger est mis en place au sein d'un simple conteneur WEB et ne nécessite pas l'utilisation d'un serveur d'applications. Contrairement aux EJB, les conteneurs légers sont indépendants de la technologie J2EE et peuvent fonctionner sur tout type de serveur d'applications.

Les applications mises en œuvre sont plus flexibles et plus facilement testables. Ceci est dû au fait que le couplage entre les composants est géré par le conteneur et non plus dans le code et que l'utilisation intensive d'interfaces permet de rendre les applications indépendantes de leurs implémentations.

De plus, Spring supporte la POA (Programmation Orientée Aspect) ou AOP (Aspect Oriented Programming). La POA permet au conteneur léger d'offrir des services similaires à ceux des conteneurs EJB avec la lourdeur en moins, car ils peuvent manipuler de simples JavaBean ou POJO (Plain Old Java Objet).

La POA est un paradigme qui structure la manière de modéliser les applications, et en conséquence, la façon de les développer. Le but est d'améliorer la modularité des logiciels afin de faciliter la réutilisation et la maintenance.

Elle offre des mécanismes complémentaires à l'approche procédurale et l'approche objet. Elle permet de gérer les éléments transversaux d'une application. Elle propose des mécanismes génériques tels que la gestion des transactions de façon transverse. Au lieu de gérer les transactions un peu partout dans le code, cet aspect est géré dans un module transverse et de façon centralisée.

Et il permet également l'intégration de frameworks spécialisés. En effet, Spring se compose d'un ensemble de modules dont :

- Spring JDBC

  Ce module propose une couche d'abstraction à JDBC permettant de réduire le nombre de lignes de code.

- Spring ORM

  Ce module permet l'intégration des frameworks ORM tels que Toplink, iBatis, Hibernate.

- Spring WEB

  Ce module permet d'intégrer les frameworks WEB tels que Struts, JSF, Tapestry … mais Spring propose également son module propre Spring MVC.

- Spring Remoting

Ce module permet l'intégration des Web Services et les appels RMI.

Spring propose divers autres modules tels que Spring JMX, Spring JMS, Spring JCA, Spring EJB …

Spring nécessite un temps d'apprentissage non négligeable comme tous les frameworks. Mais, son utilisation est plus simple que celle des standards J2EE en direct.

Il permet une meilleure structuration d'une application Web mais il oblige à gérer une surcouche supplémentaire et son utilisation nécessite que l'équipe de développement soit bien dimensionnée.

Couche Métier

Dans l'application que nous avons mise en place, il n'y a pas de couche « métier » proprement dite. Les objets « métier » correspondent aux JavaBeans qui sont échangés entre la couche de présentation et la couche d'accès aux données.

Pour implémenter une couche « métier », plusieurs solutions existent. Ces solutions sont les conteneurs lourds tels que les solutions à base d'EJB ou les conteneurs légers en utilisant un framework tel que Spring. La première solution est réputée pour être très lourde à mettre en place et convient aux projets de taille conséquente. La seconde solution correspond mieux aux petits projets et apporte des fonctionnalités similaires aux solutions à base d'EJB.

S'il fallait mettre en place une couche « métier », c'est vers cette dernière solution que nous nous orienterions, en sachant qu'elle ajoute néanmoins une complexité supplémentaire et des temps de développements plus importants. C'est donc un choix d'orientation qui doit faire l'objet d'une étude préalable approfondie.

Les EJB gèrent l'accès et le traitement des données de façon persistante ou non. Pour faire de même avec des frameworks, il faut en utiliser deux :

- Hibernate

Ce premier framework va permettre de « mapper » une base de données relationnelle aux objets POJO. Il permet ainsi de s'abstraire de l'accès à la base de données et propose un accès orienté objet aux données.

- Spring

Ce second framework prend en charge la création et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration décrivant l'ensemble de ces relations. Les contraintes d'intégration sont légères contrairement aux EJB qui impose par exemple de respecter une interface donnée.

Hibernate permet d'accéder aux données. Spring, quant à lui, sert de fabrique automatisée. De plus, ces deux frameworks s'intègrent facilement et ne nécessitent qu'un moteur de servlet et non pas un serveur d'application comme les EJB.

Hibernate permet de gagner du temps surtout si on l'utilise pour générer automatiquement la base de données et le code. Il permet de manipuler plus facilement les objets « métier ». Il permet de réduire les dépendances envers une base de données précise. Si on change de base de données, seul le fichier de configuration est à revoir et éventuellement quelques points spécifiques aux niveaux du mapping sont à revoir.

Couche Physique

Nous utilisons la base de données Oracle au niveau de la couche physique. C'est une base de données robuste et stable. Elle correspond à un choix d'entreprise. Son utilisation n'est pas remise en cause et répond tout à fait aux besoins. Cependant, le remplacement d'ORACLE par une base de données telle que MySQL peut réduire de façon significative les coûts de licence à payer pour les bases de données.

4.3. Bilan Architecture Web J2EE

Depuis la création de J2EE, les serveurs d'application se sont beaucoup développés. On peut distinguer deux grandes catégories de serveurs : les serveurs propriétaires tels que Websphere Application Serveur WAS d'IBM ou les serveurs Open Source tel que Tomcat. Les premiers évoluent selon l'éditeur. Les seconds évoluent grâce à la communauté par le moyen du WEB.

Utilisation de Tomcat

Tomcat est un conteneur de servlets qui implémente la référence officielle pour les servlets Java et les JSP, ainsi que JDBC et JNDI.

Les évolutions de la dernière version de Tomcat offrent la possibilité de concevoir des architectures permettant la montée en échelle et proposant des mécanismes de haute disponibilité. Grâce à cette solution, il est ainsi possible de mettre en place des clusters à moindre coût qui ne nécessitent pas des machines très puissantes.

Tomcat 5 apporte un ensemble d'outils d'administration et de mécanismes permettant d'étudier et d'implémenter des fonctionnalités de cluster de serveur d'applications. On peut ainsi résoudre la plupart des problématiques HA que l'on rencontre aujourd'hui dans les environnements de production.

Par rapport à une solution propriétaire

Cependant, l'administration et le déploiement restent plus lourds que les solutions propriétaires telles que WAS qui propose des outils facilitant le développement et la configuration des applications au sein d'un cluster.

Le fait de faire appel à une entreprise comme IBM permet d'accéder à une Hotline, ainsi qu'à une documentation très riche. Il est possible de recourir aux spécialistes qui travaillent de façon collaborative au niveau mondial.

La contrepartie est le paiement de licences d'utilisation de leur serveur d'application. Le coût des licences et le tarif des prestations sont élevés. L'installation d'un serveur d'application Websphere coûte près de 10 000 voire 20 000 euros par CPU. L'intervention d'un spécialiste peut, quant à elle, dépasser les 1 000 euros HT par jour.

De plus, le temps de prise en compte des modifications de code est supérieur à celui de l'Open Source.

La Solution WAS présente également certaines instabilités dues aux manques de contrôle dans les interfaces d'administration. Dans le cas de la gestion des ports correspondant aux diverses instances des serveurs d'applications, la déclaration d'un nouveau cluster peut rentrer en conflit avec une déclaration précédente. Ce qui a pour effet de provoquer l'indisponibilité du premier cluster.

Pour déclarer des ressources statiques au niveau du serveur WEB IBM HTTP server (IHS), il est nécessaire de le faire préalablement dans WAS avant de le mettre à jour au niveau du serveur WEB. Les logiques de fonctionnement du serveur ne sont pas toujours évidentes à comprendre et vont parfois à l'encontre de ce que l'on a l'habitude de faire avec une solution Open Source telle que Tomcat.

Ces solutions sont plus complexes à mettre en œuvre mais supportent mieux la charge dans le cas de déploiement d'applications très lourdes. Pour près de 100 utilisateurs concurrents, utilisant une application progiciel excessivement lourde, avec quatre instances de WAS, les temps de réponse restaient quasiment constants quelle que soit la charge, alors qu'une instance Tomcat avait du mal à servir cinq utilisateurs.

Pour les progiciels dont les applications sont très lourdes, il est donc préférable de partir sur une solution propriétaire du marché. Tomcat, lui, convient aux projets dans lesquels il est nécessaire de mettre en place une application spécifique, dans le cas où c'est l'entreprise utilisatrice de l'application qui est en charge des développements.

Evolution possible de la solution actuelle

L'architecture installée répond bien aux besoins actuels. Cependant, si nous mettons à disposition d'autres fonctionnalités ou que le nombre d'utilisateurs augmente, il sera alors nécessaire de la revoir. L'objectif est de toujours proposer une bonne qualité de service.

Si nous voulons par exemple gérer plus d'utilisateurs concurrents et améliorer la disponibilité de l'application, il serait alors nécessaire de mettre en place un second serveur Apache. En effet, le nombre maximal de clients pouvant être géré par un serveur Apache s'élève à 256. Il est préférable d'installer ce second serveur WEB sur une autre machine. Dans ce cas, l'application continuera de fonctionner malgré l'indisponibilité de l'un des deux serveurs. Il faut s'assurer que tous les composants de l'application sont dupliqués sur les deux machines.

Il sera également nécessaire de mettre en place une solution de répartition de charge sur les deux serveurs Apache *(Figure 159)*. Pour ce faire, nous pouvons utiliser une solution logicielle basée sur un serveur Maître (master) et un serveur de secours (backup).

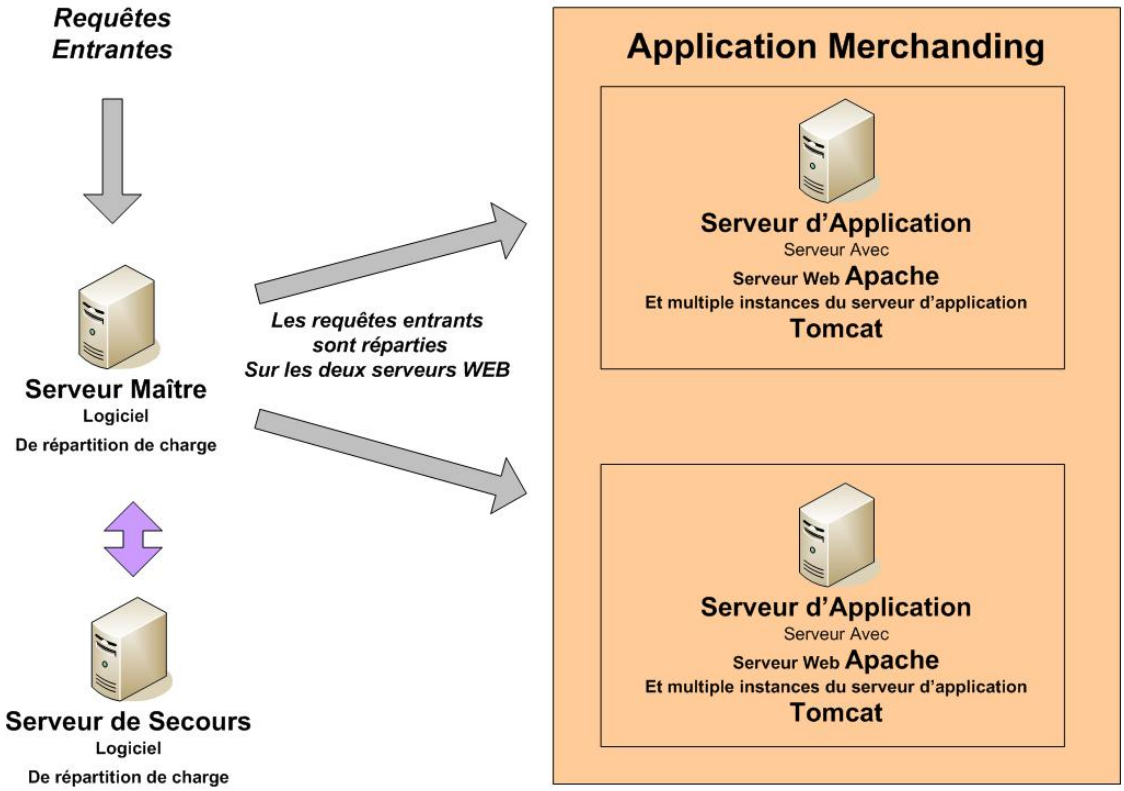


Figure 159 : Répartition effectuée à l'aide d'une solution logicielle

Une autre solution consiste à utiliser un boîtier de répartition avec ou sans backup *(Figure 160)*.

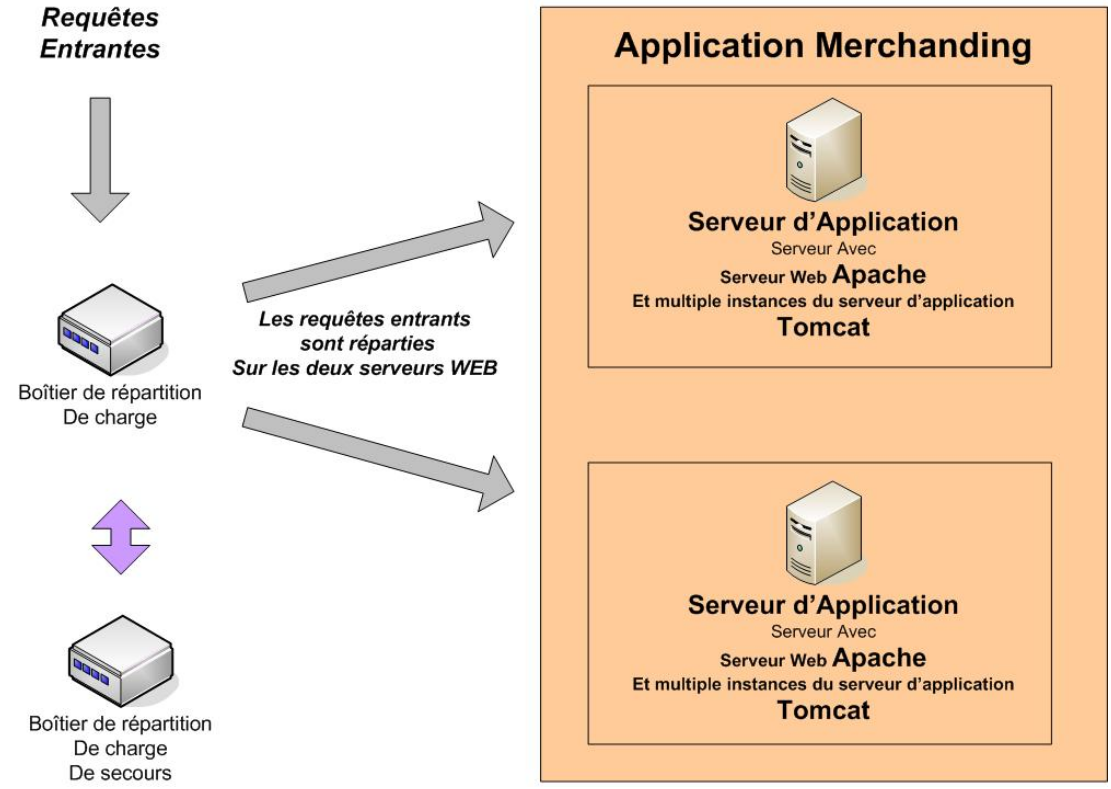


Figure 160 : Répartition effectuée à l'aide d'une solution Hardware

La haute disponibilité

En ce qui concerne la mise en place d'un mécanisme permettant le partage des sessions, compte tenu de l'utilisation minimaliste des sessions et du mode de fonctionnement que nous avons adopté, ce point n'est pas prioritaire.

L'utilisation intensive des sessions est un facteur de dégradation des performances des serveurs d'applications. Plus on conserve d'informations en session et moins le nombre d'utilisateurs supportés par une instance Tomcat est important.

Les développements

Lors du développement de l'application, il faut éviter de gérer trop de choses en session et bien libérer les ressources correspondantes lorsqu'elles ne sont plus utilisées. Le code de l'application doit être conçu en gardant la configuration du cluster à l'esprit. Si les implications du cluster ne sont pas prises en compte dans les phases d'étude, le code peut nécessiter d'être complètement réécrit pour fonctionner au sein du cluster, ce qui peut représenter un coût important.

En règle générale, plus les développements sont simples et plus la capacité pour un serveur d'applications de gérer un nombre important d'utilisateurs augmente.

La complexité du clustering

La mise en place d'un cluster ou l'ajout de membre dans un cluster ne résout pas les problèmes de performance de façon systématique. Il est impératif d'effectuer au préalable une analyse du système, d'isoler les goulots d'étranglements.

On peut ainsi lever d'autres points de blocage qui n'ont rien à voir avec la mise en place d'un cluster. Ces points de blocage peuvent être de natures diverses et variées. Ils peuvent par exemple correspondre à des logiques applicatives incompatibles avec le déploiement sur de multiples instances Tomcat, à des saturations de bande passante, à des problématiques d'accès aux bases de données…

Le clustering répond bien aux problématiques liées à la distribution de logique dupliquée dans les applications WEB devant présenter des caractéristiques de haute disponibilité.

Cependant, la mise en place d'un cluster n'a pas forcement pour conséquence d'améliorer les temps de réponse d'une application surtout dans le cas de l'utilisation de mécanismes de répartition de charge conjointement avec un système de réplication de session. Plus on met de fonctionnalités en place et plus on a besoin de ressources.

Par contre, un cluster permet d'améliorer la disponibilité de l'application. Il permet de répondre à plus de requêtes sans tomber.

Le comportement des utilisateurs

Enfin, le comportement des utilisateurs est très important. Il conditionne l'architecture technique qui doit être mise en place.

Dans le cas du module applicatif Relevé Linéaire RL, une seule instance Tomcat associée à un Apache était suffisante. Cette instance peut supporter 50 utilisateurs de façon concurrente. Les 250 magasins de l'entreprise ont saisi les caractéristiques de leurs meubles surgelés dans une période de temps assez longue. La conséquence a été que la charge s'est répartie sur plusieurs semaines et ceci sur l'ensemble des journées. Il y avait donc très peu de magasins connectés de façon simultanée, en principe seulement un ou deux. Même si cette architecture ne supportait qu'un cinquième des utilisateurs potentiels de façon simultanée, elle s'est révélée largement suffisante.

Dans le cas de la diffusion des planogrammes en magasin, le comportement des utilisateurs n'est pas le même. A chaque sortie de dossier de préconisations sur l'intranet, ils sont tous susceptibles de vouloir y accéder en même temps. Pour répondre à ce comportement, trois instances Tomcat ont été mises en place afin de s'assurer que le site réponde correctement à près de 150 utilisateurs simultanés. Avec les personnes du siège, cette application intéresse près de 450 personnes ou magasins. On est donc capable avec cette architecture de supporter jusqu'à un tiers des utilisateurs potentiels de façon simultanée.

Le choix du cluster répondant aux besoins passe avant tout par une bonne évaluation de l'utilisation qui en sera faite par les utilisateurs et non pas simplement par la résolution de problématiques techniques.

4.4. Bilan Test de Charge

En dessous de 90 utilisateurs simultanés, c'est à dire, un quart de la population totale qui utilise l'application merchandising, les ressources machines sont suffisantes, la mise en place d'un cluster a du sens. Même si une instance Tomcat est indisponible, l'application continuera de fonctionner.

Lorsque le nombre d'utilisateurs est élevé, c'est à dire supérieur à 90 utilisateurs simultanés, cela n'apporte rien. Les temps de réponse sont dégradés.

A ce niveau de charge, le goulot d'étranglement ne se situe pas au niveau du serveur d'applications.

Les temps de réponse HTTP sont moins bons et les trois instances Tomcat consomment plus de CPU qu'une instance unique.

Nous pouvons en déduire qu'un composant très consommateur de ressources conditionne directement les temps de réponse HTTP.

En fait, c'est le processus Oracle qui consomme toute la CPU, comme montré sur le graphe ci-dessous (Figure 161). Le graphe indique 400 % de consommation maximale de la CPU car la machine est un quadri processeurs et que l'indicateur correspond à la somme des consommations de chacun des CPU. Ce processus consomme jusqu'à 88 % de la CPU de la machine.

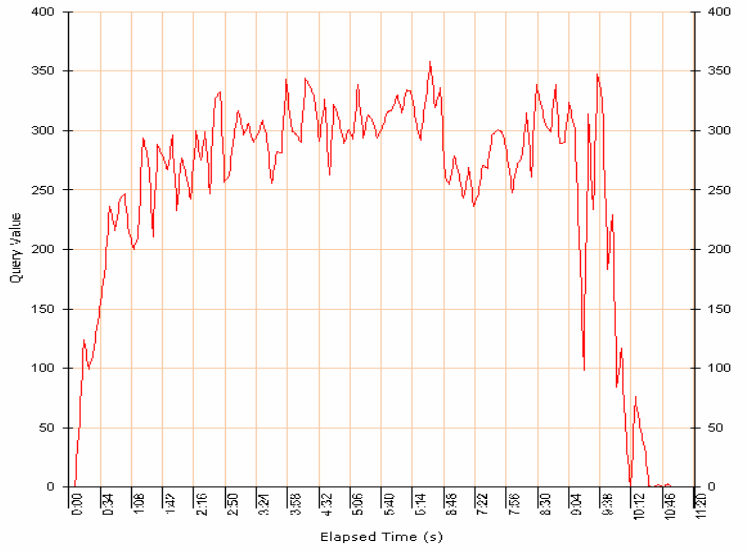


Figure 161 : Consommation en CPU du processus Oracle

La CPU non consommée par Oracle est partagée entre le serveur WEB Apache et les différentes instances de Tomcat.

Pour bénéficier pleinement de la mise en place d'un cluster, il serait nécessaire d'isoler l'instance Oracle sur un serveur indépendant. Ce qui permettrait de libérer de la CPU pour le serveur WEB et les serveurs d'applications.

Pour diminuer le besoin en ressources de la base de données nous pouvons envisager d'optimiser les requêtes de l'application. Cela peut être un axe d'optimisation important. Nous pouvons, par exemple, créer des index sur les différentes tables afin de diminuer le temps d'exécution des requêtes SQL.

Au vu des tests effectués, nous pourrions conclure que la mise en place d'un cluster comportant plusieurs instances Tomcat ne sert à rien. Ce qui est un peu hâtif. Le fait d'avoir plusieurs instances de Tomcat présente l'intérêt de pouvoir disposer de plus de mémoire vive par rapport à une instance unique. Nous pouvons ainsi, dans certain cas, éviter les problèmes de débordement de mémoire.

Ce n'est pas le goulot d'étranglement que nous avons mis en évidence, mais cela pourrait le devenir, si nous isolons Oracle sur un serveur dédié. En effet, si Oracle est capable de répondre de façon plus rapide, il faudra alors que les applications serveurs supportent plus d'utilisateurs de façon simultanée. Ce qui veut dire qu'ils auront besoin de plus de mémoire vive afin de gérer les informations en sessions correspondant aux utilisateurs. Il arrivera un moment où la mémoire vive disponible ne sera plus suffisante pour gérer l'ensemble des utilisateurs. Pour éviter alors le débordement de mémoire, nous devons augmenter la mémoire disponible associée aux JVM des applications serveurs Tomcat ou dans le cas où ce ne serait plus possible d'augmenter le nombre d'applications serveurs.

Un autre intérêt de la mise en place d'un cluster est d'assurer que l'application continue de répondre en cas de défaillance matérielle. Nous devons alors installer des instances Tomcat sur une autre machine. Mais cela suppose que la répartition de charge ne soit pas effectuée par les même serveurs, car si le serveur Web est installé sur un des deux serveurs et que c'est ce serveur qui est défaillant, la répartition de charge ne fonctionnera plus. Le module effectuant le load balancing est un point de défaillance unique. Il faut donc mettre en place un système de répartition de charge avec un mécanisme de backup.

Suite aux tests de performance que nous venons d'effectuer et aux analyses de leurs résultats, nous pouvons nous apercevoir que l'optimisation d'une application est un travail fastidieux et long. La levée d'un goulot d'étranglement et sa résolution nécessitent de nouveaux tests de montée en charge qui font apparaître de nouveaux goulots d'étranglement.

C'est pour cette raison qu'il est très important de définir des objectifs précis au début des tests de montée en charge sinon on risque de dépenser beaucoup d'énergie à vouloir mettre en place un système idéal.

On peut considérer que le système offre des performances satisfaisantes lorsque les temps de réponse correspondent aux attentes et ne s'allongent pas trop par rapport à la moyenne, quand le système est utilisé de façon simultanée par un nombre d'utilisateurs, que l'on a déterminé.

D'autres critères peuvent être utilisés pour déterminer si le système répond correctement. On peut par exemple considérer que les temps de réponse sont acceptables lorsqu'ils sont inférieurs à 4 secondes ou encore que les applications serveurs et le serveur de

base de données doivent être respectivement sollicités au maximum à 50 % et à 80 % de leurs CPU. Sinon, on considère que le système est trop sollicité.

Nous avons réalisé les tests de charge à l'aide d'Opensta, un outil OpenSource qui présente l'intérêt d'être gratuit. L'utilisation de cet outil est satisfaisante et il répond à nos besoins. Pour ce qui est de l'analyse des logs Apache, Webalizer s'avère très utile. Nous regrettons simplement qu'il n'existe pas de moyen direct avec Webalizer d'obtenir des informations extraites des journaux des différentes instances Tomcats consolidées sur l'ensemble du cluster.

Opensta ne traite que les protocoles HTTP et HTTPS [MAR04]. Si l'application utilise d'autres protocoles, il ne pourra plus être utilisé. Ceci peut arriver lorsque l'on implémente une couche applicative exploitant les Web Services. Le protocole SOAP serait alors utilisé. Il faudrait recourir à d'autres solutions complémentaires pour effectuer des tests de charge sur ces parties particulières du développement.

L'utilisation que nous avons faite d'Opensta nous a permis de stabiliser et de fiabiliser l'architecture à l'aide de tests de montée en charge, qui permettent de lever plus rapidement les erreurs.

A priori, ces tests peuvent également permettre de raccourcir les temps de développement, surtout si le logiciel permet de descendre jusqu'à la classe ou la méthode qui posent des problèmes. Malheureusement, Opensta ne propose pas de telles fonctionnalités.

La difficulté principale des tests de montée en charge est d'arriver à faire un modèle de charge qui permet de traduire le plus justement possible le comportement des utilisateurs. Ce qui est loin d'être évident.

Les utilisateurs peuvent se comporter d'une façon différente de la simulation. Il suffit pour cela qu'ils utilisent de façon fréquente une fonctionnalité très consommatrice de ressources pour que le nombre d'utilisateurs réellement supportés soit bien inférieur à ce qui a été annoncé.

De part sa conception, la charge simulée reste très homogène. Les tests de montée en charge sont fondés sur la capture de scénarios. Des fichiers correspondant aux informations relatives aux différents utilisateurs sont ensuite définis afin de rejouer le scénario pour chacun des utilisateurs. Les tests effectués correspondent bien aux différents utilisateurs mais présentent une homogénéité très forte. Ce n'est qu'un même scénario que l'on va rejouer avec des valeurs différentes pour chaque utilisateur.

De plus, il ne faut pas négliger les scénarios correspondant aux administrateurs. En effet, les tâches d'administration peuvent être très lourdes et très consommatrices de ressources. Ce n'est pas parce qu'il n'y a qu'un ou deux administrateurs qu'il ne faut pas en tenir compte. Si on veut pouvoir lancer les tâches d'administration pendant que les utilisateurs travaillent, il est impératif d'en tenir compte.

Dans la réalité, lorsque 100 utilisateurs différents sont connectés à une même application, ils exécutent tous des tâches de façons différentes.

Pour ces raisons, le système peut ne pas réagir tout à fait de la même façon. Il peut y avoir des différences de résultat entre les tests et la réalité.

Enfin, l'évaluation des performances ne peut être qu'un processus à caractère itératif. Les évolutions de l'application et la mise en place d'une nouvelle fonctionnalité conditionnent le nombre d'utilisateurs supportés. Il est donc impératif d'effectuer des tests de montée en charge après chaque évolution majeure.

4.5. Bilan Rails

Nécessité de plug-in

Ruby on Rails nécessite des plug-ins afin de supporter la notion de clef étrangère ou de clef composite ou la localisation…

Ces plug-ins ne sont pas toujours identifiés comme suffisamment stables pour être utilisés en production.

Base de données

L'utilisation de plusieurs bases de données dans une même application n'est pas gérée de manière native.

Rails impose des conventions sur le nom des tables et des identifiants, ainsi que la présence d'une clef primaire nommée id.

Le respect de ces règles n'est pas toujours évident. Quand on ne respecte pas les règles établies par Rails, il est nécessaire de spécifier un autre comportement que celui par défaut.

Le non respect de ces règles peut survenir dans le cas où la base de données existe déjà ou pour des raisons d'optimisation des traitements qui peut imposer une structuration particulière de la base de données pour minimiser les temps d'exécution.

Limites

Le clustering de plusieurs applications Mongrel n'est pas disponible sous Windows. D'autre part, un serveur Mongrel n'est capable de gérer qu'une seule application. Enfin, les possibilités d'administration d'un serveur sont très limitées.

Points positifs

Ruby est un langage simple clair et concis. Il y a peu de code à maintenir.

Rails bénéficie d'une communauté très active. C'est un framework assez complet qui possède de multiples plug-ins et d'outils.

L'architecture d'une application est fixe et claire.

Discussion

Rails est facile à prendre en main et ne nécessite pas une montée en compétence longue.

Pour certains types de projet, les développements Rails sont jusqu'à 10 fois plus rapide que les développements J2EE.

Rails s'avère très adapté aux applications d'administration qui reposent essentiellement sur la mise à jour de tables de façon classique basée sur des CRUD.

Si la base de données n'existe pas et qu'il n'y a pas de contrainte sur la façon de nommer les objets relationnels, les gains attendus par l'utilisation de Rails peuvent être maximaux. Dans ce cas, il est en effet possible de respecter les conventions préconisées d'où les gains.

La façon de respecter les conventions peut être aménagée si la base de données est utilisée par des traitements ETL afin de minimiser les temps d'exécution.

Compte tenu de la rapidité des développements, Rails peut être utilisé lors d'une phase de prototypage afin de préciser les besoins du client.

La plus grosse inconnue à propos de Rails concerne la tenue en charge. Il pourrait ne pas être adapté à des sites à forte fréquentation et dans le cas de besoins complexes.

La nouvelle mouture de Rails, Rails 2 reposant sur une machine virtuelle devrait limiter cette mauvaise tenue en charge.

4.6. Bilan Gestion du projet

Le chef de projet doit avant tout diriger son équipe en donnant des objectifs clairs et un environnement de travail stable.

Si un rapport d'avancement fait apparaître un retard, il est nécessaire de prendre des décisions afin de maîtriser le déroulement du projet. On peut alors augmenter les capacités de travail en intégrant de nouveaux membres dans l'équipe, diminuer l'étendue de la prestation, modifier l'ordre des tâches à effectuer ou être plus souple sur les dates de livraisons.

L'échec d'un projet peut être due à un recours abusif des normes et des méthodes : trop de normes tue la norme.

La gestion de projet ne correspond pas qu'à de la planification dans son coin. Elle implique de diriger une équipe et donc, par voie de conséquence, de prendre des décisions. Dans certains cas, la réutilisation a ses limites et il est plus rapide de refaire que de vouloir réutiliser à tout prix.

Pour ce qui est du management, on peut considérer qu'il est bon lorsque le chef de projet met à profit le potentiel de chacun des membres de l'équipe. Pour cela il est nécessaire

d’éliminer les obstacles rencontrés, en particulier ceux sur le chemin critique. Il est également important que les règles établies soient stables et que les opinions de chacun des membres de l’équipe soient écoutées. Il est important d’écouter les idées et de faire participer au maximum les membres de l’équipe. Par exemple, pour l’estimation des charges, il est préférable de faire valider la charge par la personne qui doit exécuter la tâche correspondante.

Une attention toute particulière doit être apportée à l’autonomie, à la formation et à la montée en compétence. Le travail des collaborateurs ne doit pas être sans cesse interrompu. Leur motivation et leur implication ne pourront être importantes que s’ils ont le sentiment de s’accomplir dans leurs travaux et qu’ils se sentent soutenus et écoutés. Il est important d’établir une confiance réciproque entre les différents acteurs. Le chef de projet n’est pas là pour abuser des membres de son équipe mais il doit faire en sorte qu’ils puissent avoir un rythme tenable et ainsi équilibrer leur vie professionnelle et privée. Demander à ses équipes de faire des heures supplémentaires à ne plus en finir et de tenir un rythme trop élevé ne fait que masquer un problème latent et débouchera à coup sûr sur des situations ingérables pouvant conduire au départ de certains membres de l’équipe.

Le management de projet a une fonction de catalyseur. Il permet d’atteindre les objectifs du projet au travers de la réalisation et la gestion de la documentation, de la communication et de la planification. Il doit permettre de remplacer les éléments inadaptés au projet ou en modifier la combinaison.

Conclusion

Choix logiciels

Afin de mettre en place l'application merchandising, nous avons choisi la solution propriétaire qui nous paraissait la plus simple à mettre en place et qui répondait le plus à nos besoins. Compte tenu des contraintes fortes que représente le fait de travailler avec un éditeur, nous avons rapidement fait le choix de gérer tous les aspects « métier » dans un outil spécifique, qui sert à alimenter le paramétrage du progiciel, ceci pour une meilleure réactivité et éviter de subir des coûts abusifs.

Le fait de gérer les notions « métier » dans une application spécifique permet une meilleure maîtrise des risques et un meilleur contrôle d'éventuels dérapages du projet. L'offre de l'éditeur est considérée comme un outil permettant l'élaboration des planogrammes qui doit rester indépendant des évolutions des règles « métier ». Lors d'évolutions, les impacts sur le progiciel sont alors minimisés.

ETL / EAI

Afin d'intégrer l'application merchandising constituée du progiciel et de l'application spécifique dans le Système d'Information, nous avons utilisé l'ETL GENIO. Cet ETL couvre de façon satisfaisante nos besoins à l'aide de quelques outils complémentaires : la mise en place d'agents sur les serveurs autres que le serveur de l'ETL, l'installation de services permettant le transfert via FTP ou le lancement d'exécutables et de traitements sur une machine distante avec REXEC.

L'utilisation d'un EAI tactique ou d'un ETL possédant un certain nombre d'adaptateurs applicatifs simplifie les développements mais présente l'inconvénient d'être très contraignante. Il est nécessaire que les adaptateurs proposés soient compatibles avec la version du système cible, ce qui n'est plus garanti après une migration de version. Enfin, les coûts de licence de ces adaptateurs sont très importants et on est obligé de se conformer aux comportements pour lesquels ils ont été prévus. Si on souhaite mettre en œuvre quelque chose pour laquelle ils ne sont pas prévus, par exemple la mise en place d'agents locaux sur des serveurs distants, la tâche devient vite ardue.

L'utilisation d'une telle solution peut être envisagée lorsque la présence d'un adaptateur propre à une application permet d'éviter d'utiliser des API dont l'emploi peut s'avérer lourd et fastidieux.

Dans le cas contraire, on peut préférer concevoir un framework afin de mettre en place des traitements locaux à une application. L'utilisation d'un framework permet d'augmenter la réutilisation des différents composants et de faciliter la maintenance. De plus, la mise en place du paramétrage de ces traitements à l'aide de fichiers de configuration XML permet de diminuer les modifications de code.

L'utilisation d'un EAI prend tout son sens lorsqu'il est mis en place au niveau d'un ensemble d'applications d'une entreprise et non pas au niveau d'une seule application.

Architecture SOA

La question est de savoir si c'est vraiment vers une solution EAI qu'il faut se tourner pour intégrer les différentes applications qui constituent le Système d'Information de l'entreprise.

Depuis quelques années, de nombreux éditeurs tels que Software AG avec son offre Crossvision, proposent des architectures SOA (Service Oriented Architecture).

Cette architecture Crossvision repose sur la mise en place d'un ESB (Entreprise Service Bus) qui est utilisé par les différentes applications de l'entreprise au travers d'un format pivot XML.

La solution offre de nombreux connecteurs qui permettent de faire interagir l'ESB avec de l'EDI, MOM, ETL et les ERP utilisés par les entreprises.

Les données référentielles sont gérées à l'aide d'une application MDM (Master Data Management) afin d'éviter la duplication des données référentielles dans chacune des applications.

Les différents services sont gérés par le biais d'une solution BPM (Business Process Management). Le BPM désigne l'ensemble des méthodes et des outils nécessaires pour traiter la totalité du cycle de vie des processus. L'objectif est de permettre à l'entreprise d'identifier et d'extraire les processus « métier » des applicatifs afin de les orchestrer selon la logique choisie par l'entreprise avec des outils dédiés.

Il est alors possible de mettre en place des flux entre les différentes applications aussi bien en mode batch qu'au fil de l'eau. Les utilisateurs peuvent valider les différentes étapes des processus « métier » grâce à un workflow.

Le projet merchandising à lui seul ne justifie pas la mise en œuvre d'une telle solution. C'est une réflexion qui doit être menée à un niveau macroscopique par les architectes de l'entreprise. Ces derniers, en prenant en compte l'ensemble des applications de l'entreprise, seront à même de faire le bon choix.

L'ETL Powercenter 7 d'Informatica

De la même façon, le choix d'une solution ETL doit être un choix d'entreprise. Son utilisation prend tout son sens lorsqu'on n'en utilise qu'un afin de réduire la complexité du Système d'Information de l'entreprise.

L'ETL Genio a été le premier utilisé au sein de l'entreprise. Par contre, un audit a déterminé que l'ETL Powercenter d'Informatica répondait de façon plus satisfaisante aux besoins de l'entreprise que l'ETL GENIO. Informatica est leader sur ce marché et investit énormément en recherche et développement.

Le remplacement d'un ETL par un autre suppose la réécriture de l'ensemble des processus « métier ». Ce qui ne peut être fait qu'à raison de forts investissements ou au fil de l'eau compte-tenu de l'ampleur du projet, sachant que GENIO est utilisé depuis près de dix ans au sein de l'entreprise.

Les bons temps de traitements représentent la force de powercenter par rapport à ses concurrents.

Par rapport à GENIO, le designer de Powercenter offre des fonctionnalités en moins telles que la prise en compte de répertoires partagés sous Windows qui permet d'écrire directement dans un fichier distant. On est obligé de créer d'abord le fichier sur le serveur de l'ETL avant de le transférer via FTP.

De plus, il ne permet pas de gérer la notion d'événement comme le propose GENIO. Avec GENIO, sur l'envoi d'un événement, on peut déclencher des traitements contrairement à Powercenter où on passe par la création de fichiers qui servent de flags sur le serveur. Powercenter détecte la création du fichier, le workflow associé à la création du fichier flag est lancé. Ceci suppose la mise en place de services FTP pour les transferts de fichiers.

Powercenter manipule des mappings, ce qui correspond aux modules sous GENIO. Ces mappings sont associés à des sessions. C'est une notion qui n'a pas d'équivalent sous GENIO. Et les sessions sont regroupées en workflow qui correspond au process de GENIO. L'introduction de cette notion de session permet la prise en compte de différents environnements via des fichiers de paramétrage. Ceci n'est pas possible sous GENIO. On est obligé de dupliquer des traitements pour faire l'équivalent.

La notion de mapping oblige à avoir une source et une cible. Il n'est donc pas possible de faire des suppressions dans une base de données sans mettre une source fictive en entrée du mapping.

Powercenter est en une certaine mesure plus contraignant que GENIO mais a l'avantage d'offrir des meilleures performances. Les notions utilisées dans les deux produits sont très similaires, ce qui permet à une personne connaissant GENIO de rapidement prendre en main Powercenter.

Au-delà de l'ETL utilisé, l'organisation joue un rôle déterminant dans la réactivité lors de la mise en place des traitements. L'architecture Informatica repose sur trois environnements : test, pré-production et production, dont la gestion est assurée par un prestataire spécialisé, ce qui alourdit les cycles de mise en production des traitements à mettre en place.

L'utilisation d'un ETL permet une certaine souplesse par rapport aux évolutions du Système d'Information de l'entreprise. D'autant plus qu'il est géré en direct dans le cas de GENIO.

Outil spécifique

Le fait de vouloir gérer nous même les notions « métier » dans une application spécifique nous a imposé un choix de plateforme de développement. Notre choix, effectué en 2003, s'est porté tout naturellement sur J2EE qui offrait une solution très structurée, un cadre clair, une architecture et une infrastructure puissantes.

Mais J2EE présente certains inconvénients. Il est complexe, lourd, long à maîtriser et parfois difficile à adapter. Son utilisation suppose que l'équipe de développement soit correctement dimensionnée.

D'autres offres étaient également présentes sur le marché. Parmi elles, nous pouvons citer : .NET et php.

.NET de Microsoft est une offre propriétaire et coûteuse.

Php présente l'avantage d'être interactif, rapide et facile à apprendre. Il ne comporte pas beaucoup de contraintes et est très souple, mais le code produit est peu structuré, difficile à maintenir, et il n'existe pas de séparation entre la logique et le rendu.

Ruby on Rails

Dans la mouvance du WEB 2.0 sont apparues des nouvelles solutions afin de mettre en oeuvre les sites Web de façon différente. Parmi les nouvelles technologies, est apparu le très médiatique Ruby on rails [SAU06]. Rails est un framework basé sur le langage Ruby. Ce Framework permet, entre autres, de proposer des fonctionnalités AJAX sans devoir écrire du code javascript complexe. Il propose également des outils permettant de mettre en place des Web Services.

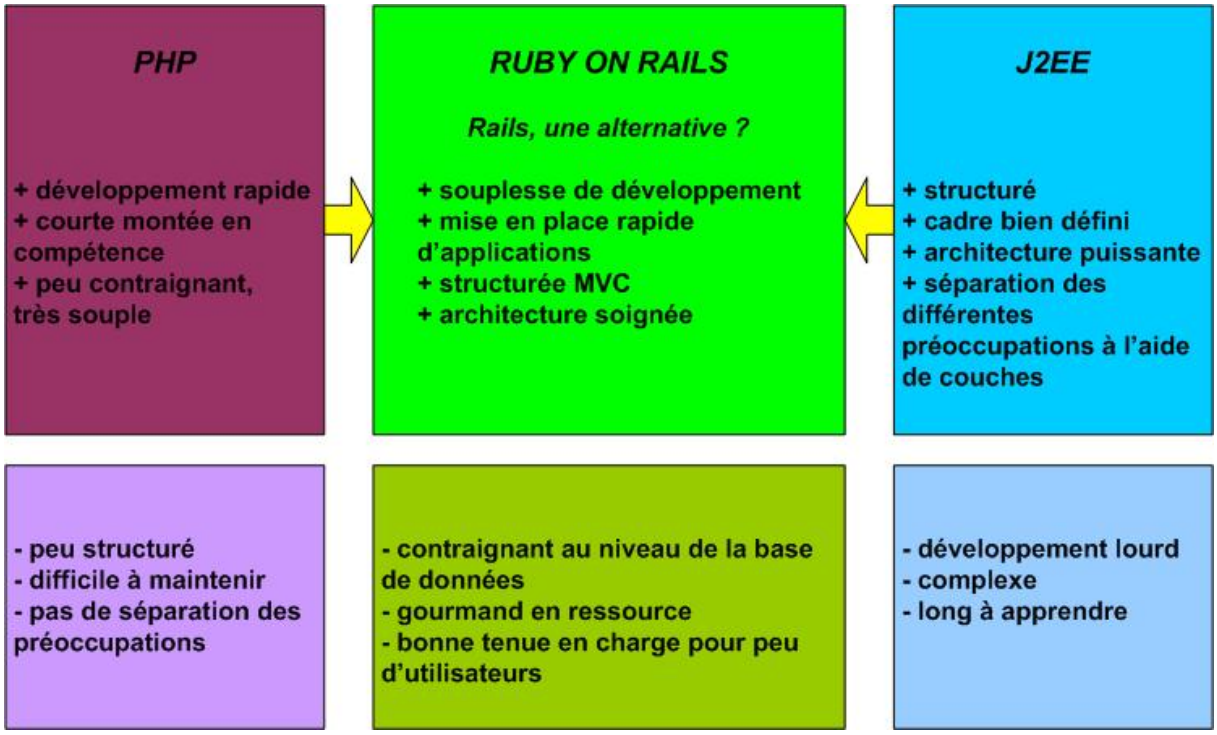


Figure 162 : Ruby on Rails

Rails permet de mettre en œuvre de façon rapide, facile et souple des sites WEB tout en présentant un cadre très structuré. Il permet de concilier la souplesse de développement avec des architectures d'application soignées.

Le langage ruby possède une syntaxe très simple. De plus, Rails utilise de nombreuses conventions de nommage qui simplifient efficacement les développements de sites WEB.

Il permet de mettre en œuvre des applications reposant nativement sur le modèle MVC.

Dans le cas où il n'y aurait pas de contrainte forte par rapport à la base de données et que le nombre d'utilisateurs du site ne serait pas élevé, il est possible de gagner énormément de temps par rapport à une utilisation classique de J2EE.

En effet, les conventions utilisées par Rails imposent de nommer les colonnes de la base de données de manière particulière pour permettre une gestion native des clefs primaires. Les colonnes correspondant aux clefs doivent s'appeler ID. On est donc amené à modifier le modèle de la base de données, ce qui peut poser des problèmes dans le cas où la base de données serait utilisée par d'autres technologies. Toute modification du modèle de la base de données a alors pour conséquence de revoir les développements effectués à l'aide de ces autres technologies.

Le nombre d'utilisateurs quant à lui ne doit pas être élevé. Ruby étant un langage interprété a besoin de plus de ressources que des langages basés sur une machine virtuelle tels que java ou des langages compilés tels que le C.

Pour que des sites WEB développés à l'aide de Rails supportent de nombreux utilisateurs, il est indispensable de se pencher sur l'étude des architectures à mettre en place, ce qui peut nécessiter des connaissances pointues dans le domaine des serveurs d'applications Mongrel et la mise en cluster de ces serveurs avec un serveur Web Apache ou Lighthttpd. Par défaut, Rails est installé avec un serveur WebRick qui ne supporte pas des charges importantes.

Rails convient tout à fait aux développements de sites d'administration WEB qui sont utilisés par peu d'administrateurs et ne présentent pas d'innovation dans leurs interfaces. Ces applications effectuent essentiellement de la mise à jour de tables de base de données (CRUD) ou de la lecture de tables.

De nombreux aspects de Rails simplifient la vie des développeurs et permettent de gagner du temps. Le mapping relationnel objet est quasi immédiat. Les conventions de nommage font disparaître les fichiers de configuration tels que le struts-config.xml de Java.

On peut ainsi espérer diminuer de 70% les temps de développement en utilisant Rails en lieu et place de Java pour les interfaces d'administration WEB.

Il n'est pas nécessaire d'avoir un nouvel IDE. Il existe un plug-in Eclipse permettant de faire du Rails. On peut ainsi effectuer les développements Java et Rails dans un seul et même IDE.

On peut donc conserver J2EE pour implémenter les frontaux WEB susceptibles d'être utilisés par des centaines d'utilisateurs et se servir de Rails pour les consoles d'administration.

Cette alternative peut représenter un gain important pour le projet. Le poste de dépense le plus important correspond aux développements WEB. Et dans le cas de l'application merchandising près de 80% des écrans sont dédiés à l'administration.

Architecture

Avec l'introduction de nouvelles technologies telles que Rails, l'architecture de l'application n'est pas à remettre en cause du fait de sa structuration en modules. Un module de l'application pourra être basé sur une autre technologie.

L'architecture J2EE mise en place répond à nos besoins. La seule évolution envisageable à court terme consiste à isoler la base de données sur un serveur indépendant afin que les serveurs d'applications puissent supporter une charge plus importante.

Il n'est pas nécessaire de la revoir, à moins de vouloir introduire une véritable couche « métier » dans les applications. Cela suppose alors l'utilisation de serveurs d'applications tels que Jonas ou JBOSS qui permettent de mettre en place les EJB.

Base de données

Si on reste dans la même optique de réduction des coûts, on peut envisager de remplacer le SGBD Oracle par MySQL. Les coûts licence de ces deux offres étant très différents. MySQL est beaucoup moins coûteux.

Cette migration suppose de revoir, dans le cas des développements J2EE, la couche d'accès aux données, en particulier les fichiers de configuration du mapping ORM. Certaines requêtes spécifiques seront également à revoir.

Tests de montée en charge

Pour ce qui est des tests de montée en charge, l'utilisation d'Opensta est satisfaisante même avec l'emploi de Rails. Son utilisation reste pertinente afin de bien calibrer le système, même si le nombre d'utilisateurs virtuels supportés ne correspond pas toujours à la réalité du fait qu'il n'est pas facile de prévoir le comportement des utilisateurs.

Tests fonctionnels

En complément des tests de montée en charge, afin de vérifier la qualité des développements et la non régression lors de la mise en place d'évolutions, on peut effectuer des tests fonctionnels.

On peut utiliser l'outil Selenium. Il permet de tester la compatibilité de l'application avec les différents navigateurs du marché ou de vérifier les fonctionnalités de l'application.

Cet outil permet d'effectuer des tests fonctionnels à partir d'un navigateur Web tel que FireFox. Il est simple à prendre en main. Il permet d'enregistrer de façon rapide des scénarios et de les rejouer.

Il est possible de positionner des contrôles de présence de texte dans les pages de résultat. Dans le cadre d'un test, on peut faire des actions sous forme de Javascript. Via Selenium RC, il est également possible d'utiliser d'autres langages tels que Java, Ruby, Python, Perl ou C#.

L'interface est très conviviale. Lorsque l'on rejoue un test fonctionnel, les actions qui ont échoué apparaissent en rouge permettant une identification rapide des actions en échec. On peut jouer un test étape par étape ou modifier des enregistrements de scénario à partir d'une action donnée.

Application merchandising

L'application merchandising répond aux besoins des utilisateurs. Elle permet, par sa simplicité d'utilisation et l'automatisation des tâches manuelles, de gagner du temps. La productivité de la cellule merchandising s'en trouve améliorée. Les merchandisers sont ainsi capables de produire davantage de dossiers de préconisations.

De plus, la suppression des impressions papier au profit de la diffusion via l'intranet a permis de faire des économies importantes. Dans ce cas, les problèmes n'étaient pas que techniques. La résistance au changement de certains utilisateurs a été assez vive. Il est important de ne pas oublier le volet humain. Dans tout projet, il est nécessaire de faire adhérer et de rechercher l'approbation par les utilisateurs du projet.

Néanmoins, la solution telle quelle ne propose pas aux administrateurs des outils satisfaisants en terme de paramétrage et de contrôle. Il y a un besoin fort de transparence au niveau des traitements d'alimentation de la solution.

Il est nécessaire d'enrichir la partie d'administration du site afin de pouvoir superviser les traitements d'alimentation de la solution, détecter de façon rapide les erreurs et les anomalies à l'aide d'alertes, gérer leurs recyclages. Nous devons nous doter d'un véritable outil de gestion de la qualité des données et de Reporting.

Ceci doit être mené à bien tout en répondant aux évolutions demandées par les utilisateurs telles que la gestion des différentes enseignes du groupe MONOPRIX. Nous voulons nous doter d'une unique solution de merchandising pour l'ensemble du groupe.

D'autre part, il est nécessaire d'aller vers une meilleure intégration de l'application de merchandising et de l'application permettant de gérer les assortiments de produits. Cette dernière permet de gérer les assortiments qui sont ensuite utilisés par l'application merchandising.

Un projet à plus long terme consisterait à mettre en place un progiciel permettant l'élaboration de plan magasin (Annexe I). A l'aide d'un plan Autocad et des informations récupérées de l'application merchandising, nous serons capables d'optimiser la rentabilité de la surface de vente d'un magasin dans sa globalité.

Pour finir, notre objectif reste toujours le même : répondre à un maximum de besoins avec des coûts réduits dans les meilleurs délais avec le souci de mettre en œuvre des solutions simples, maintenables, évolutives dans un cadre bien défini.

Gestion de projet

La gestion de projet est basée sur la recherche constante de l'amélioration. Le but n'est pas de respecter des méthodes ou des pratiques mais d'obtenir des résultats concrets avec la préoccupation d'arriver à atteindre un bon niveau de qualité, de produire un logiciel de qualité, évolutif, intégrable, réutilisable. On recherche l'efficacité et non un respect absolu d'un règlement.

C'est un aspect crucial de la gestion de projet lié aux méthodes Agiles.

XP est une méthode de l'extrême dont les principaux inconvénients sont la forte sollicitation des équipes et la légèreté du formalisme.

C'est pour cela qu'il est important de bien veiller à ce que le rythme soit tenable et faire en sorte dès qu'on le peut de documenter ce qui a été fait.

Cette méthode est très appropriée dans le cas de petites équipes et peut se révéler insatisfaisante dans le cadre de projets importants impliquant des centaines d'ingénieurs.

La méthode que nous avons utilisée est inspirée d'XP. La programmation en binôme qui est l'un des aspects les plus décrié d'XP n'est pas utilisée de manière systématique mais elle est fréquemment pratiquée. Nous pouvons ainsi répondre de façon plus rapide à des problématiques épineuses.

Aujourd'hui les techniques utilisées dans les projets informatiques évoluent vite. Il n'existe pas de méthode de gestion de projet informatique universelle.

Nous pouvons nous demander si nous ne sommes pas parfois dépassé par les avancées technologiques qui supposent de plus en plus de connaissances à maîtriser et nécessite davantage de ressources humaines.

Les méthodes de modélisation ne permettent pas toujours de répondre aux problématiques changeantes associées à un environnement en constante évolution.

La réalisation de ce projet est basée sur une démarche scientifique. L'expérimentation vient compléter les aspects théoriques. Les tests ont un rôle central. Nous pouvons ainsi mieux gérer les risques et obtenir une meilleure conduite de projet.

Par analogie à CMM, nous pouvons dire que la façon dont est mené le projet correspond à un niveau 2, un niveau qualifié de reproductible. Le processus de gestion de projet n'est pas clairement établi et n'est pas connu de tous.

La gestion du projet est basée sur l'expérience des anciens projets. Le fonctionnement du projet repose sur l'engagement permanent des ressources.

La répartition des rôles dans l'organisation de l'équipe a un inconvénient majeur : trop de rôles sont concentrés sur le chef de projet. D'après ce constat, nous pouvons considérer que l'équipe est sous dimensionnée. Il serait certainement nécessaire d'avoir une personne de plus afin d'épauler le chef de projet.

D'autre part, il est préférable que le chef de projet ne prenne pas en charge une partie complète des développements. Il y a au moins deux inconvénients majeurs à cela :

- le chef de projet ne peut pas se concentrer de façon continue aux rôles principaux qui lui sont attribués
- Il est le seul à avoir la connaissance de certaines parties des développements. Dans le cadre de ce projet, les ingénieurs d'étude prennent en charge les développements J2EE et RAILS. Le chef de projet prend en charge les développements Oracle, PACBASE et ETL ainsi que le cadrage des développements J2EE et RAILS et l'intégration de l'ensemble.

L'équipe ne comporte que 3 personnes. Elle est sous dimensionnée. La gestion de projet XP repose normalement sur une équipe de 6 personnes sans compter le remplaçant du chef de projet lors des absences de ce dernier.

Il est cependant nécessaire que les charges soient suffisantes afin de fournir des tâches en quantité suffisante à l'équipe projet.

Si les charges correspondantes au projet ne sont pas suffisantes, il peut être envisageable que l'équipe prenne en charge un autre projet, afin d'être en accord avec les réalités économiques et d'avoir une équipe autonome même en l'absence de certains de ses membres.

Un autre point d'amélioration serait la documentation du processus suivi lors du projet et la mise à disposition de ce document à l'ensemble des acteurs du projet. Ceci permettrait que toutes les personnes impliquées aient une bonne visibilité sur la façon dont est géré le projet auquel ils participent. Et surtout, cela leur donne la possibilité d'améliorer, voire dans un second temps d'optimiser la méthodologie projet. La diffusion des informations pourrait d'autre part être plus rapide en incluant ces informations directement dans l'application.

L'avantage est alors d'avoir une version unique consultable des documents relatifs à la conduite du projet permettant d'éviter le risque d'avoir plusieurs versions des documents dans des répertoires distincts.

Par la diffusion de l'information, la collaboration au sein de l'équipe est largement améliorée, ce qui favorise la réussite du projet.

L'utilisation d'une application de gestion de projet pourrait également être un plus, ainsi que l'utilisation d'une application de suivi d'anomalies. L'avantage serait de permettre une centralisation de l'information relative au projet.

Mes apports personnels :

Au sein du projet de merchandising, qui est l'objet de ce mémoire, j'occupe la fonction de Chef de Projet Informatique (CPI).

Gestion de projet utilisée

Dans ce projet, la maîtrise d'ouvrage a su être suffisamment flexible pour déterminer le périmètre fonctionnel et fixer des priorités aux fonctionnalités demandées ainsi que les niveaux de criticité. Cela a permis de piloter le projet en jouant sur le périmètre fonctionnel sans faire de concession sur la qualité du logiciel, les coûts ou les délais.

Grâce à cet état d'esprit, le projet a pu être géré en s'inspirant des méthodes d'eXtreme Programming comme on a pu le voir en parcourant ce mémoire. Le fait de dissocier la résolution des problématiques techniques et des problématiques fonctionnelles, comme c'est le cas dans un processus de type 2TUP, fut un autre atout de la gestion de projet utilisée.

Cette approche de la gestion de projet n'est pas répandu au sein de Monoprix et fait figure d'une approche peu structurée du fait qu'elle ne repose pas sur la réalisation d'une documentation exhaustive avant de commencer les développements.

C'est cependant l'approche qui correspond le mieux au projet que nous avons à mener compte tenu du contexte Monoprix qui s'avère très changeant et dans lequel il est difficile, voire impossible, de faire une liste complète des besoins utilisateurs et de figer ces besoins. Le risque de suivre une démarche traditionnelle est de fournir à l'utilisateur au bout de quelques semaines ou quelques mois une solution informatique qui ne répond plus aux besoins du moment.

Le processus que nous avons suivi nous permet de diminuer les coûts de mise en place des solutions informatiques dont a besoin l'entreprise avec des délais courts et de contribuer à sa compétitivité.

Compte tenu des contraintes de coût, de temps et d'évolutivité du cahier des charges, l'effort s'est porté sur la mise en œuvre d'une application immédiatement opérationnelle au détriment d'une documentation exhaustive, laquelle n'a pas été considérée comme un pré-requis obligatoire. Le besoin de documentation n'a pas pour autant été négligé mais satisfait lorsque cela était indispensable. Lorsqu'un problème se présente une alternative peut être envisagée :

- soit mettre par écrit ce qui est envisageable de faire pour le résoudre, au risque de ne plus avoir le temps d'élaborer la solution envisagée et donc d'échouer
- soit raisonner le problème de la même façon sans le mettre par écrit, puis le résoudre et enfin mettre par écrit la solution retenue.

L'avantage de la deuxième approche est de résoudre le problème dans les délais. On peut comparer cela à l'intervention de pompiers sur un bâtiment en flamme. Ces derniers ne commencent par faire un cahier des charges complet avant de passer à l'action. La documentation initiale est minimaliste : un ordre de mission et, suite à l'intervention, une documentation plus fournie peut être élaborée lorsque plus rien ne brûle : un compte rendu

d'intervention. Cette solution permet au final d'être plus compétitif en permettant de répondre aux besoins de l'entreprise avant ses concurrents.

De plus, la conduite du projet a été guidée par l'expérimentation. Les solutions proposées sont élaborées à l'aide de tests. Les tests sont effectués de façon systématique comme on peut le voir dans l'illustration de la démarche utilisée dans le projet. La modélisation d'une solution est validée par des expériences, ce qui permet à tout moment la mise en place de solutions fonctionnelles qui offrent un service de qualité et une bonne disponibilité. En sciences, l'expérience est en quelque sorte le garde fou de la théorie. Par analogie, en informatique, les tests permettent de valider ou non les modèles des solutions qui sont mises en place.

La responsabilité d'un projet implique d'aller de l'avant et d'avoir un rôle moteur. Cela permet d'améliorer de façon continue les solutions informatiques choisies afin de répondre aux besoins des utilisateurs. Le fait de rechercher cette amélioration continue et de prendre en compte les modifications des besoins tout au long du projet peut être perçu comme étant un phénomène d'effet tunnel. Le projet est composé d'une succession d'itérations et on peut se demander s'il va s'achever un jour. C'est une façon de concevoir l'élaboration d'un logiciel qui s'oppose au cycle séquentiel traditionnel. Néanmoins, des jalons sont clairement identifiables, ce sont les différentes versions des modules correspondant à l'application. Si on reprend l'exemple de l'application merchandising, en quatre ans le module permettant la diffusion des dossiers de préconisations PGC a connu 4 versions. Par contre, les modules d'administration sont sans cesse enrichis en fonctionnalités et sont plutôt en version bêta perpétuelle.

La recherche de la simplicité

L'un des points les plus importants est de veiller à ce que les développements effectués soient homogènes et intégrés de façon intelligente. Une bonne homogénéité permet d'obtenir une solution simple facilement maintenable et évolutive. Une bonne intégration permet quant à elle de diminuer le complexité de la solution et de faciliter la maintenance.

La simplicité de la solution mise en œuvre passe également par le choix et l'utilisation des outils appropriés. Pour cette raison, il est nécessaire de suivre l'évolution des nouvelles technologies au travers de la veille technologique. Il faut garder un esprit ouvert sans sombrer dans l'over engineering en voulant à tout prix utiliser la dernière technologie à la mode.

Rôle élargi de chef de projet

En tant que chef de projet, j'ai été amené à travailler sur l'intégralité des étapes du processus suivi dans le projet. J'ai donc largement « débordé » de la fonction traditionnellement attribuée à un chef de projet, ceci afin de mener à bien ce projet.

Ma participation va de l'assistance à la maîtrise d'ouvrage jusqu'au suivi de production ou la maintenance, le dépannage et le support téléphonique aux utilisateurs finaux de l'application.

La collaboration avec la maîtrise d'ouvrage a été basée sur l'écoute, le dialogue et une implication forte sur les problématiques fonctionnelles. Cette approche a conduit à remettre en

cause la structuration fonctionnelle du projet et ne s'est pas limitée à un simple recueil des besoins utilisateurs.

Au-delà de la rédaction d'un contrat et à la réalisation des clauses de ce dernier, mon rôle a été de privilégier la satisfaction du client ainsi que la prise en compte des évolutions de ses besoins.

Les problématiques des utilisateurs ont été prises en charge même si elles n'avaient pas été énoncées. L'anticipation a compté pour beaucoup dans la bonne réussite du projet. Des solutions ont été apportées à des problématiques fonctionnelles non déclarées ou indirectes sans que la maîtrise d'ouvrage ait eu besoin de les soulever.

Plus qu'une simple solution technique aux problématiques que nous avons rencontrées, nous avons élaboré une solution fonctionnelle reposant sur des choix technologiques.

Au début du projet, aucune technologie WEB n'était utilisée. La solution existante reposait sur des applications installées de façon autonome sur les postes des utilisateurs. Des choix technologiques ont été nécessaires. Ils ont été effectués à partir de recherches sur Internet, d'ouvrages et de l'expérience de prestataires, de collègues et de connaissances. La structuration des applications WEB et la mise en place d'une solution centralisée ont été une préoccupation forte du fait qu'elle était définie comme telle par le directeur de l'informatique.

Compte tenu des budgets alloués, j'ai pris en charge les développements correspondant aux bases de données et aux traitements ETL, et j'ai également assuré le cadrage de développements WEB des Ingénieurs d'Etudes (IE) afin d'obtenir une mise en oeuvre structurée et évolutive.

Ma contribution ne s'est pas limitée à l'apport de savoir et de savoir-faire technique. La mise en place d'un mode de collaboration et l'emploi de méthodes, en concertation avec les utilisateurs et les membres de l'équipe, en a été une autre. La recherche d'une solution économiquement compétitive préservant la qualité de la solution mise en œuvre peut être considérée comme un plus apporté au projet.

Un travail collaboratif

Un autre aspect de la gestion de projet est la collaboration entre les individus participant au projet. Il est possible d'utiliser des méthodes de collaboration utilisées dans le domaine sportif ou militaire, mais il est impératif de ne pas négliger la collaboration car elle conditionne la bonne réussite du projet. Une bonne entente doit s'établir entre les membres de l'équipe projet.

Avec le client, la collaboration repose principalement sur l'écoute, la réflexion sur les propositions de ce dernier et la prise en compte des remarques lorsqu'elles sont pertinentes. Dans la méthode suivie dans le projet, le client est positionné comme un acteur ayant un rôle important et la prise en compte des modifications de ses besoins est acceptée. Le changement n'est pas systématiquement perçu comme un manque de réflexion. Il est accepté et l'équipe projet cherche à s'y adapter.

Au sein de l'équipe, la collaboration passe par le travail en binôme. C'est une méthode qui a du mal à être acceptée du fait de son coût. Des critiques du type : « l'un travaille, l'autre regarde » peuvent être fréquentes. C'est pourtant une méthode qui peut être appliquée à toutes les étapes du projet et permettre de trouver des solutions fonctionnelles plus fiables, de résoudre les problématiques techniques de façon plus rapide ou d'améliorer la qualité du code du logiciel par des revues en binôme.

Choix, constitution et développement de l'équipe

Dans tout projet collaboratif, il est nécessaire de considérer à sa juste valeur la dimension humaine. Le choix des membres qui vont composer l'équipe est crucial. Les candidats doivent adhérer au projet et à son mode de gestion. Il est très fréquent de rencontrer des personnes qui n'arrivent pas ou n'acceptent pas de travailler à l'aide de méthodes agiles et préfèrent se réfugier dans des méthodes traditionnelles moins contraignantes pour elles, même si elles ne sont pas adaptées au contexte. Un échec ou le retard d'un projet dû à l'inadéquation d'une méthode traditionnelle peuvent alors passer presque inaperçus. En informatique, les projets coûteux ou qui dérapent en termes de budgets ou de délais sont monnaie courante et, par voie de conséquence, fréquemment tolérés.

Le recrutement des membres de l'équipe n'est pas une chose aisée. Des entretiens ne permettent pas toujours de cerner la personnalité d'un individu et la façon dont il va se comporter au sein de l'équipe. Il vaut mieux alors compter sur des commerciaux ouverts et compréhensifs qui mettront les moyens nécessaires pour résoudre les éventuels problèmes rencontrés. Le maintien d'une bonne relation avec les représentants des sociétés de service que sont les commerciaux et la mise en place de relations « gagnant-gagnant » avec ces derniers permet d'anticiper la résolution de situations difficiles.

Les connaissances, savoir-faire et attitudes dans le travail des utilisateurs et des membres de l'équipe, que ceux-ci soient internes ou en prestation, sont à prendre en considération et ne doivent pas être sous-estimés. Les spécificités de chacun doivent permettre de mener à bien l'objectif commun qu'est la réussite du projet.

Chacun des membres de l'équipe a un rôle

Chaque personne a un rôle à jouer et la bonne réussite d'un projet ne peut pas être attribuée à une personne en particulier.

La réalisation d'un projet, quel qu'il soit, est avant tout un travail d'équipe. Il est important que chaque membre de l'équipe ait un rôle bien défini. Le client et le fournisseur doivent savoir tenir leur rôle et ne pas prendre de décisions sur des domaines dont ils ne sont pas responsables.

Le rôle principal du client, le Chef de Projet Utilisateur, est d'exprimer son besoin, ses priorités et de valider que les réalisations effectuées lui conviennent.

Les Ingénieurs Etudes quant à eux prennent en charge les développements des applications WEB basées sur des technologies J2EE, Rails ou BIRT. Ils travaillent donc sur les étapes de conception détaillée, de codage et de test.

Compte tenu de la petite taille de l'équipe, chacun de ses membres est amené à prendre en charge des tâches qui ne devrait pas lui incomber dans une gestion traditionnelle de projet.

Ce que j'en ai retiré

Pour finir, ce projet m'a permis d'être confronté à des problématiques auxquelles j'ai pu répondre en m'adaptant d'un point de vue technique, méthodologique et organisationnel. Cette démarche a nécessité un fort investissement en termes de montée en compétence sur ces différents domaines. La plupart des technologies employées telles que J2EE, apache, tomcat, RoR m'étaient inconnues au départ du projet. Les méthodologies utilisées et l'organisation adoptée au cours du projet ont été mises en place au fil de l'eau afin de répondre à un contexte changeant. Compte tenu des fortes contraintes de coût, de temps, l'application merchandising n'aurait jamais pu voir le jour sans l'aménagement des règles établies dans l'entreprise à commencer par la façon de gérer le projet et par le recours à des solutions « open source ».

Sommaire détaillé

Liste des Figures

Liste des Tableaux

Index

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Glossaire

2

2TUP	Two Track Unified Process. Processus unifié utilisé pour le développement orienté objet. Les préoccupations techniques et fonctionnelles sont traitées de façon dissociée

A

ActiveX	Composant applicatif conforme au modèle COM/DCOM
Agents locaux	Programmes java reposant sur un framework spécifique maison. Ce framework permet de paramétrer des traitements à la manière de Struts à l'aide d'un fichier analogue au struts-config.xml
AGL	Atelier de Génie Logiciel
AJP	Protocole de communication utilisé par Apache
ANT	Outil permettant de construire des applications de façon automatique
AOP	Aspect-Oriented Programmation
API	Application Programming Interface. Bibliothèque de fonctions
AT	Tâche Automatisée sous Windows
Axis	Framework permettant l'implémentation de WEB Services

B

Back Office	Système permettant la remontée des informations des systèmes d'encaissement en centrale
BIRT	Business Intelligence and Reporting Tools. Système de création de rapports pour les applications WEB
BPM	Business Process Management
Broker	Courtier. Intermédiaire entre serveurs et utilisateurs

C

C#	Langage de programmation orienté objet à typage fort, créé par la société Microsoft
C++	Langage de programmation orienté objet
CA	Chiffres d'affaires
CASS	Moteur de template pour CSS
CD	Compact Disk
CICS	Customer Information Control System. Serveur de transactions
CGI	Common Gateway Interface. Programme installé sur le serveur
CLUSTER	Ensemble de serveurs ou d'instances de serveurs d'applications
CMM	Capability Maturity Model. C'est un système qualité qui vise à améliorer le processus de développement logiciel.
COBOL	COmmon Business Oriented Language. Langage de programmation

COM	Components Objects Model. Mécanisme normalisé au niveau du système d'exploitation
Completion	Permet d'obtenir le complément d'un mot
CORBA	Common Object Request Broker Architecture Approche orientée objet dans les architectures distribuées
CORIG	COnception et Réalisation en Informatique de Gestion
CPU	Central Processing Unit. Microprocesseur
CRM	Customer Relationship Management
CRUD	Acronyme pour signifier les actions : Create, Read, Update, Delete sur une table
Crystal clear	Méthode Agile pour la gestion de projet
CSS	Cascading Style Sheets. Feuilles de style en cascade.
CVS	Current Version System. Permet d'une part le travail collaboratif entre plusieurs développeurs et, d'autre part la gestion de versions.

D

Datamart	Base de données dédiée à un sujet fonctionnel précis
DAO	Data Access Object. Couche d'accès aux données
DataWareHouse	Entrepôt de données
DB2	Base de données relationnelle
DCOM	Distribued Components Objects Model. Permet aux programmes de partager des données et d'offrir des services
Débogage	Action consistant à rechercher les erreurs d'un programme et de les corriger
DHTML	Dynamic HTML. Extension du langage HTML qui permet de rendre les pages beaucoup plus interactives
DIFF	Module de l'application merchandising permettant la diffusion des dossiers de préconisations PGC. Ce module repose sur les technologies J2EE, Apache et Tomcat
DIIOP	Protocole de communication utilisé dans CORBA
DNS	Domain Name Server. Système de noms de domaine
DOJO	Outil permettant de construire des interfaces DHTML
DOM	Document Object Model. Interface indépendante de la plateforme permettant d'accéder directement aux documents XML
DSDM	Dynamic Systems Development Method. C'est une méthode Agile de gestion de projet
DSI	Direction des Systèmes d'Information

E

EAI	Entreprise Application Integration
EAN	European Article Numbering. Code permettant d'identifier de façon précise et au niveau international un produit
EDI	Echange de Données Informatisées
ERP	Entreprise Ressource Planning
ESB	Entreprise Service Bus

ETL	Extract Transform Load

F

Facings	Nombre de produits que l'on peut exposer
Failover	Commutation automatique vers un système redondant ou en attente, lors de la survenue d'un échec ou la fin de tâche anormale d'un serveur, d'un système ou d'un réseau couramment actif
Farming	Déploiement automatique d'une application sur un ensemble de serveurs
FireFox	Navigateur OpenSource proposé par le projet Mozilla
Framework	Cadre d'application. Bibliothèque de classes fournissant une ossature générale pour le développement d'une application
FTP	File Transfer Protocole

G

GANTT	Tableau de répartition des ressources disponibles pour un projet dans le temps. Permet de planifier un projet.
GDP	Module de l'application merchandising permettant le suivi de l'élaboration des dossiers de préconisations. Il implémente une gestion de processus basée sur un workflow
GQL	Outil de Business Intelligence qui permet aux utilisateurs finaux d'extraire de l'information des systèmes sans faire de SQL. Nouvelle appellation BI-QUERY.
GOLD	Application permettant la gestion des articles locaux, les prix de ventes spécifiques, voire le contrôle des factures.

H

HA	High Availability. Haute disponibilité
HAML	Moteur de template pour HTML
Hardware	Equipement matériel
Hibernate	Framework permettant de faire du Mapping Relationel Objet
HMI	Human Machine Interface
HTML	Hypertext Transfert Markup Language. Langage servant à décrire des pages WEB et des documents hypertexte
HTTP	HyperText Transfert Protocol. Protocole de transfert des pages HTML sur le WEB
HTTPS	HyperText Transfert Protocol Secure Protocole de transfert de données sécurisées sur le WEB
Hub and Spoke	Hub : concentrateur. Spoke : rayon. Le concentrateur est le point de connexion central, les rayons sont autour. La notion est identique à celle du réseau en étoile
Hypertext	Se dit d'un document comportant des liens renvoyant au même document ou vers d'autres documents

I

IDE	Integrated Development Environment.
IHM	Interface Homme Machine
IP	Internet Protocole. Protocole réseau utilisé sur Internet
IP Spoofing	Technique consistant à utiliser une autre adresse IP que celle du poste
IRIS	Module applicatif du progiciel Smart Office de KLEE Commerce qui permet la gestion de la base des images
ISO	International Organization for Standardization

J

J2EE	Java 2 Enterprise Edition
JAAS	Java Authentification and Authorization Service
Java	Langage de programmation orienté objet, créé par Sun Microsystems et reposant sur une JVM
JavaBean	Composant logiciel destiné à être inclus aisément dans un programme Java
Javascript	Langage de programmation de type script, orienté objet
JCA	Java Connector Architecture
JCL	Job Control Language
JDBC	Java DataBase Connectivity
JMS	Java Message Service
JMX	Java Management eXtension
JNDI	Java Naming and Directory Interface. API permettant d'interfacer une application Java aux bases d'annuaire.
JSF	Java Server Faces. Permet de développer des interfaces graphiques améliorées. L'API sert à représenter des composants graphiques, à gérer des états et à supporter des événements
JSP	Java Server Pages
JTA	Java Transaction API
JTS	Java Transaction Services
JUnit	Outil permettant la réalisation de tests unitaires
JVM	Java Virtual Machine

K

KLEE	Société éditrice de la solution Smart Office permettant l'élaboration des planogrammes

L

LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol. Protocole Standard permettant de gérer des annuaires
Lighthttpd	Serveur WEB

M

Mainframe	Gros système
Manager	Module de l'application merchandising permettant une administration centrale des tables utilisées par l'application spécifique.
MCD	Modèle Conceptuel de Données
MDM	Master Data Management
Merchadmin	Module de l'application merchandising permettant la gestion de la diffusion PGC et du paramétrage du progiciel
MERISE	Méthode de génie logicielle pour la construction des systèmes d'information
Mini-réassort	Colisage minimal avec lequel peut être livré un magasin
MOA	Maîtrise d'OuvrAge. Personnes ou entité. Doit prendre en charge les activités suivantes : définition de l'expression des besoins, des fonctionnalités à réaliser en priorité et également de la recette
Module	Caractéristique d'un produit qui détermine à quel assortiment il appartient
MOE	Maîtrise d'OEuvre. Personnes ou entité répondant à la maîtrise d'ouvrage. Elle définit et conçoit la solution, la réalise et assiste la maîtrise d'ouvrage
MOM	Message Oriented Middleware. Permet aux applications d'échanger des messages avec d'autres applications
Mongrel	Serveur d'application Ruby
MPD	Modèle Physique de Données
MultiThread	Thread : ensemble de suites d'instructions soumis à la CPU. Le MultiThread correspond à l'exécution de plusieurs threads afin d'exécuter plusieurs tâches en même temps
MVC	Modèle d'architecture permettant la séparation des préoccupations « métier », des contrôles et de l'interface de l'utilisateur
MVS	Multiple Virtual Storage. Système d'exploitation
MySQL	Base de données relationnelle Open Source.

N

NIS	Network Information System

O

ODBC	Open DataBase Connectivity. Permet à une application sous Windows d'accéder à une base de données, indépendamment du type de cette dernière
ODS	Operational DataStore
OMEGA	Application permettant de gérer les ventes
OMG	Object Management Group. Consortium de l'industrie informatique ayant pour objectif la promotion de la programmation orientée objet
ORACLE	Base de données relationnelle propriétaire, fournie par Oracle Corporation
ORM	Objet Relational Mapping

OSI	Open Systems Interconnection. Une norme internationale développée par l'ISO pour décrire un modèle générique de l'interconnexion de systèmes hétérogènes

P

PACBASE	Provient de PAC, Programmation Automatique Corig. Méthode de programmation structurée
Pattern	Solution récurrente décrivant et résolvant un problème général dans un contexte particulier
PDA	Personal Digital Assistant. Assistant personnel numérique.
PDCA	Plan, Do, Check, Act. Méthode qualité. Roue de Deming
PDF	Portable Document Format. Document pouvant être visualisé dans son exacte mise en page quel que soit l'ordinateur sur lequel on le consulte
PERL	Langage de programmation.
PERT	Program Evalution and Review Technic. Méthode de représentation de l'enchaînement entre les différentes sous unités d'un projet. Elle permet de détecter les possibilités de parallélisme dans l'accomplissement des tâches.
PGC	Produit de Grande Consommation
POA	Programmation Orientée Aspect
POC	Proof Of Concept. Test aidant à la compréhension plus précise et plus en profondeur de l'utilisation d'un outil afin de fournir une solution plus précise
POJO	Plain Old Java Objet. Classe Java classique
Python	Langage de programmation interpreté, orienté objet

Q

…

R

RAD	Rapid Application Design and Development. Méthodologie Siebel. Technique de développement accéléré d'applications basée sur l'utilisation extensive des interfaces graphiques
RAID	Redundant Array of Independent Disks. Types de contrôleur de disque dur (0 à 5). Permet une tolérance aux pannes.
Rails	Framework permettant la mise en place d'applications structurées.
RAM	Random Access Memory. Mémoire à accès aléatoire
RCOV	Outil de couverture de code pour Ruby
RDBMS	Relational DataBase Management System
Refactoring	Opération consistant à retravailler le code source pour améliorer sa lisibilité et simplifier sa maintenance
REPORT	Module de l'application merchandising permettant de proposer des rapports
REST	REpresentational State Transfer. Architecture orientée ressource
REXEC	Services permettant de déclencher des exécutables à distance à partir d'un autre poste

RL	RL ou RLE signifie Relevé Linéaire ou Relevé Linéaire Etendu. C'est un module de l'application merchandising permettant de faire des relevés linéaires en magasin
RLAdmin	Module applicatif de l'application merchandising permettant de faire l'administration et le suivi du module RL
RMI	Remote Method Invocation. Interface permettant aux objets Java situés sur différentes machines de communiquer ensemble
ROI	Return On Investment. Retour sur investissement
RoR	Ruby on Rails. Framework libre écrit en ruby
RPC	Remote Procedure Call
Ruby	Langage orienté objet interpreté
RUP	Rational Unified Process. Méthologie de développement logiciel orienté objet de la société Rational

S

SADT	Structure Analysis and Design Technique. Méthode de conception par spécification graphique servant à la conduite de projet
SCAFFOLD	Echafaudage permettant de la réutilisation de structure
Scalability	Adaptabilité d'un système. Capacité d'un système à évoluer en puissance
SCM	Supply Chain Management
Scoring	Classement
Selenium	Outil permettant d'effectuer des tests fonctionnels basés sur la capture de scénario et le rejeux
Sfoffice	Module application du progiciel Smart Office de la société Klee Commerce permettant de gérer la base de données utilisée par l'application SMART
Scrum	Méthode Agile pour la gestion de projet informatique
SI	Système d'Information
SMART	Outil permettant l'élaboration de planogrammes
SMARTBatch	Outil permettant d'extraire des informations et des images du référentiel utilisé par SMART afin de bâtir un site WEB statique
SNMP	Simple Network Management Protocol. Protocole de gestion de réseau TCP/IP
SOA	Service Oriented Architecture. Architecture orientée Service
SOAP	Simple Objet Access Protocol. Permet l'invocation de composant logiciel à travers Internet. L'interopérabilité quelle que soit la plateforme
SpoF	Single point of Failure
SQL	Standard Query Language
Struts	Framework permettant de mettre en place des applications structurées suivant MVC
SYGMA	Application permettant de gérer le référentiel des produits et leurs nomenclatures

T

Tapestry	Framework permettant de créer des applications WEB divisées en des ensembles de pages, chacune construite à partir de composants
TCP	Transfert Control Protocol. Protocole de couche de transport
Teradata	Base de données relationnelle pouvant atteindre un tera de données.
TSE	Service Windows permettant l'accès à des bureaux distants, à partir de son poste
TSO	Time Sharing Option. Interface de temps partagé interactif

U

UB	Unité de besoin. Elle est rattachée à une UG et se compose d'un ensemble de produits
UDDI	Universal Decription Discovery and Integration
UG	Unité de Gestion. Elle est rattachée à un rayon et se compose d'un ensemble d'UB
UML	Unified Modeling Language. Langage de modélisation pour la programmation orientée objet
UNIX	Système d'exploitation
UP	Unified Process. Processus Unifié
URL	Uniform Ressource Locator. Adresse codifiée d'une ressource de l'Internet
UTIL	Module de l'application merchandising permettant de supprimer ou mettre à jour une base de données de façon ensembliste

V

V	Cycle de vie logiciel ayant la forme d'un V
VBIS	Vignette Business Integration Studio.
VISU	Module de l'application merchandising permettant la diffusion des dossiers de préconisations du textile aux magasins
VISUadmin	Module de l'application merchandising permettant la gestion de VISU
VU	Virtual User. Utilisateur virtuel

W

WAP	Wireless Application Protocol. Version allégée d'HTTP pour permettre l'affichage des pages WML
WAS	Websphere Application Server
WAR	Web ARchive. Fichier contenant une application WEB (module et ressources JSP/Servlet, HTML)
WEB	Dans l'Internet, système réparti géographiquement et structurellement, de publication et de consultation de documents faisant appel aux techniques de l'hypertexte
WebRick	Serveur d'application Ruby

WML	Wireless Markup Language. Une version simplifiée d'HTML pour le protocole WAP
WSDL	Web Service Description Langage

X

XML	eXtensible Markup Language. Facilite la définition, la validation et le partage de différents formats de document sur le WEB
XMLHttpRequest	Objet Javascript permettant d'obtenir des données au format XML à l'aide de requêtes http
XP	Méthode Agile qui pousse un certain nombre de principes à l'extrême afin de gérer un projet informatique
XSL	eXtensible Styleshet Language. Langage pour définir des feuilles de style
XSLT	XSL Transformation. Language dédié à la transformation de données XML, faisant partie de XSL

Y

YML	Type de fichier géré à l'aide d'indentation

Z

…

.

.NET	Framework dans lequel les applications Windows peuvent être développées et exécutées. Il constitue la réponse de Microsoft à la plateforme Java

Bibliographie

Ouvrages

[BAU05] Christian BAUER , Gavin KING. - *Hibernate* - Campus Press : 2005. 432p.

[BEN05] Jean-Louis BENARD, Laurent BOSSAVIT, Régis MEDINA, Dominic Williams. – *Gestion de Projet eXtreme Programming* - Eyrolles : 2005. 300p.

[CHO04] Vivek CHOPRA, Amit BAKORE, Jon EAVES, Ben GALBRAITH, Sing LI, Chanoch WIGGERS. – *Professional Apache Tomcat* - WROX : 2004. 237-270 457-498.

[CRA06] Dave CRANE, Eric PASCALLERO, Darien JAMES. - *Ajax en pratique* - Campus Press : 2006. 612p.

[DUB06] Julien DUBOIS, Jean-philippe RETAILLE, Thierry TEMPLIER. – *Spring par la pratique* - Eyrolles : 2006. 517p.

[FEL06] Jean charle FELECITE. - Développement JAVA sous Struts (version 1.2) - ENI editions : 2006. 415p.

[FRE05] Eric FREEMAN, Elisabeth FREEMAN. - *Design Patterns* - O'Reilly : 2005. 639p.

[MAN06] Pascal MANGOLD. – *Gestion de Projet Informatique Compact* - Eyrolles : 2006. 122p.

[PAT05] Anthony PATRICIO. - *Hibernate 3.0* - Eyrolles : 2005. 317p.

[ROQ00] Pascal ROQUES, Franck VALEE. - *UML en action* - Eyrolles : 2000. 402p.

[ROQ07] Pascal ROQUES, Franck VALEE. - *UML 2 en action* - Eyrolles : 2007. 384p.

[SAU06] Eric SAURION. – *Ruby on Rails* – O'Reilly : 2006. 579p.

Articles

[ARN05] Stéphane ARNAULT *« Hibernate - Persistance objet – relationnel »* décembre 2005
http://www.labo-sun.com/resource-fr-essentiels-840-0-java-persistance-hibernate-persistance-objet-relationnel.htm

[BOR04] Xavier BORDERIE *« Passer à UML 2»* Septembre 2004
http://www.journaldunet.com/developpeur/tutoriel/cpt/040915-passer-a-uml2.shtml

[JOU05a] Cyril JOUI *« Architecture J2EE - Comment organiser son application J2EE »* décembre 2005
http://www.labo-sun.com/resource-fr-essentiels-833-0-java-j2ee-architecture-j2ee-comment-organiser-son-application-j2ee.htm

[JOU05b] Cyril JOUI *« Introduction J2EE – Applications d'entreprise »* décembre 2005
http://www.labo-sun.com/resource-fr-essentiels-837-0-java-j2ee-architecture-j2ee-introduction-j2ee-applications-d-entreprise.htm

[LAB00] Corinne LABORDE, Sébastien MATTER *«Capability Maturity Model»* Mai 2000
http://www.polytech.unice.fr/~hugues/GL/CMM/cmm.html

[MAR04] Antony MARCANO *«OpenSTA, the free performance testing tool, versus the big-guns... »* juillet 2004
http://www.testingreflections.com/node/view/361

[PEN04a] Srini PENCHIKALA « Clustering and Load Balancing in Tomcat 5, Part 1» mars 2004
http://www.onjava.com/pub/a/onjava/2004/03/31/clustering.html

[PEN04b] Srini PENCHIKALA « Clustering and Load Balancing in Tomcat 5, Part 2» avril 2004
http://www.onjava.com/pub/a/onjava/2004/04/14/clustering.html

[PEN04c] Srini PENCHIKALA « *Session Replication in Tomcat 5 Clusters, Part 1* » novembre 2004
http://www.onjava.com/pub/a/onjava/2004/11/24/replication1.html

[PEN04d] Srini PENCHIKALA *«Session Replication in Tomcat 5 Clusters, Part 2»* décembre 2004
http://www.onjava.com/pub/a/onjava/2004/12/15/replication2.html

[REN05] Jean-baptiste RENAUX *« Ant - L'automatisation des tâches du programmeur »* octobre 2005
http://www.labo-sun.com/resource-fr-essentiels-711-0-java-xml-ant-l-automatisation-des-taches-du-programmeur.htm

[VIV05] Vivek VISWANATHAN *«Load Balancing Web Applications»* septembre 2001
http://www.onjava.com/pub/a/onjava/2001/09/26/load.html

Ressources sur le Web

Ressources Apache Tomcat:
http://tomcat.apache.org/

Ressources BIRT:
http://www.eclipsetotale.com/articles/BIRT.html

Ressources Eclipse:
http://www.eclipse.org/resources/

Ressources Hibernate:
http://www.hibernate.org/

Ressources Opensta:
http://www.opensta.org/
Ressources Selenium:
http://www.openqa.org/

Ressources Struts:
http://struts.apache.org/

Ressources wikipedia:
http://fr.wikipedia.org/wiki/Cycle_de_d%C3%A9veloppement
http://fr.wikipedia.org/wiki/M%C3%A9thode_agile
http://fr.wikipedia.org/wiki/Unified_Process#PU.2C_m.C3.A9thode_agile_.3F
http://fr.wikipedia.org/wiki/Refactorisation

Divers

Le site de définition :
http//www.alaide.com

Les guides utilisateurs de :

- VBIS de la société Vignette
- GENIO de la société Hummingbird
- POWERCENTER de la société Informatica
- WEBLOAD de la société Radview

Le cours GLG203 « Architectures Logicielles Java » dont le responsable est Louis DEWEZ, en particulier l'introduction.
http//iagl.free.fr/glg203/tp/Introduction.pdf

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

PARIS

MËMOIRE

Présenté en vue d'obtenir le

DIPLÔME D'INGÈNIEUR C.N.A.M

en

INFORMATIQUE

par

Laurent DONGÈ

Mise en œuvre d'une application de merchandising

Annexes

Soutenu le jeudi 6 décembre 2007

Jury

Présidente : Isabelle COMYN-WATTIAU, Professeur

Membres : Tatiana AUBONNET, Maître de conférence

Jacky AKOKA, Professeur

Pierre AUDOIN, Chef de service

Pierre ESTIVALET, Merchandiser Senior

Annexe A : Exemples de développement ETL GENIO

Voici une illustration de l'utilisation de GENIO au travers d'un exemple.

Cet exemple va se baser sur l'extraction de la liste des codes EAN (code sur 13 caractères identifiant les produits de façon unique) présents dans le référentiel afin de recevoir les visuels correspondant aux nouveaux produits.

Une fois par mois, la liste complète des EAN caractérisant les produits ainsi que leurs libellés sont récupérés à partir du référentiel Oracle et écrits dans un fichier.

Ce fichier est ensuite envoyé par email au prestataire qui en retour nous fait parvenir un CDROM contenant les visuels des nouveaux produits.

Ce traitement est décrit dans le point Extraction de la liste des codes EAN pour envoi des images de la partie 2.5 Mise en œuvre des traitements.

Tous les objets manipulés par GENIO sont regroupés en projet.

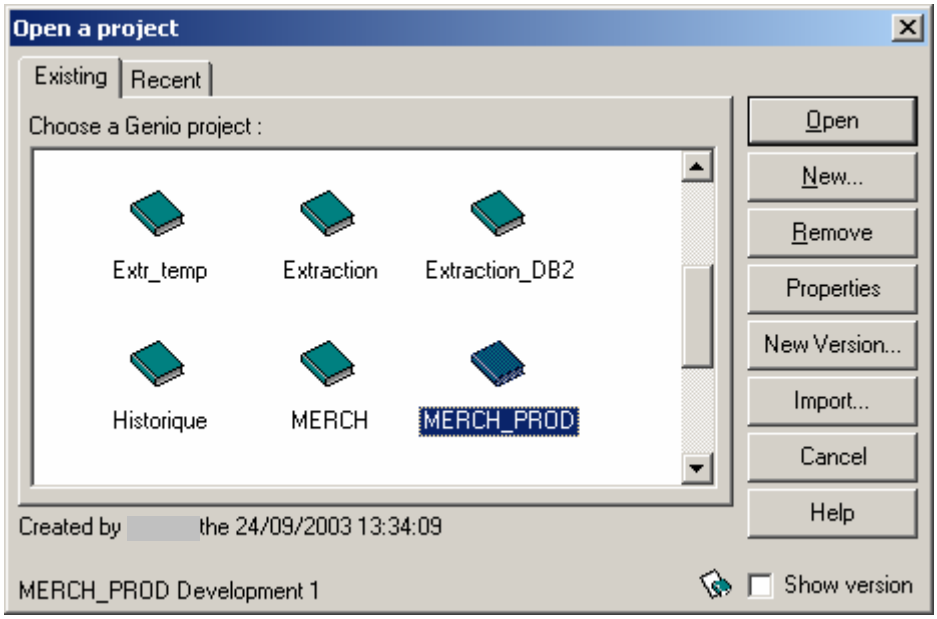


Figure 163 : Ouverture du projet GENIO

Nous définissons une connexion vers l'instance Oracle qui héberge le référentiel KLEE commerce. Pour cela, il faut définir une source de données ODBC au niveau du serveur. L'instance Oracle est ici identifiée par le nom SFMONOPRIX.

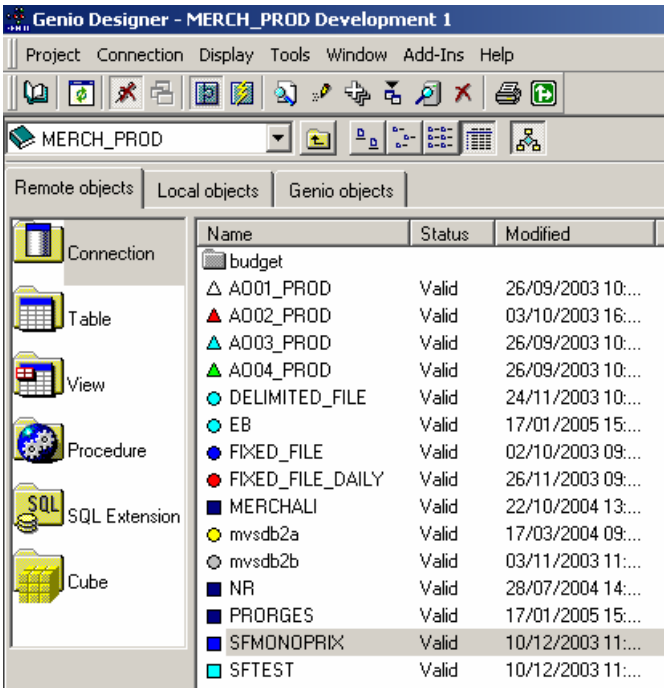


Figure 164 : GENIO Designer - liste des connexions existantes

Une fois la connexion créée, nous créons une table GENIO à partir d'une table ORACLE. La table GENIO nommée PRODUCT est le reflet de la table ORACLE PRODUCT appartenant au schéma SFMONOPRIX.

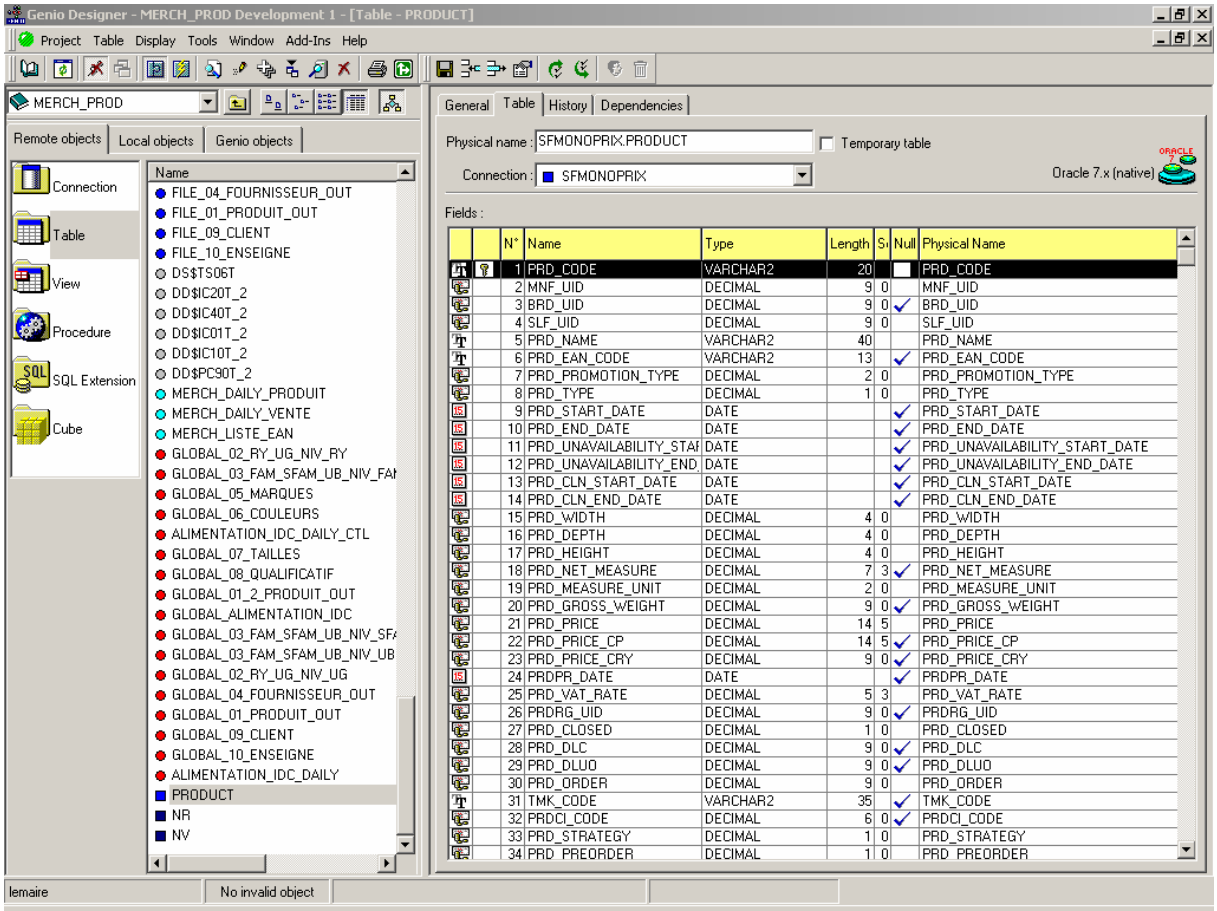


Figure 165 : GENIO designer - structure de la table produit

Le fichier de sortie LISTE_EAN.txt se présente également sous la forme d'une table.

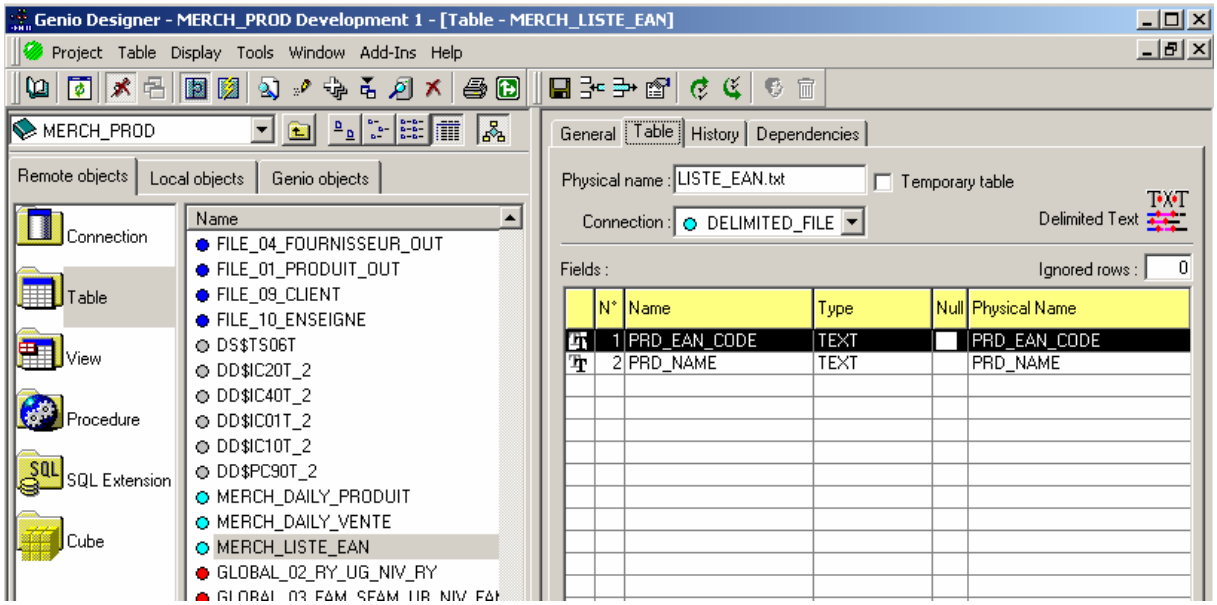


Figure 166 : GENIO designer - structure de la table associée au fichier de sortie

Il est associé à une connexion DELEMITED_FILE qui correspond aux fichiers dont les colonnes sont délimitées par un séparateur.

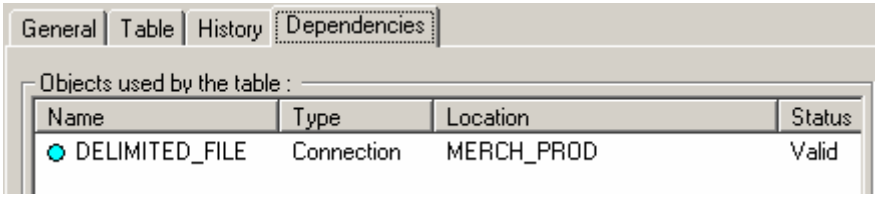


Figure 167 : GENIO designer - connexion associée au fichier de sortie.

Le processus comporte deux modules.

Un premier module qui recopie les enregistrements de la table PRODUCT dans une table de référence MERCH_REF_LISTE_EAN appartenant au Datamart du merchandising *(Figure 168).*

Les champs PRD_CODE_EAN et PRD_NAME sont mis en correspondance (mapping) avec les champs de la table cible *(Figure 169).*

Préalablement, nous supprimons l'ensemble des enregistrements présents dans la table cible.

Le deuxième module supprime le fichier de données en sortie et recopie la table de référence dans le fichier *(Figure 170).*

Les sources de données en entrée et en sortie du module sont indiquées dans l'onglet Data *(Figure 171).*

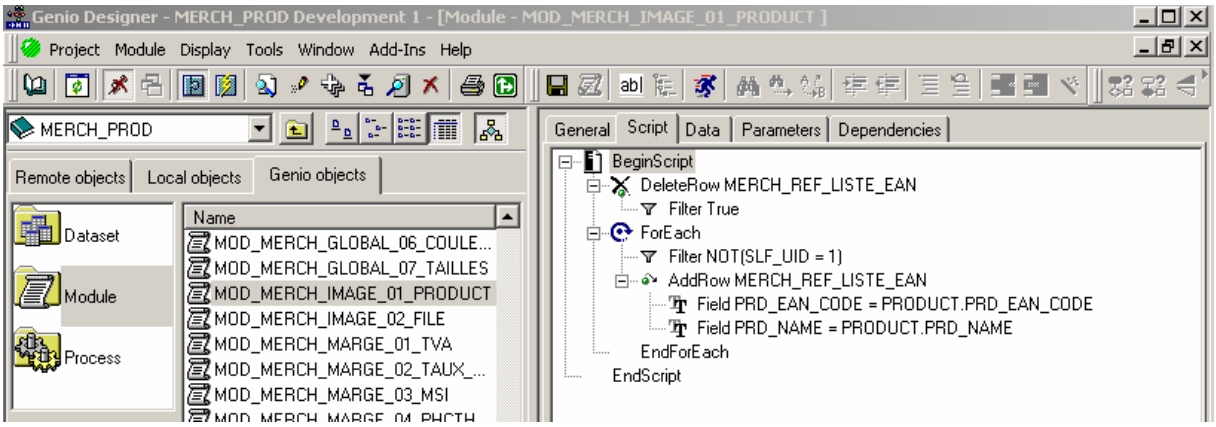


Figure 168: GENIO designer - premier module

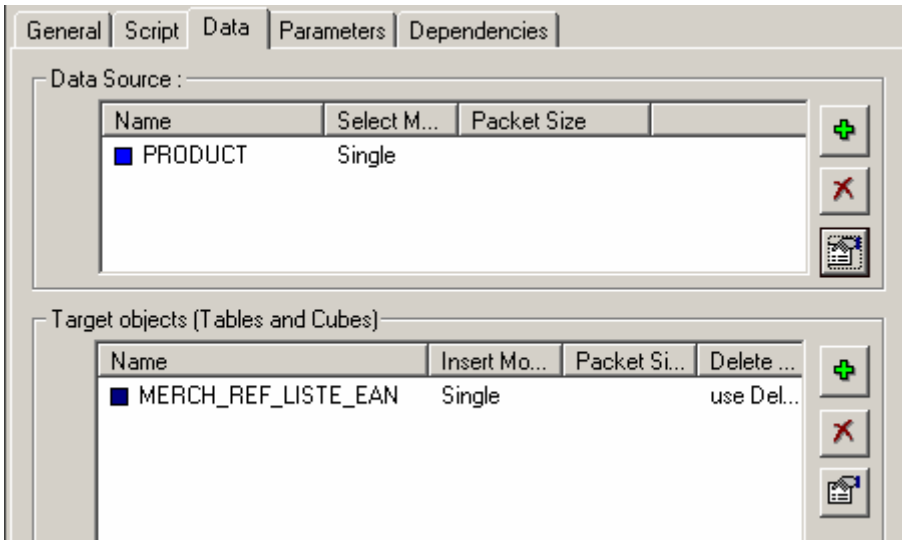


Figure 169 : GENIO designer - sources et cibles utilisées par le premier module

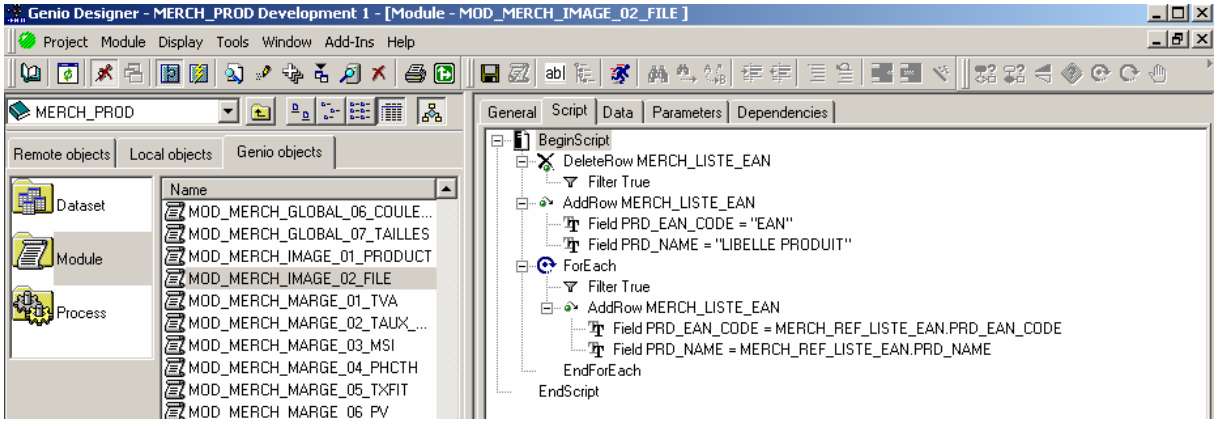


Figure 170 : GENIO designer - deuxième module

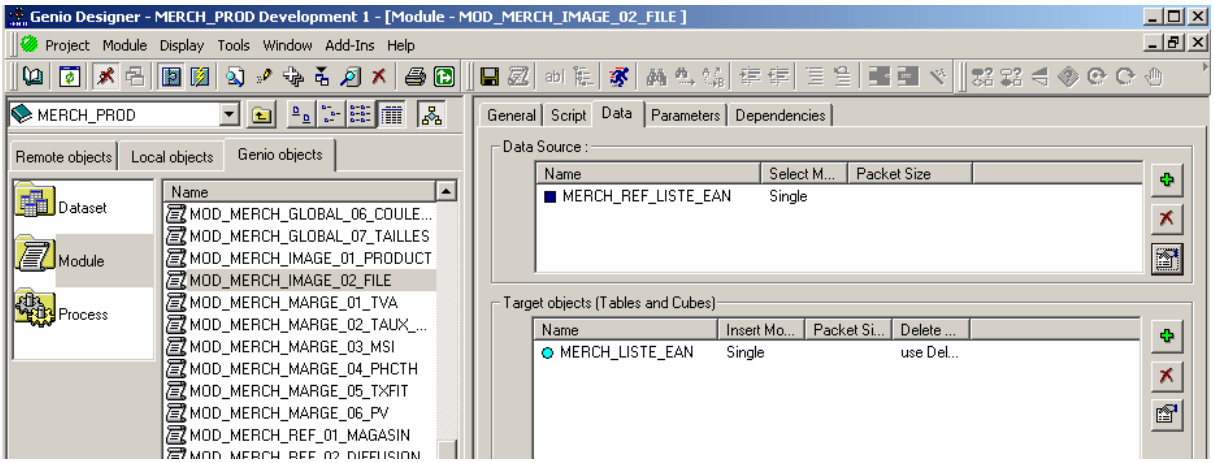


Figure 171 : GENIO designer - sources de données utilisées par le deuxième module

Il est possible de passer d'une représentation graphique à du scripting. La représentation graphique du deuxième module donne le script suivant :

```
BeginScript
	DeleteRow MERCH_LISTE_EAN
		Filter True
	AddRow MERCH_LISTE_EAN
		Field PRD_EAN_CODE = "EAN"
		Field PRD_NAME = "LIBELLE PRODUIT"
	ForEach
		Filter True
		AddRow MERCH_LISTE_EAN
			Field PRD_EAN_CODE = MERCH_REF_LISTE_EAN.PRD_EAN_CODE
			Field PRD_NAME = MERCH_REF_LISTE_EAN.PRD_NAME
	EndForEach
EndScript
```

Le processus appelle les deux modules précédents puis envoie un email aux personnes en charge de la préparation des visuels des nouveaux produits.

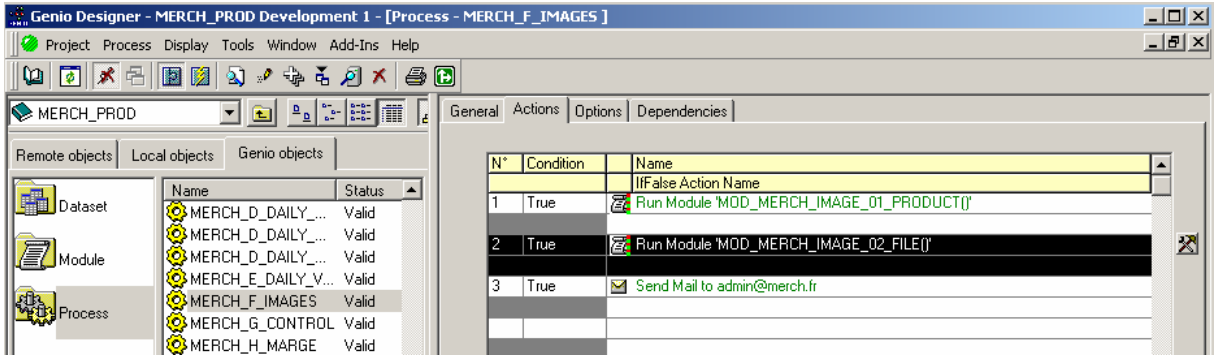


Figure 172 : GENIO designer - détail du processus

Comme nous pouvons le voir ci-dessous *(Figure 173)*, Send Mail permet de renseigner les champs habituellement présents dans un email. Il est également possible d'associer le fichier « texte » en pièce jointe. Il suffit de spécifier FILE dans la colonne Information type et de renseigner le chemin et le nom du fichier que l'on souhaite joindre dans la description.

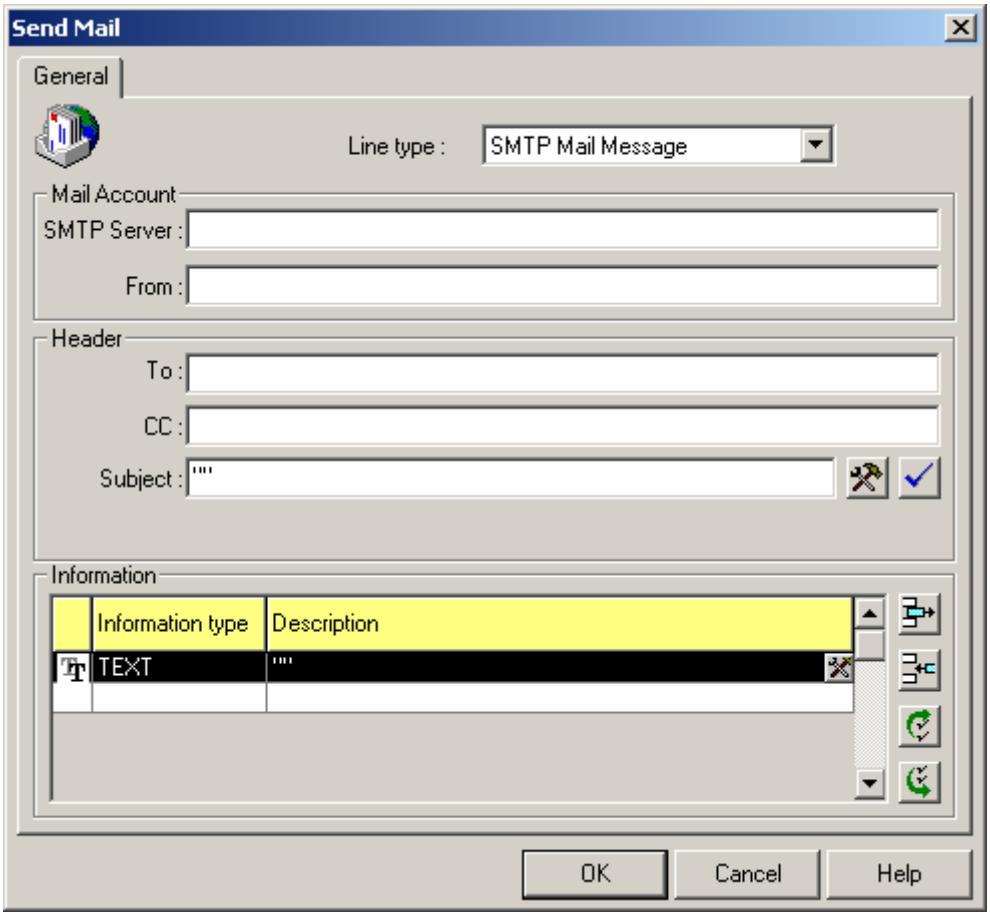


Figure 173 : GENIO designer - composant d'envoi des emails

Il est possible de conditionner l'exécution d'un module en spécifiant @Error = 0 au niveau de la condition associée au module.

Il est alors nécessaire d'incorporer les lignes ci dessous dans le module précédent :

```
If @Error = 0 and @DbError <> 0 Then
        Let @Error = @DbError
Else
EndIf
```

Une fois le processus élaboré, il est possible de planifier son exécution dans le scheduler de GENIO.

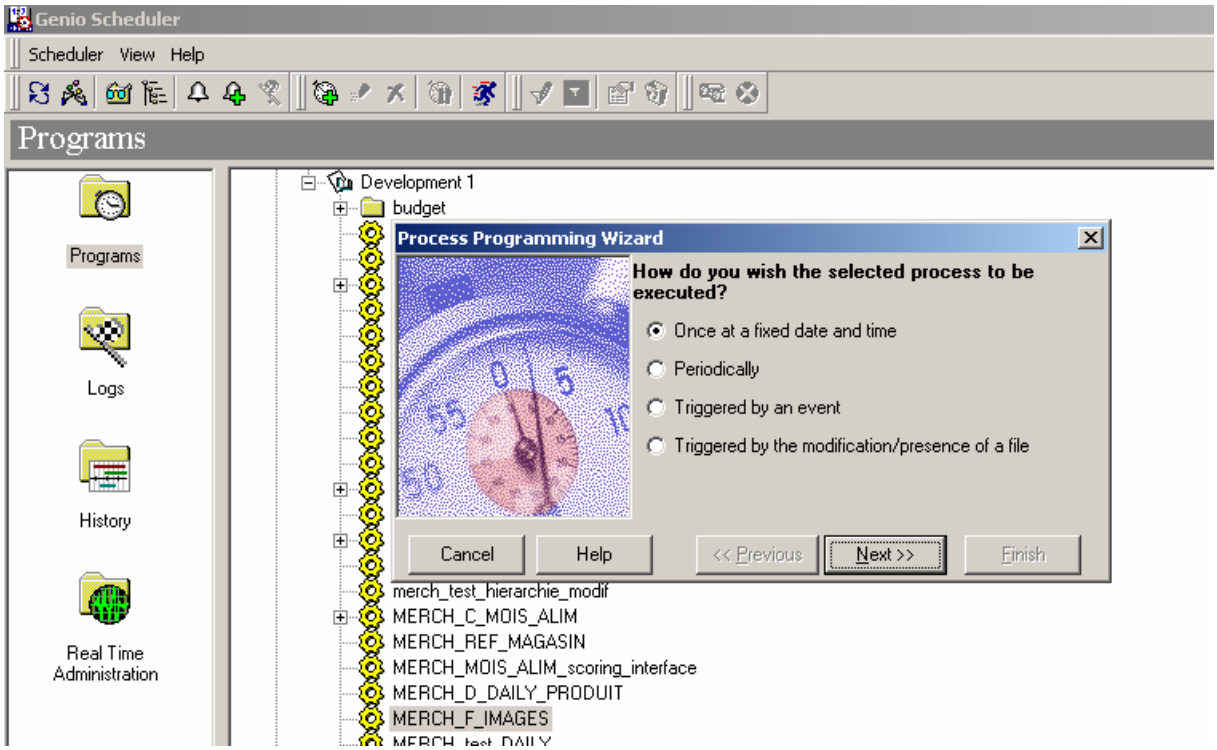


Figure 174 : GENIO scheduler - planification de l'exécution d'un processus

Les différentes programmations apparaissent sous le processus lorsque nous nous positionnons sur le choix Programs.

Figure 175 : GENIO scheduler - programmation d'une exécution par événement

Nous pouvons déclencher un événement à partir d'un module, comme dans l'exemple ci-dessous.

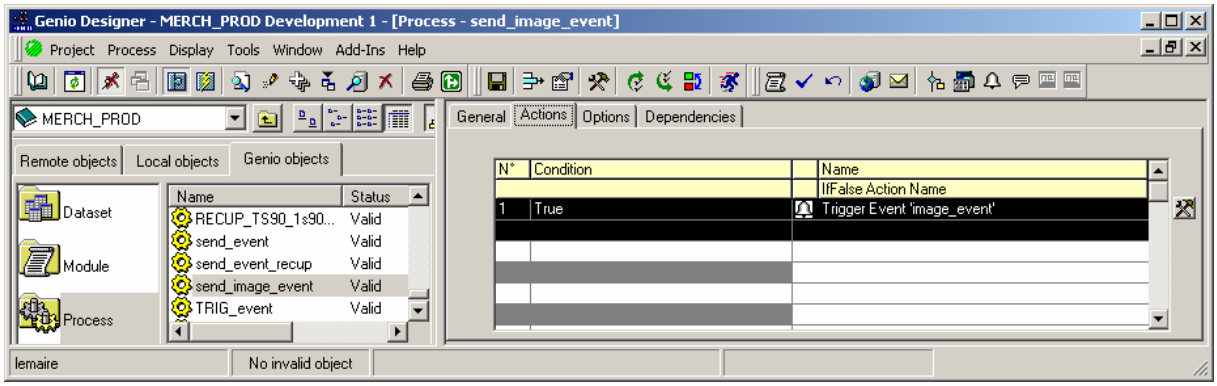


Figure 176 : GENIO designer - déclenchement d'un événement à partir d'un module

Dans le scheduler, nous pouvons voir en temps réel l'exécution d'un processus.

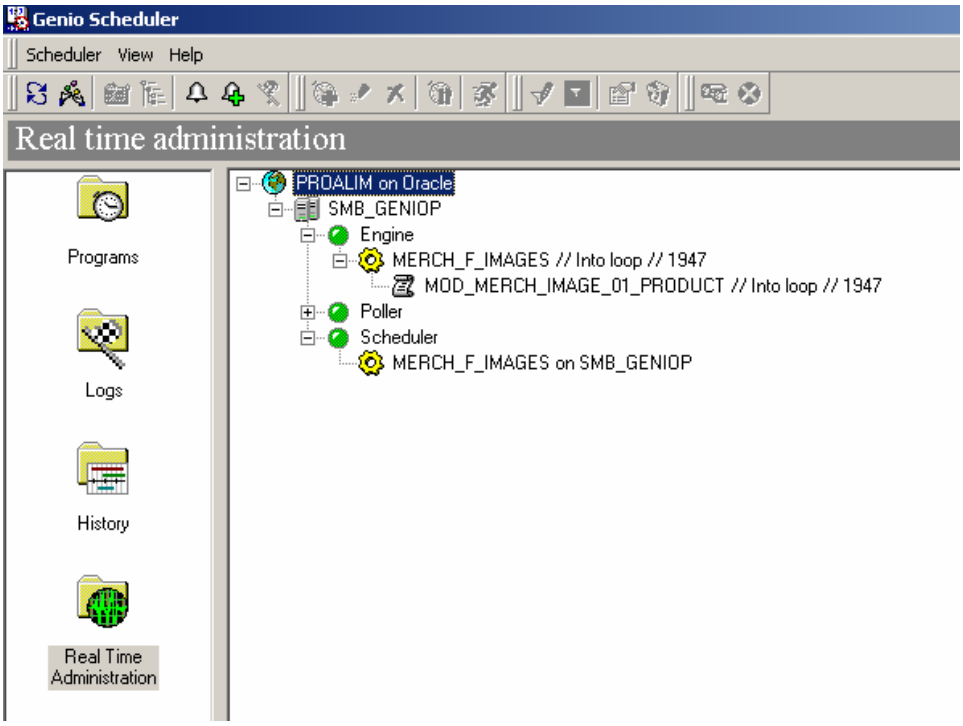


Figure 177 : GENIO scheduler - visualisation d'une exécution en temps réel

En double cliquant sur le nom du processus nous pouvons visualiser le statut courant du processus.

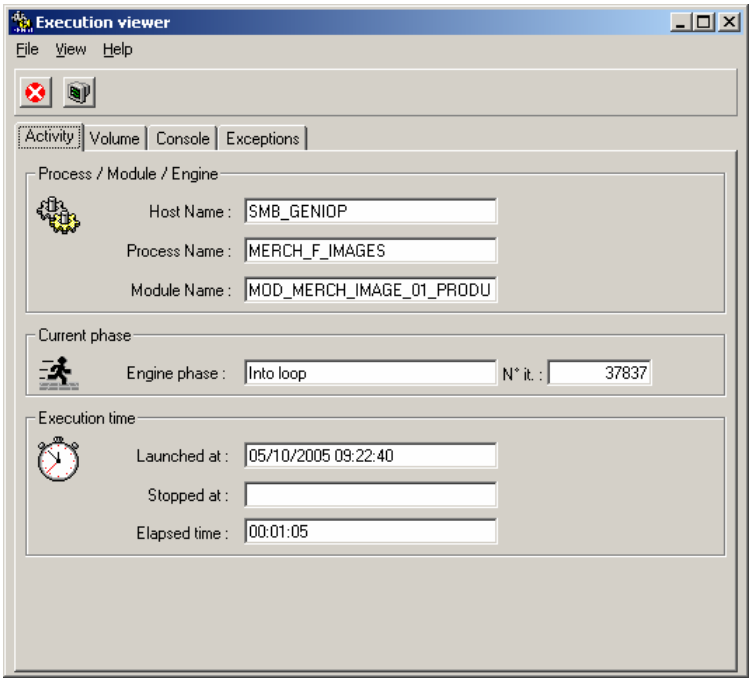


Figure 178 : GENIO scheduler - statut courant du processus

Une fois le processus terminé, nous pouvons obtenir des informations sur les volumes traités par le processus.

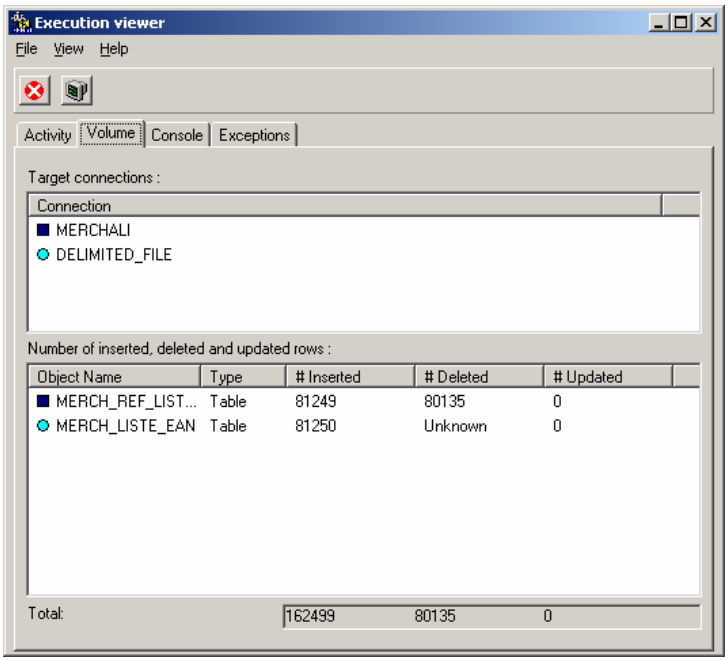


Figure 179 : GENIO scheduler - visualisation des volumes

En cours d'exécution, nous pouvons visualiser les sorties sur la console. Ces sorties correspondent aux autres SQL utilisés par le module, ainsi que les messages placés par le développeur dans les modules à l'aide de la commande Write.

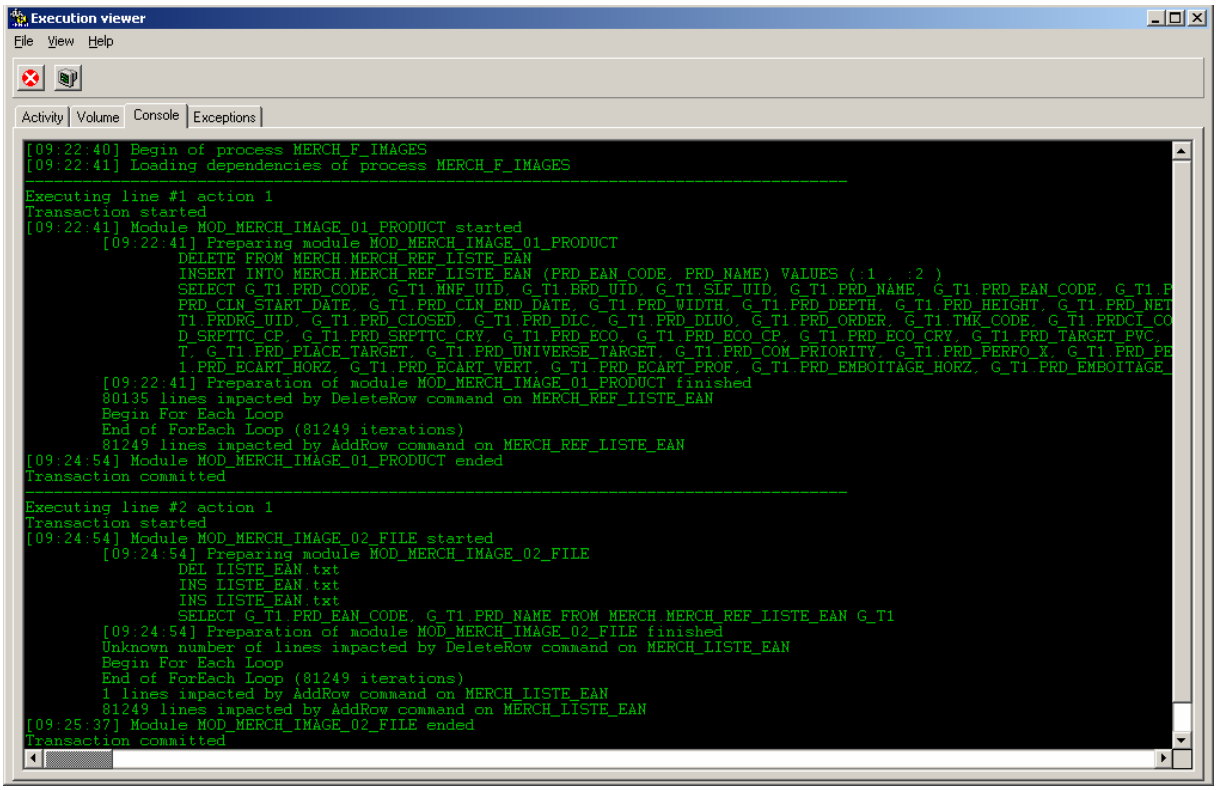


Figure 180 : GENIO scheduler - visualisation de la console

A l'aide du choix Logs, nous avons accès à l'intégralité des journaux d'exécution correspondant à un processus donné.

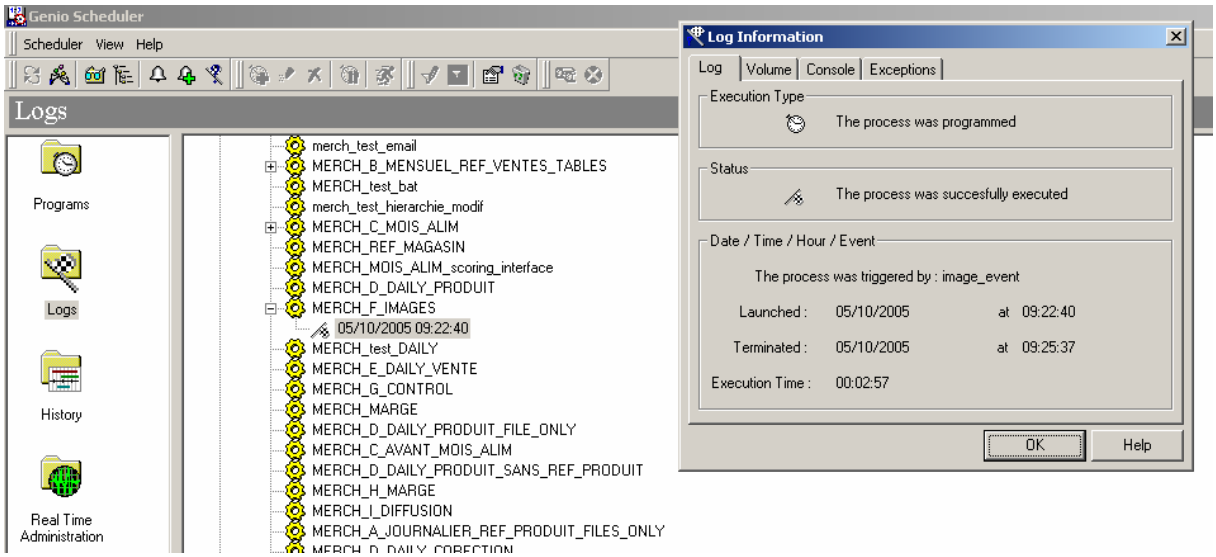


Figure 180 : GENIO scheduler - Consultation des logs d'exécution

Quant au choix History, il permet de consulter les journaux d'exécution triés par date. Il indique également la liste des processus planifiés.

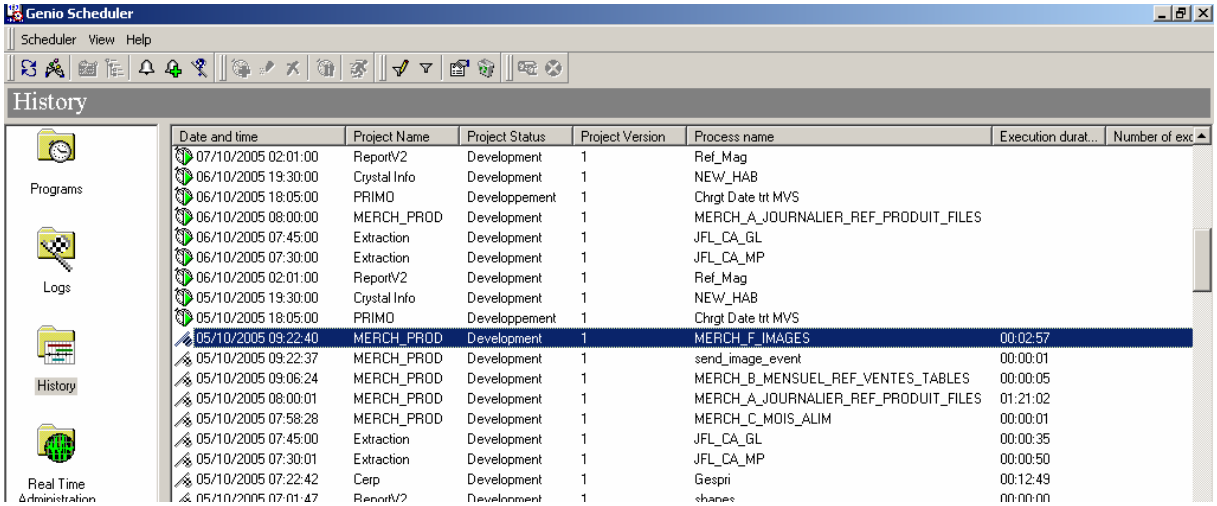


Figure 181 : GENIO Scheduler - consultation de l'historique des exécutions

Annexe B : Exemple de développement VBIS

Ci-dessous une illustration de l'utilisation de VBIS au travers d'un exemple.

Pour un certain nombre de raisons, nous désirons gérer les aspects « métier » relatifs à l'entreprise en dehors du progiciel d'élaboration des dossiers de préconisations.

Cet exemple va se baser sur l'extraction des informations relatives à la gestion des univers du référentiel Klee Commerce et à leur chargement dans le Datamart.

Ce traitement est décrit dans le sous point a) Exportation des données dans le datamart du point Application spécifique de la partie 2.5 Mise en œuvre des traitements.

Klee a mis en place des procédures fournissant des fichiers « texte ». Ces fichiers « texte » doivent ensuite être exploités pour alimenter le datamart merchandising.

A l'aide de VBIS, nous pouvons

- lancer l'exécution des procédures de KLEE
- gérer leurs codes retour
- chercher des erreurs dans les fichiers de log créés
- vérifier l'existence des fichiers d'export
- mettre à jour une table pour indiquer le statut de l'exécution
- enchaîner un traitement qui permet d'utiliser les fichiers « texte » pour charger la base ORACLE du datamart

C'est le dernier point que nous allons détailler pour illustrer l'utilisation de VBIS.

Un projet se compose d'un ensemble de diagrammes de flux comportant des adaptateurs.

L'ensemble de ces objets et leurs utilisations sont visualisables dans l'onglet Where Used ci-dessous (*Figure 183*).

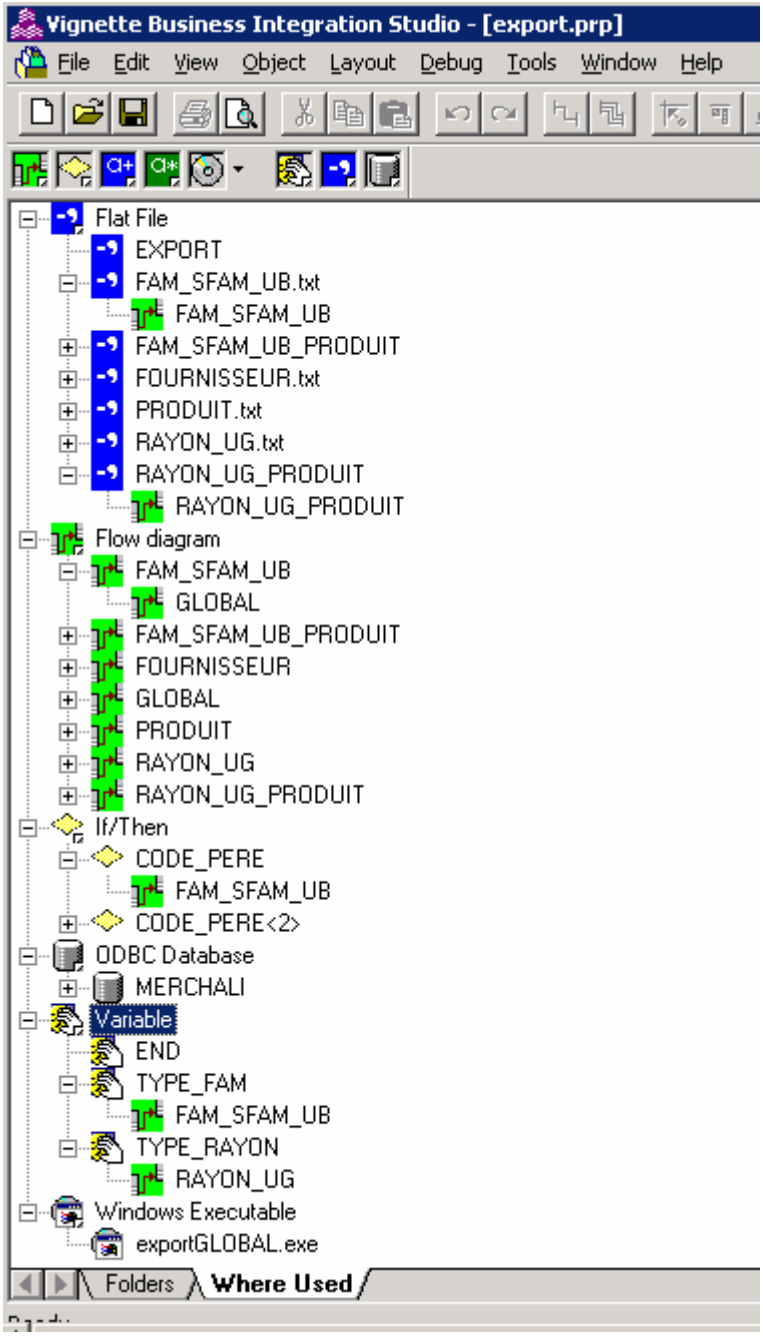


Figure 183 : Liste des objets utilisés dans le projets et dépendances

Un diagramme principal nommé GLOBAL a été créé et appelle un ensemble d'autres diagrammes de flux dont l'exécution a été conditionnée par la réussite du précédent.

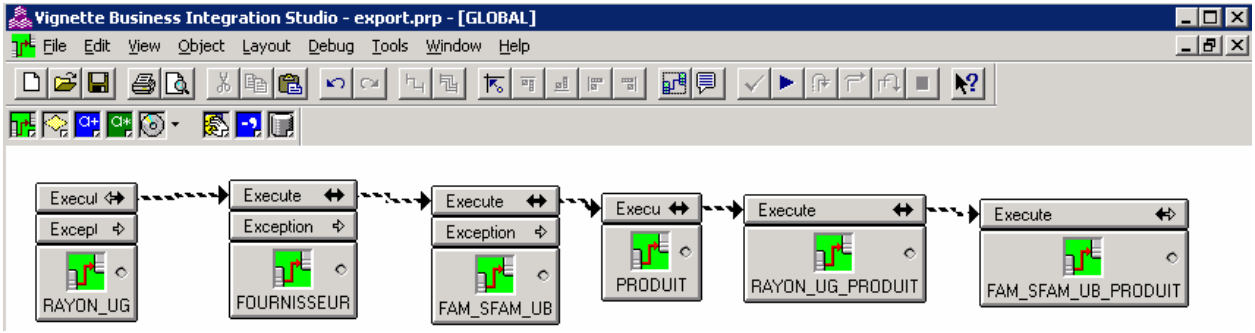


Figure 184 : Diagramme de flux principal du projet

Le diagramme de flux RAYON_UG permet de récupérer la hiérarchie des rayons et des unités de gestion correspondant aux produits.

Le diagramme de flux RAYON_UG se compose des adaptateurs suivants *(Figure 184)* :

- Le premier adaptateur permet de supprimer les enregistrements de la table MA_CATALOGUE.
- Le deuxième adaptateur permet de lire le fichier « texte » RAYON_UD.txt
- Un autre adaptateur permet de supprimer les espaces dans le port catalogue_code.
- Le catalogue_code est construit à partir d'une expression du port catalogue_code.
- Le catalogue type est alimenté à partir d'une variable.
- Enfin, la cible MA_CATALOGUE, qui est une table ORACLE, est alimentée en Batch.

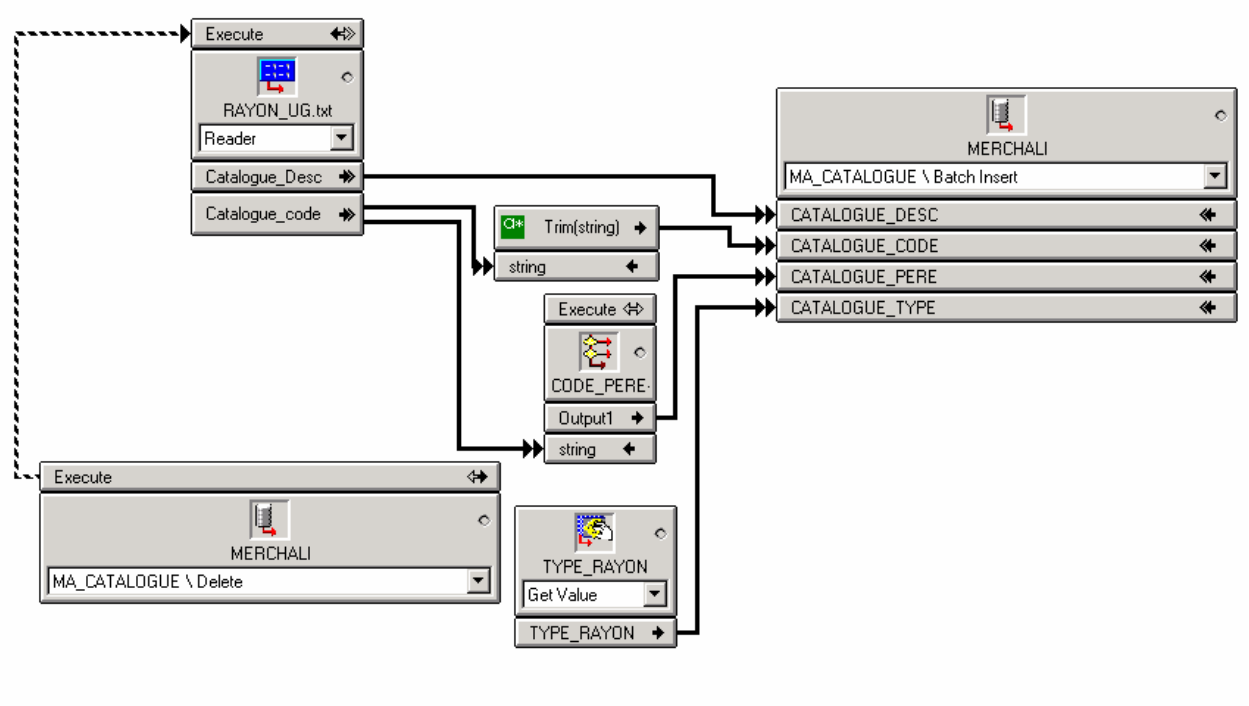


Figure 185 : Adaptateurs composant le diagramme de flux

Ci-dessous, les propriétés de l'adaptateur permettant de supprimer les espaces en début et en fin de valeur d'un port. Ici c'est la fonction Trim qui est utilisée.

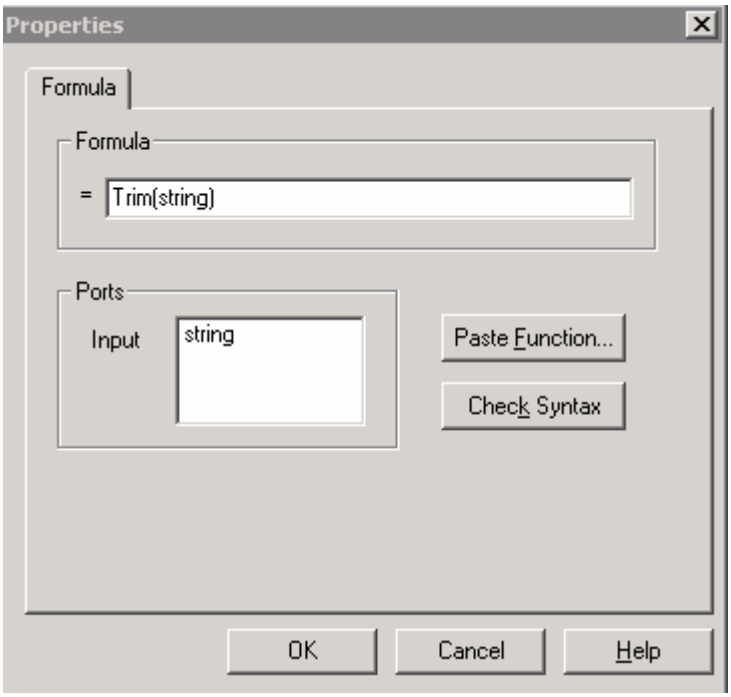


Figure 186 : Adaptateur Formula utilisant une fonction et un port d'entrée

Mais de nombreuses autres fonctions sont également disponibles *(Figure 187)*.

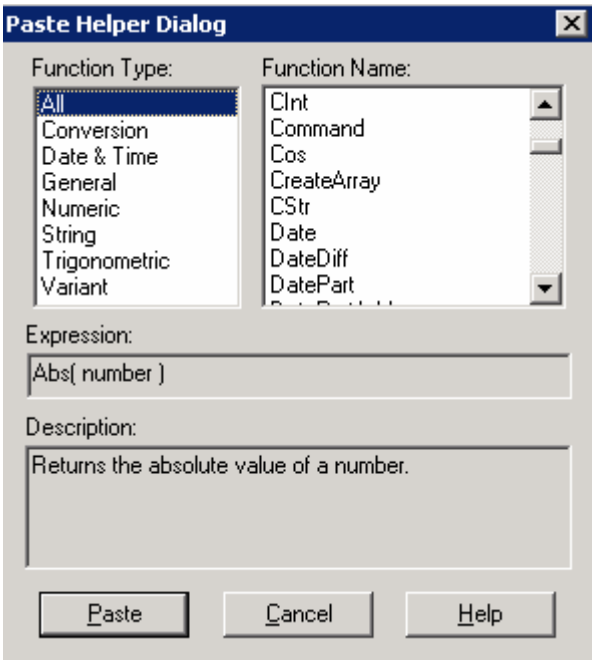


Figure 187 : Liste non exhaustive des fonctions disponibles

Ci-dessous, les propriétés de l'adaptateur code_pere qui permet de renseigner le port catalogue_pere.

Si la longueur de la valeur du port est égale à 2, la valeur de sortie est NULL.

Sinon, les deux caractères de gauche du port d'entrée string servent à renseigner le port de sortie outpout1.

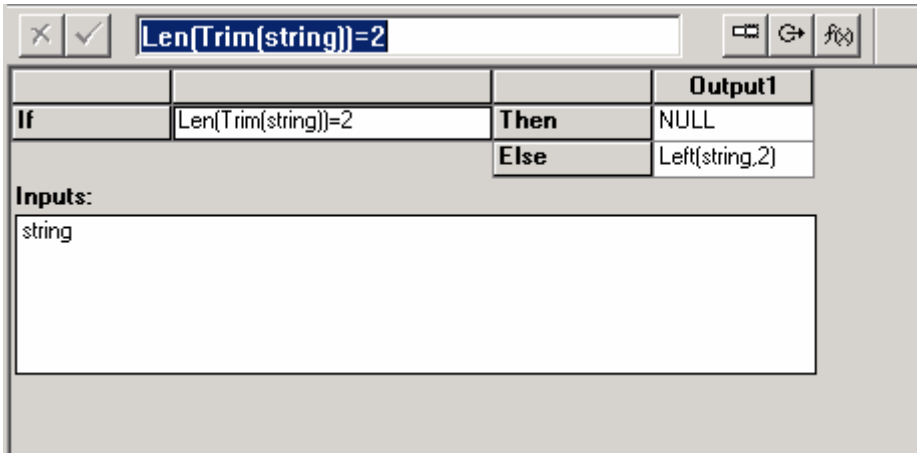


Figure 188 : Adaptateur If / Then

L'adaptateur TYPE_RAYON est une variable qui contient la valeur 1. Cette valeur caractérise la hiérarchie des rayons et des unités de gestion.

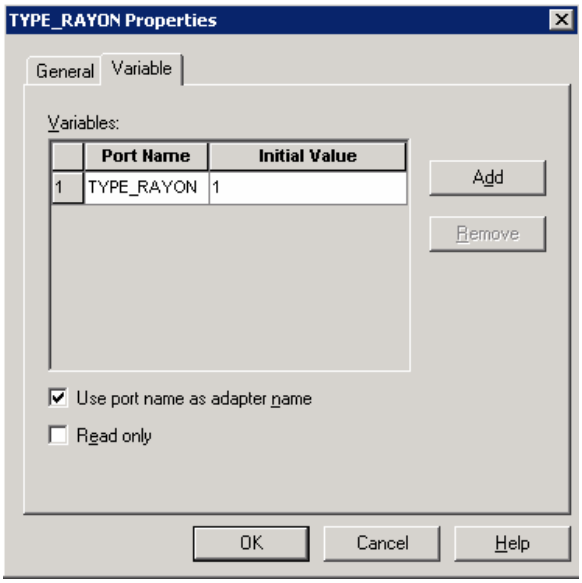


Figure 189 : Adaptateur variable dont le nom est TYPE_RAYON et la valeur est 1

L'ensemble des diagrammes de flux utilisés dans GLOBAL est construit de la même manière.

Afin d'utiliser l'adaptateur permettant de manipuler des fichiers plats, il faut renseigner de nombreuses propriétés, comme nous pouvons le voir sur les onglets ci dessous *(Figure 190, Figure 191, Figure 192).*

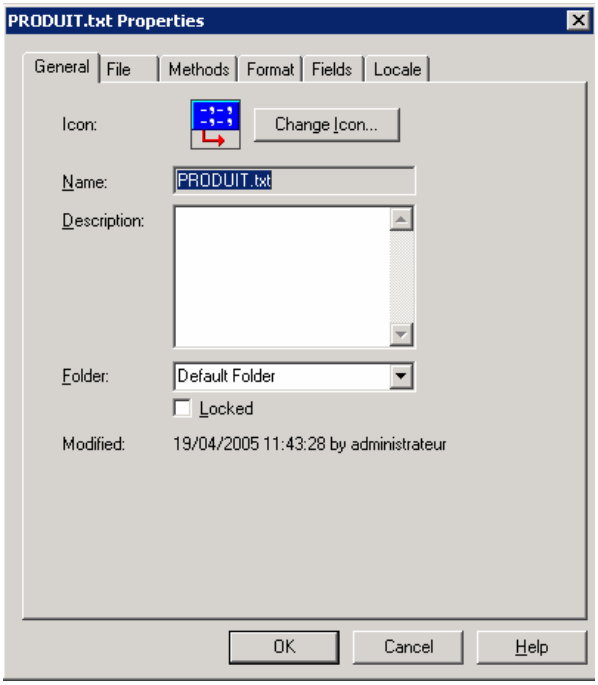


Figure 190 : Adaptateur Flat File - information générale

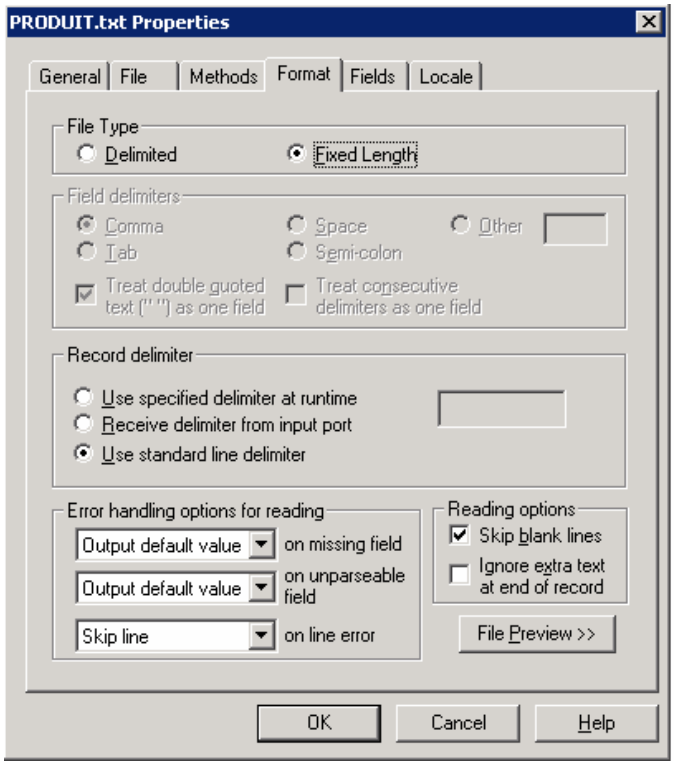


Figure 191 : Adaptateur Flat File - format du fichier

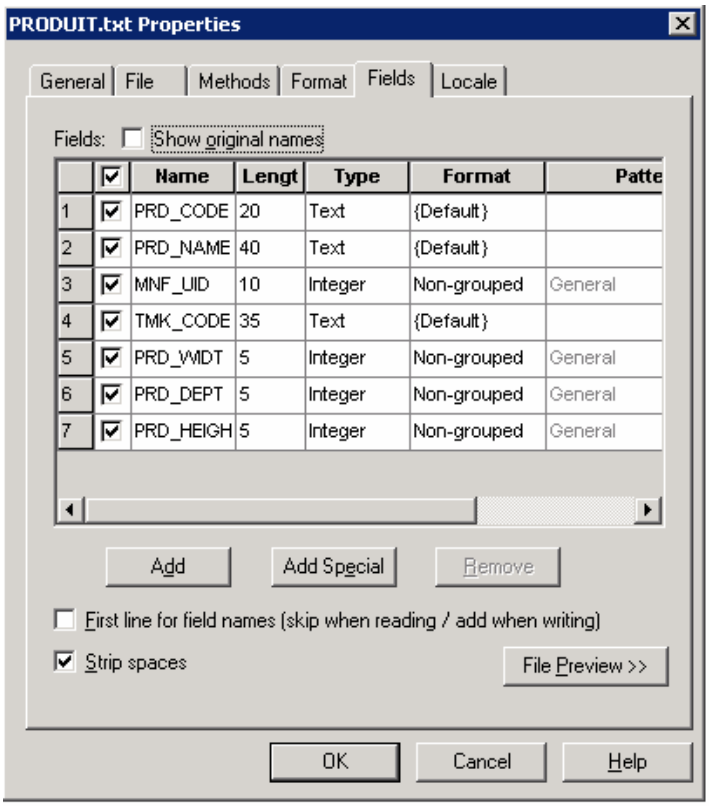


Figure 192 : Adaptateur Flat File - propriétés des champs du fichier

Ce dernier onglet permet de préciser les caractéristiques des champs du fichier.

L'adaptateur permettant de manipuler les tables d'un schéma d'une base de donnée, comporte lui aussi de nombreuses propriétés afin d'être exploité.

Dans l'onglet tables, on sélectionne les tables que l'on souhaite manipuler.

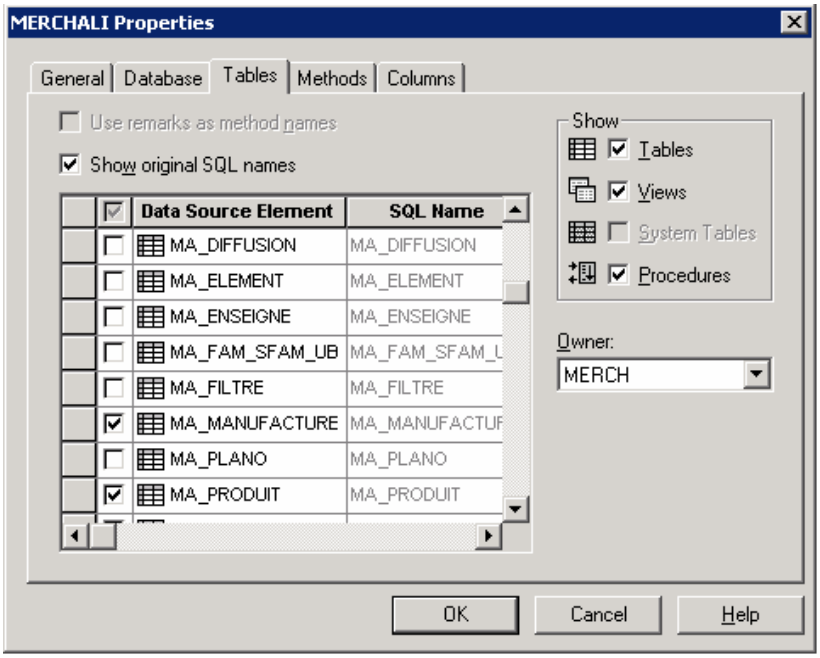


Figure 193 : Adaptateur ODBC - choix des tables

Dans l'onglet méthode, on indique les méthodes que l'on veut utiliser.

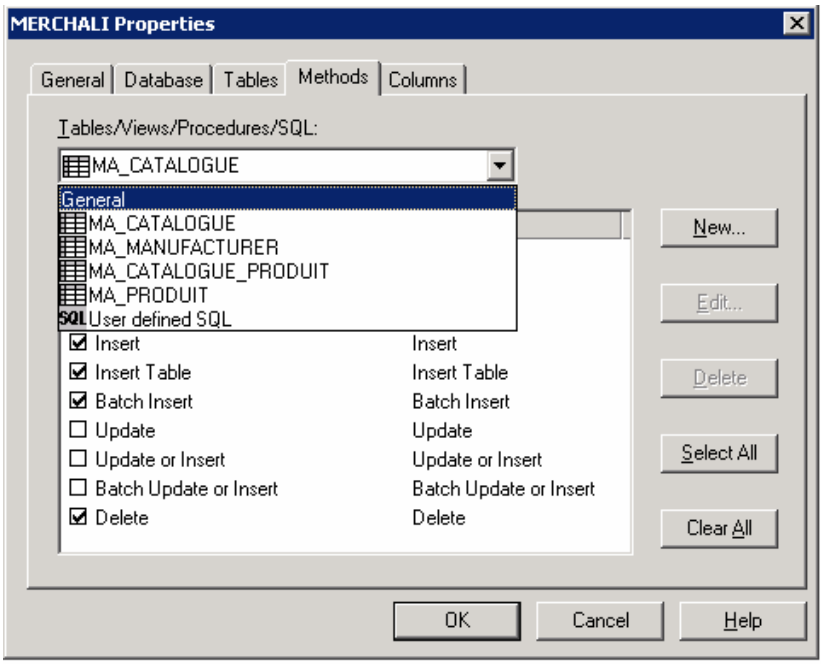


Figure 194 : Adaptateur ODBC - sélection des méthodes

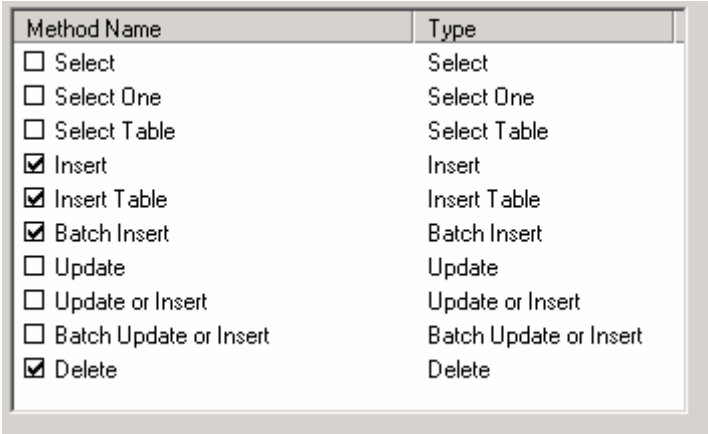


Figure 195 : Adaptateur ODBC - sélection des méthodes (suite)

La liste des adaptateurs utilisés dans cet exemple figure dans la librairie ci-dessous.

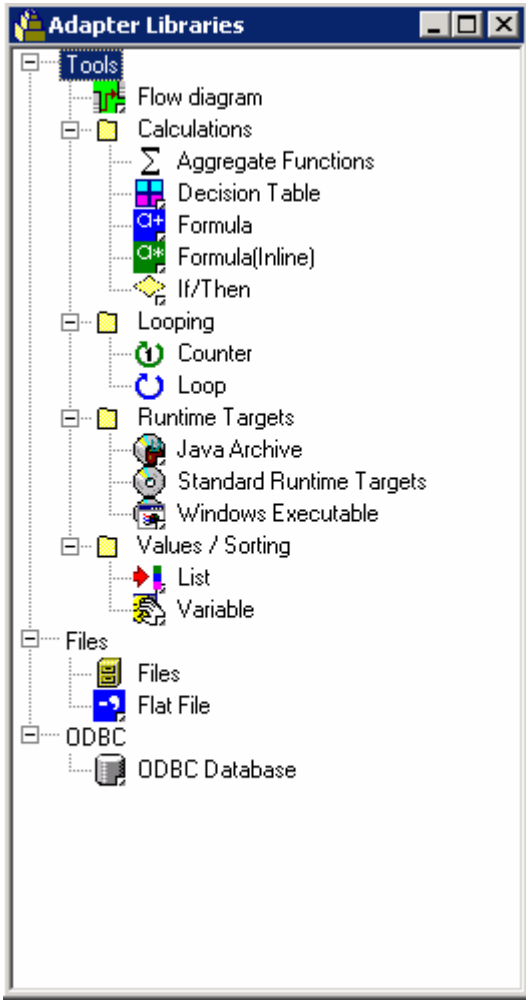


Figure 196 : Librairie basique des adaptateurs

L'objectif de cet exemple n'est pas de passer en revue l'intégralité des options et paramétrages possible dans VBIS, mais d'appréhender la façon dont nous pouvons effectuer des développements avec ce progiciel. Dans cet exemple, nous avons utilisé très peu d'adaptateurs par rapport au nombre disponible et seulement les plus basiques. Nous pouvons

déjà imaginer que, plus nous utilisons d'adaptateurs différents et plus les traitements à implémenter sont complexes, plus le projet VBIS devient rapidement complexe. Il est donc nécessaire d'investir beaucoup de temps afin de réduire cette complexité.

Annexe C : Exemple de développement J2EE

Présentation

Afin d'illustrer les développements qui ont été mis en place, nous allons utiliser l'application GDPADMIN.

Comme nous l'avons vu, cette application permet de gérer les notions utilisées par l'application frontale GDP.

Parmi ces notions, l'administrateur peut gérer les processus. Un écran permet de consulter la liste des processus, de créer, de modifier ou de supprimer un processus (Figure 197).

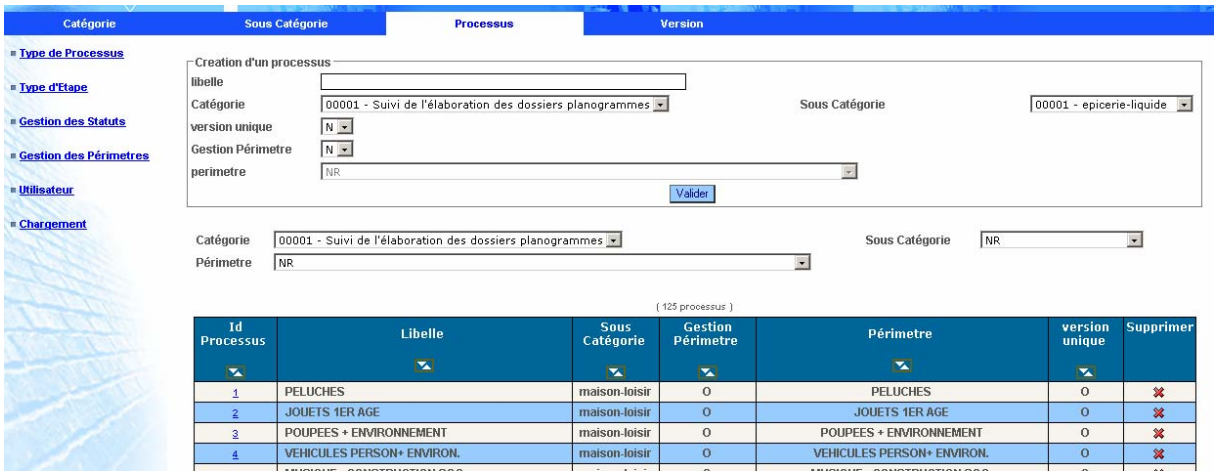


Figure 197 : Ecran de gestion des processus

Par exemple, pour créer un processus, l'administrateur doit :

- Indiquer un libellé qui correspond au nom du processus.
- Choisir une sous catégorie. Il choisit dans un premier temps une catégorie, puis une sous-catégorie appartenant à cette catégorie. Cela suppose que la sous-catégorie et la catégorie correspondante aient été préalablement créées.
- Indiquer si la version correspondant au processus est unique ou pas. Si on choisit la version unique, dans le frontal n'apparaît qu'une seule version courante. Sinon, les utilisateurs du frontal voient plusieurs versions courantes d'un même processus. Dans le cas de la catégorie « suivi de l'élaboration des planogrammes », la version est toujours unique.
- Indiquer si un périmètre est associé au processus. Si c'est le cas, il faut indiquer le périmètre. Dans le cas de la catégorie « suivi de l'élaboration des planogrammes », un périmètre est toujours associé à un processus.

Nous allons maintenant voir :

- le cas d'utilisation correspondant à la gestion des processus
- le modèle de données utilisé
- le design de l'application WEB GDPADMIN
- des exemples de code correspondant à la fonctionnalité de création du processus.

Cas utilisation

Comme nous pouvons le voir sur le cas d'utilisation ci-dessous, pour créer un processus, l'administrateur doit créer la sous catégorie, la catégorie correspondante et le périmètre correspondant. La création du périmètre implique automatiquement sa sauvegarde en base de données. Pour ne pas surcharger le diagramme nous n'avons pas indiqué les possibilités de visualisation des listes, des modifications, des suppressions et des sauvegardes correspondant aux catégories, sous catégories, périmètres et versions. En fait, nous avons adopté la même organisation de l'application pour l'ensemble des notions gérées.

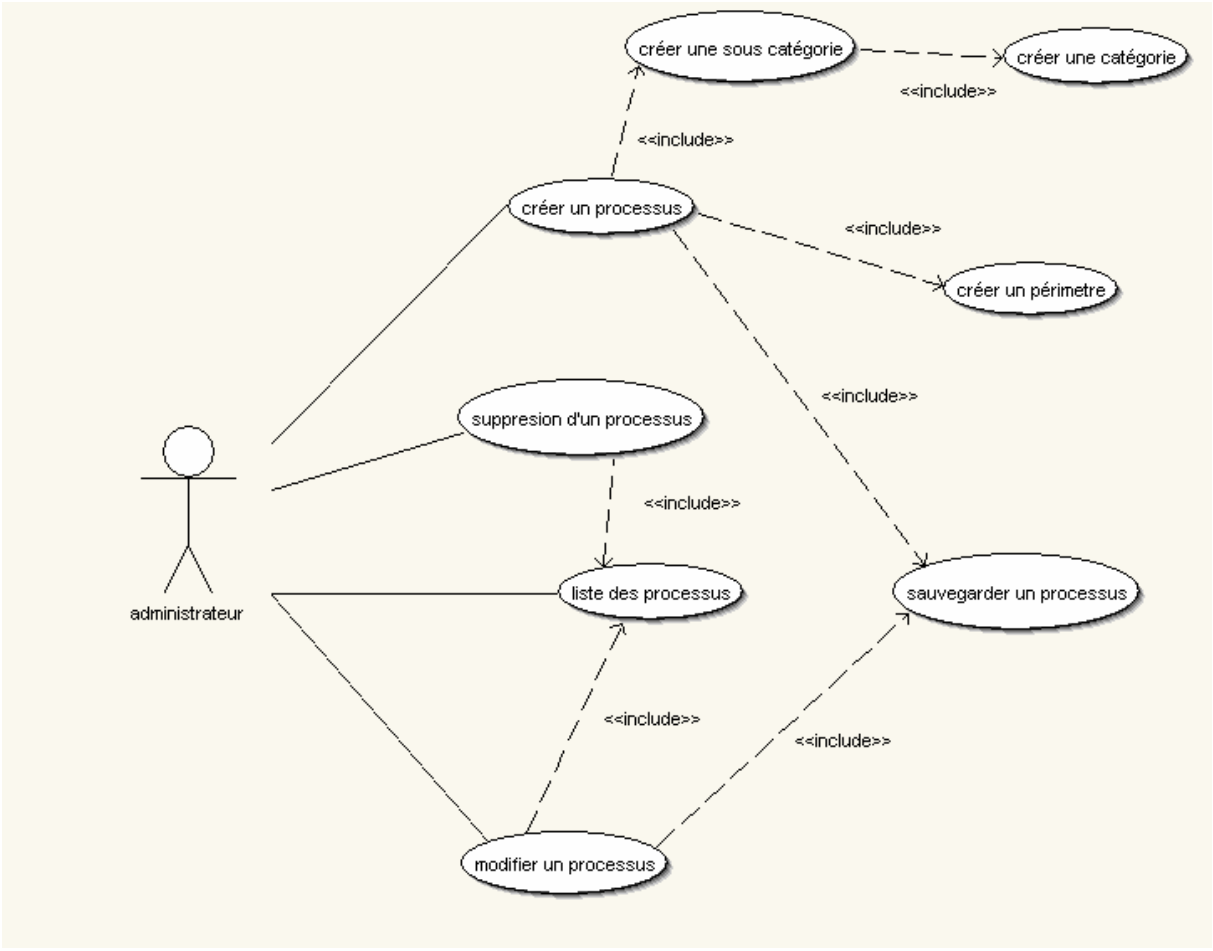


Figure 198 : Cas d'utilisation de la gestion d'un processus

Modèle de données

Le Modèle Conceptuel de Données ou MCD utilisé par l'application est présenté sur les deux schémas ci-dessous.

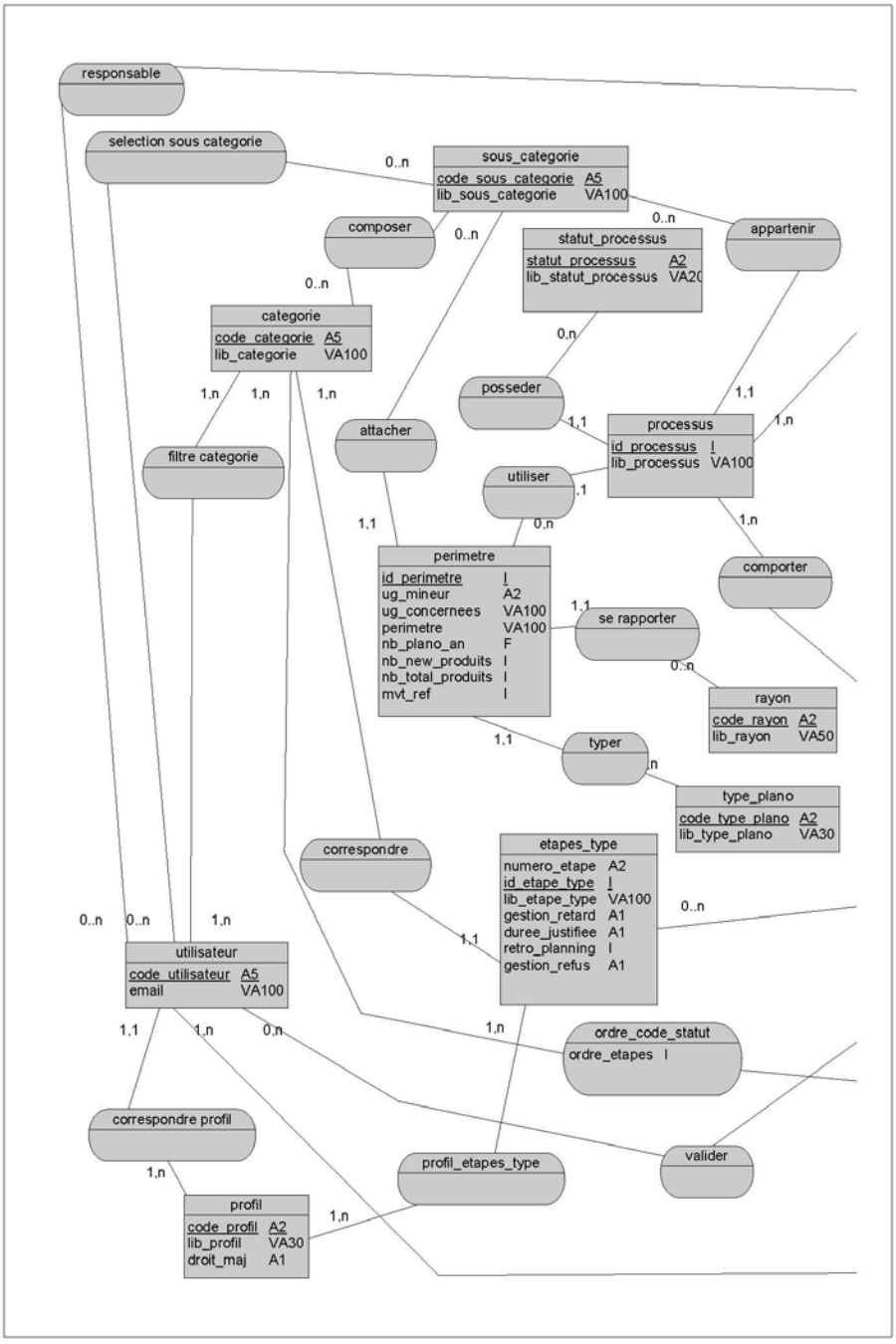


Figure 199 : MCD partie 1/2

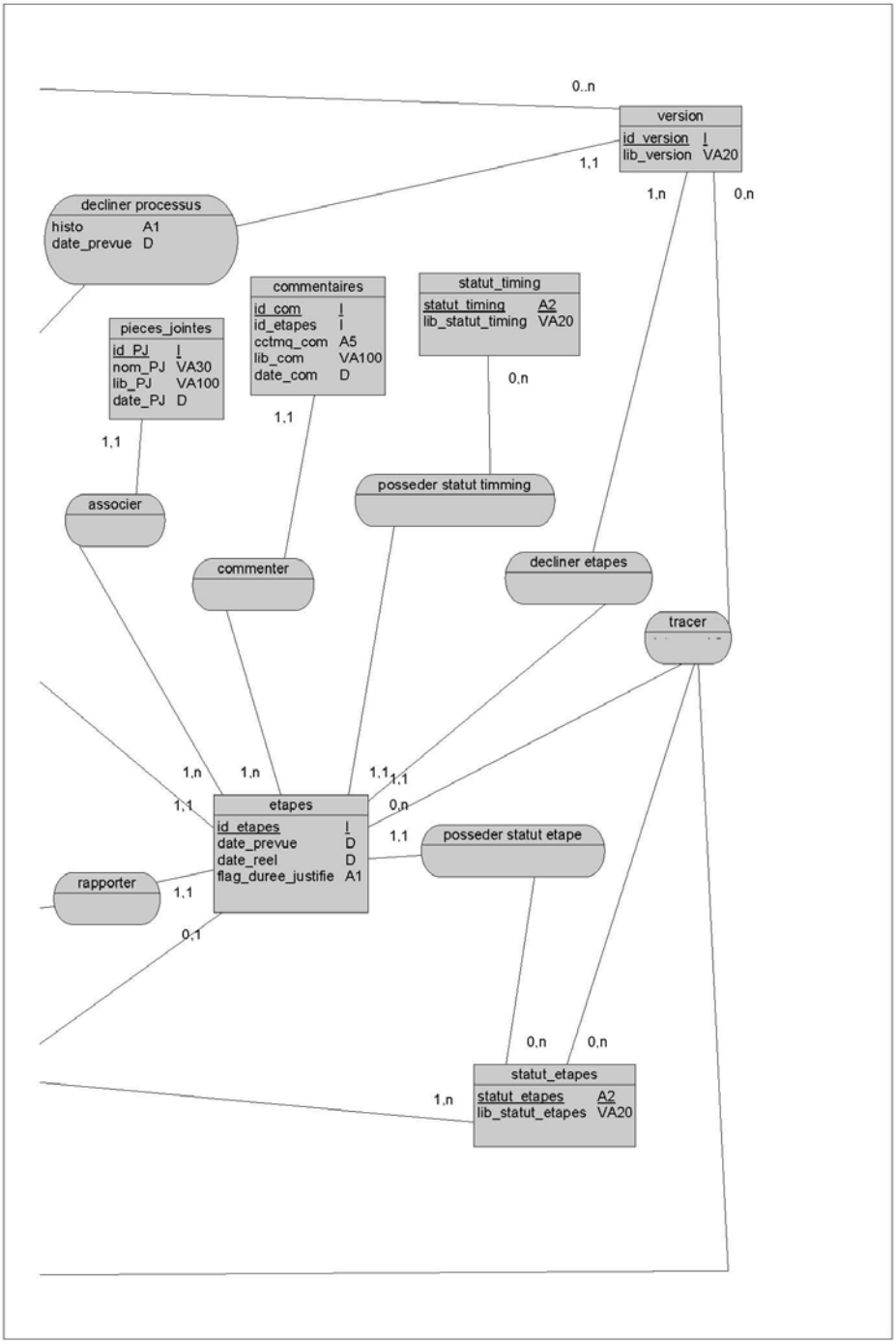


Figure 200 : MCD partie 2/2

Une catégorie correspond à un ensemble de sous catégories qui regroupent des processus. Une catégorie comprend un ensemble ordonné d'étapes types.

Un processus se décline en versions qui comportent des étapes. Un processus utilise un périmètre. Ce périmètre se rapporte à un rayon et est associé à un type de planogramme.

Une étape possède une étape type, des pièces jointes associées, des commentaires, ainsi qu'un statut timing. Elle est caractérisée par une version et un statut étape.

Un utilisateur appartient à un profil. Un profil peut avoir des autorisations sur les étapes types. Un utilisateur peut valider une étape. Un utilisateur pour un profil donné peut être responsable d'une version. Enfin, il n'accède qu'aux sous catégories auxquelles il a le droit.

Nous ne nous attarderons pas sur ce MCD. En effet, nous allons plutôt utiliser la notion de processus. Le fait de passer en revue tout le modèle en détail serait long et fastidieux et ne présente que peu d'intérêt par rapport à ce que nous allons voir.

Le MLD correspondant à ce MCD a été mis dans l'Annexe F. La base de données relationnelle utilisée pour implémenter ce modèle physique de donnée est une base de données ORACLE 9i.

La conception de l'application

L'application GDPADMIN comporte 3 couches distinctes :

- La couche Présentation

Elle est réalisée à l'aide du framework Struts.

- La couche DAO

Elle a été implémentée à l'aide de classe Java.

Les Patterns Fabrique Abstraite, Singleton et DAO ont été mis en œuvre afin de fournir des objets « métier ». Ces objets « métier » se présentent sous la forme de JavaBeans et sont directement exploités par les actions Struts. Ils sont envoyés aux Vues sous forme de collections. Le framework Hibernate a été mis en place, afin de bénéficier des avantages de la conception objet. Les tables Oracle sont mappées à des classes java. Ces classes java sont ensuite utilisées pour fabriquer des JavaBeans « métier ». Les JavaBeans « métier » sont fournis à la couche de présentation.

Il n'y a pas de couche « métier » en soit. C'est pour cette raison que nous parlerons uniquement d'une couche d'accès aux données. Les JavaBeans « métier » ne sont que des classes java que les couches de présentation et les couches d'accès aux données utilisent pour communiquer entre elles.

- La couche physique

Elle est implémentée à l'aide de tables Oracle.

Pour rappel, voici le schéma correspondant à l'architecture en couche de l'application.

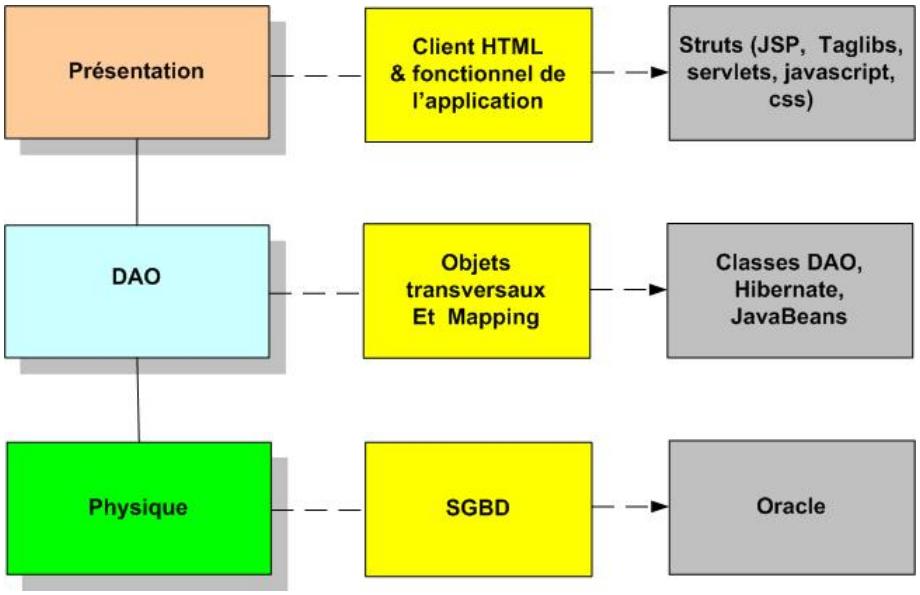


Figure 201 : Architecture de l'application

Arborescence de l'application

L'arborescence de l'application comprend un répertoire de base qui porte le même nom que l'application.

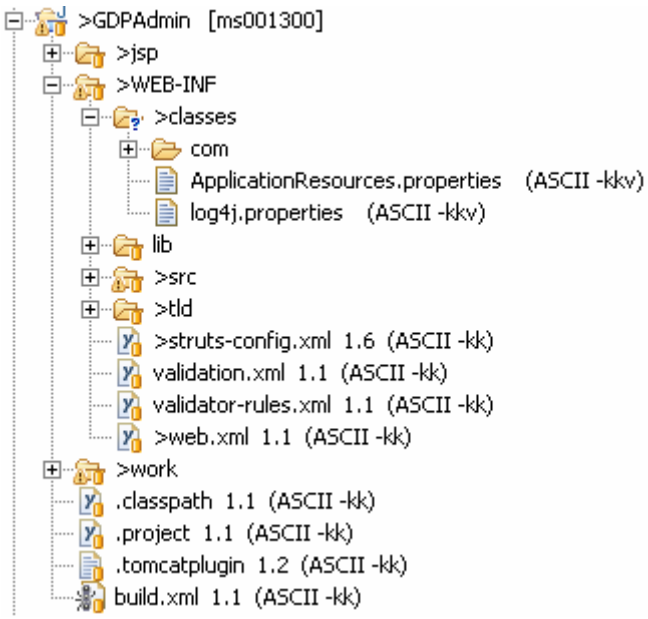


Figure 202 : Arborescence de l'application GDPADMIN

Ce répertoire de base comporte 2 dossiers importants :

- un dossier jsp qui contient les pages jsp utilisées par l'application.
- un dossier WEB-INF contenant principalement les sources de l'application

Le dossier WEB-INF possède 4 sous dossiers :

- classes
  Ce répertoire contient les fichiers java compilés nécessaires à l'application. Ces fichiers possèdent l'extension .class et sont organisés selon leur package.

- lib
  Ce répertoire contient les bibliothèques nécessaires et spécifiques à l'application. Leur extension est .jar. On y trouve les archives java correspondant à Stuts, Hibernate, Ant, Oracle, log4j, mail …

- src
  Ce répertoire contient les sources java (.java).

- tld
  Ce répertoire contient les fichiers de description des librairies Struts et des Tags personnalisés. Ces fichiers possèdent l'extension .tld.

Dans le dossier jsp, les vues ou les pages jsp sont classées selon les choix proposés dans le menu vertical de l'application. L'écran de gestion des processus est accessible à l'aide du choix Processus Type. La page Processus.jsp correspondant à cet écran appartient au dossier processustype.

Le dossier common contient les pages jsp correspondant aux en-têtes, menus verticaux utilisés pour construire toutes les pages de l'application.

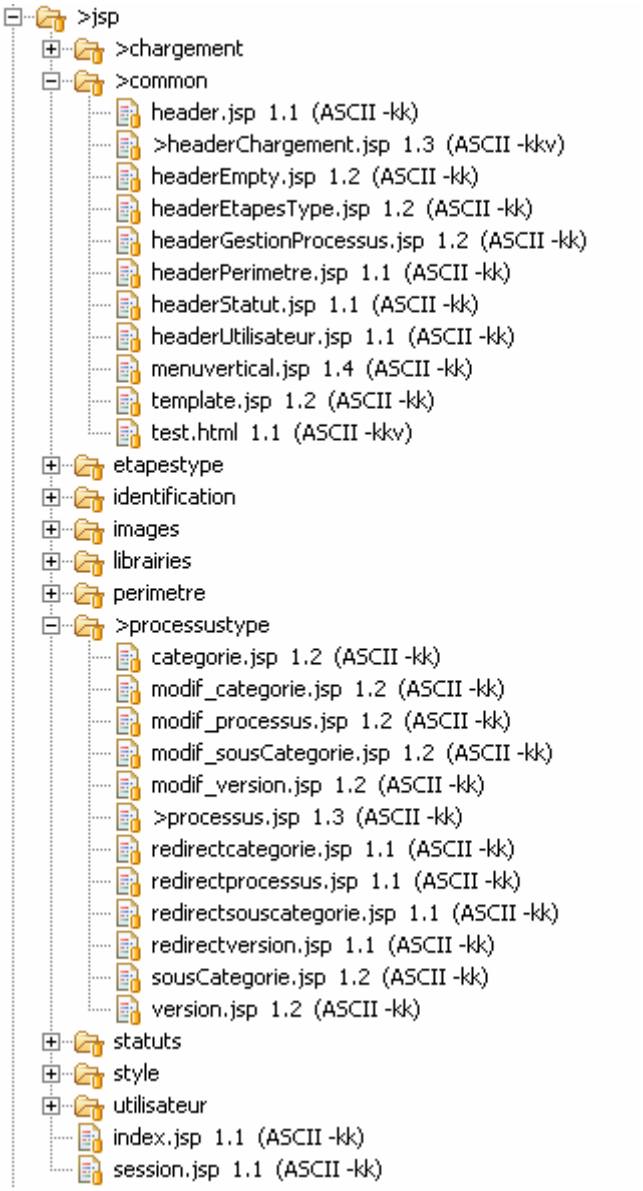


Figure 203 : Dossier jsp contenant les pages jsp correspondant aux vues utilisées dans l'application

Le dossier src contient toutes les sources java utilisées par l'application. Ces sources sont organisées en package.

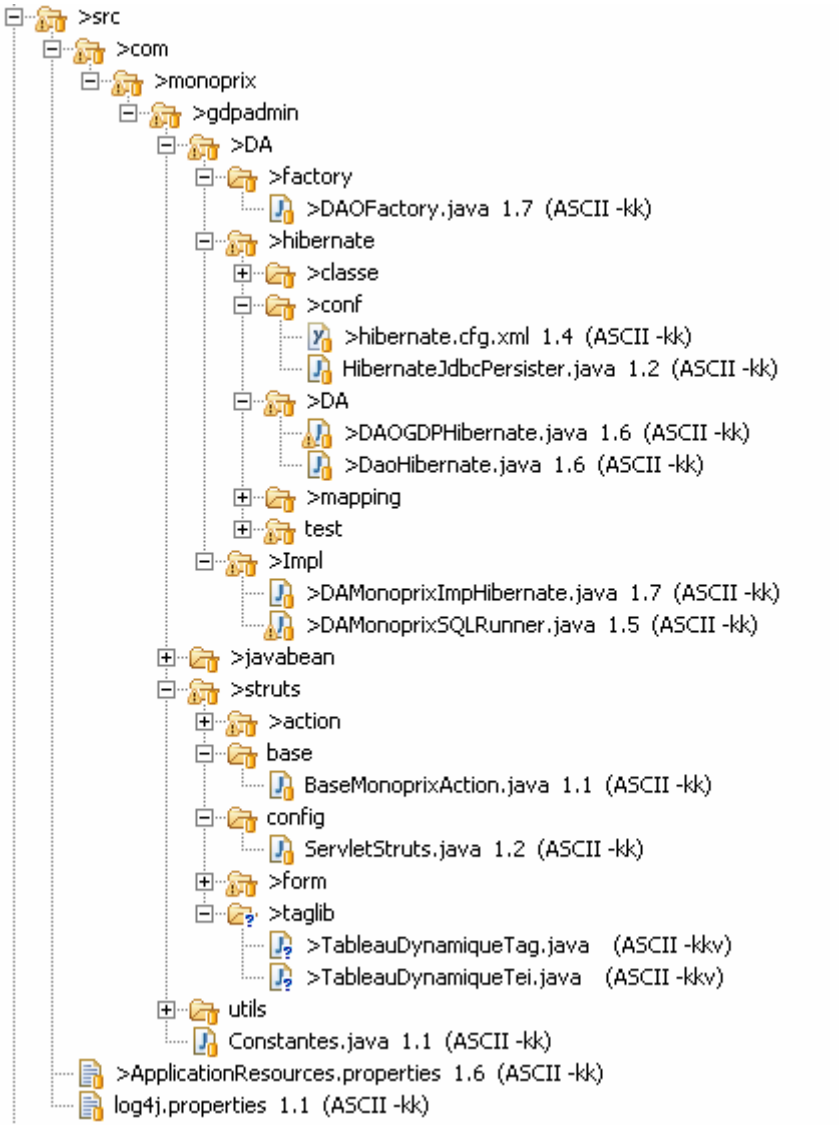


Figure 204 : Dossier src contenant les sources java de l'application

Packages

Le tableau ci-dessous passe en revue l'ensemble des packages utilisés dans l'application.

Tableau 35 : Liste des packages de l'application

Nom du Package	Description
Com.monoprix.gdpadmin.DA	Ce package regroupe les classes utilisées pour accéder aux données.
Com.monoprix.gdpadmin.DA.factory	Contient la classe de fabrication abstraite
Com.monoprix.gdpadmin.DA.hibernate	Contient les classes java qu'utilise Hibernate
Com.monoprix.gdpadmin.DA.hibernate.classe	Contient les classes java auxquelles sont mappées les tables Oracle
Com.monoprix.gdpadmin.DA.hibernate.conf	Contient le fichier de configuration d'Hibernate
Com.monoprix.gdpadmin.DA.hibernate.DA	Contient les classes d'accès aux données proprement dites qui fournissent à la factory des JavaBeans « métier » : DAOGDPHibernate et Daohibernate.
Com.monoprix.gdpadmin.DA.hibernate.mapping	Contient les fichiers de mapping Hibernate
Com.monoprix.gdpadmin.DA.hibernate.test	Permet de tester les composants d'accès aux données
Com.monoprix.gdpadmin.DA.impl	Contient les différentes fabriques concrètes correspondant à la fabrique abstraite.
Com.monoprix.gdpadmin.javaBeans	Ce package regroupe les classes correspondant au modèle « métier »
Com.monoprix.gdpadmin.struts	Ce package regroupe les classes utilisées par la couche de présentation
Com.monoprix.gdpadmin.struts.action	Contient les classes actions utilisées dans l'application
Com.monoprix.gdpadmin.struts.base	Contient la classe de base dont héritent toutes les actions de l'application. Cette classe BaseMonoprixAction hérite de la classe action de Struts.
Com.monoprix.gdpadmin.struts.form	Contient les actionsForms de Struts
Com.monoprix.gdpadmin.struts.taglib	Contient les classes java correspondant aux tags personnalisés
Com.monoprix.gdpadmin.utils	Ce package contient des classes utilitaires telles que le formatage de date

Fichiers importants

Les principaux fichiers de paramétrage utilisés dans l'application sont présentés dans le tableau ci-dessous.

Tableau 36 : Fichiers de configuration importants

Nom du fichier	Origine	Description
Web.xml	Standard J2EE	Fichier nommé descripteur de déploiement. Il contient la configuration de l'application.
Struts-config.xml	Struts	Fichier de configuration Struts
Validation.xml	Struts	Ce fichier XML définit les règles utilisables pour tester les valeurs des champs des vues. Il effectue le lien entre le type de validation (champ requis, champ email …) et le chemin vers la méthode et la classe correspondante qui permettront d'effectuer l'opération de vérification des données (valeur des champs). Par défaut, Struts Validator prend en charge les champs vides, les formats d'email ou de date invalides, les longueurs minimales ou maximales des champs …
Validator-rules.xml	Struts	Ce fichier permet de mapper les champs de nos formulaires aux validations définies dans validator-rules.xml.
ApplicationsRessources.properties	Struts	Ce fichier permet d'associer pour chaque valeur dynamique, une clé unique indépendante. Par exemple, pour afficher les messages dans l'application, on utilise les identifiants et non les valeurs. Ainsi, en cas de changement d'une valeur, il suffit simplement de modifier ce fichier.
Hibernate.cfg.xml	Hibernate	Fichier de configuration utilisé par Hibernate.
Fichiers .hbm.xml	Hibernate	Fichiers permettant de mapper les tables d'une base de données relationnelle telle qu'Oracle aux classes java. Dans le cas de la gestion des processus, le fichier de mapping correspondant est Processus.hbm.xml.
Build.xml	Ant	Fichier décrivant les tâches de compilation et de déploiement.
Log4j.properties	Log4j	Fichier de configuration de log4j

Diagramme de classe

L'écran processus dont nous avons parlé au début correspond en fait à une page JSP nommée Processus.jsp. Cette page jsp en Struts correspond à une vue.

A cette vue est associée une classe héritant de org.apache.struts.action.ActionForms. Cette classe nommée ProcessusForm permet de valider et de récupérer les données saisies par l'administrateur (*Figure 205*).

Dans le cas de la création d'un processus, elle est utilisée par la classe CreationProcessusAction.java qui hérite de la classe baseMonoprixAction héritant elle-même de la classe struts org.apache.struts.action.Action.

L'action transforme l'objet de la classe ProcessusForm en objet de la classe ProcessusMetier. Elle sollicite la fabrique abstraite, en appelant la méthode correspondante à la création d'un processus.
L'objet de la classe ProcessusMetier est fourni comme un argument de la méthode appelée.

Une implémentation de fabrique est alors choisie à l'aide d'un paramètre.
Dans notre cas, c'est l'implémentation DAMonoprixImpHibernate qui est utilisée.

Cette fabrique concrète appelle une méthode de la classe DAOGDPHibernate en lui passant l'objet de la classe ProcessusMetier en argument.
Cette classe hérite de la classe Hibernate HibernateJdbcPersister qui gère les sessions Hibernate en se basant sur le fichier de configuration d'Hibernate hibernate.cfg.xml.

DAOGDPHibernate appelle une méthode de la classe DaoHibernate en lui passant l'objet de la classe ProcessusMetier en argument.
L'objet de la classe ProcessusMetier est alors converti en objet de la classe Processus.

A l'aide d'une session Hibernate, l'objet de la classe Processus est utilisé pour effectuer une sauvegarde en base de données.

Les informations saisies par l'administrateur correspondant à la création de son processus sont alors enregistrées en base de données.

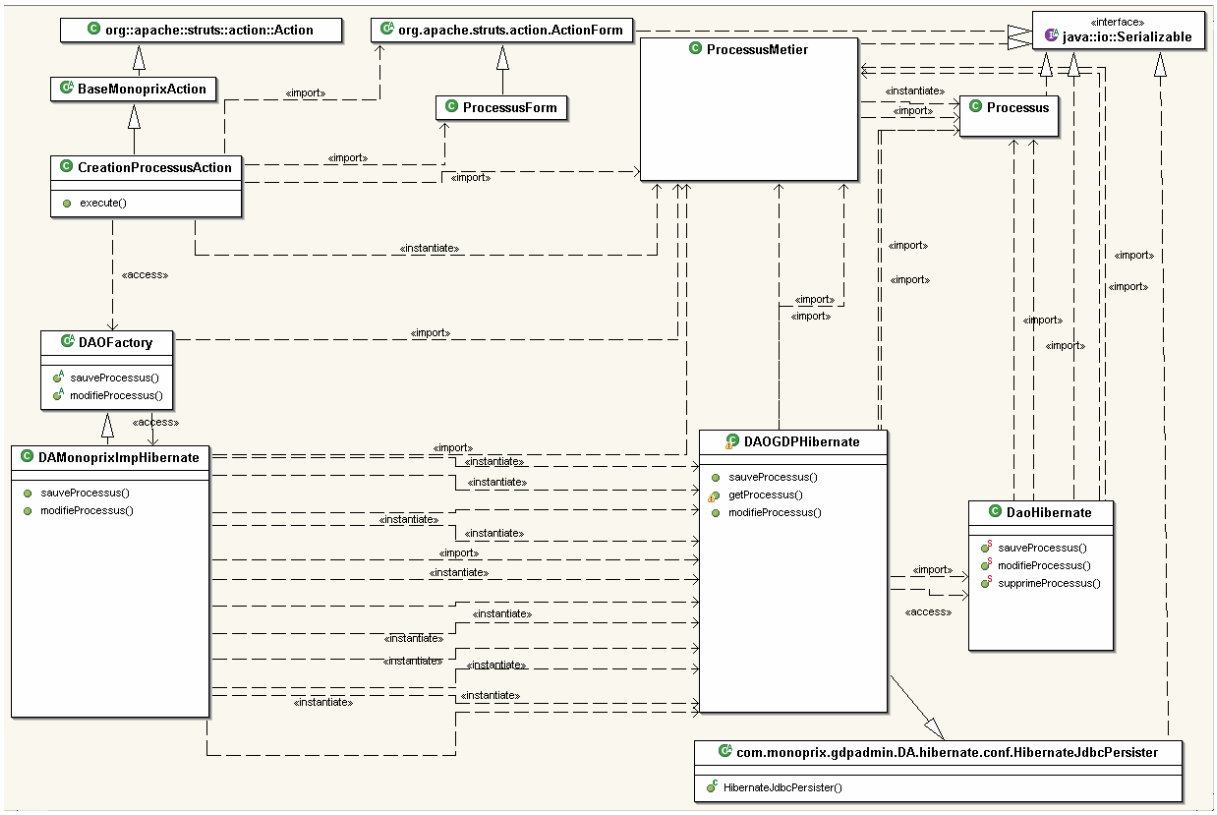


Figure 205 : Diagramme de classes global

Les classes ActionForm telles que ProcessusForm et les JavaBeans « métier » tels que ProcessusMetier ne présentent pas la même structure (Figure 206). Tous les attributs de la classe ActionForm possèdent un type String afin de faciliter leur manipulation. Ces chaînes sont ensuite converties aux formats correspondant à la classe « métier » dans l'action Struts.

La modification et la création d'un processus sont gérées dans deux vues différentes. Nous utilisons le même ActionForm pour ces deux vues, mais les champs utilisés pour la modification ne sont pas les mêmes que pour la création. Leurs noms sont suffixés de mod.

Des champs suffixés par select ont également été introduits. Ils permettent de filtrer la liste des processus sur une catégorie ou une sous catégorie par exemple.

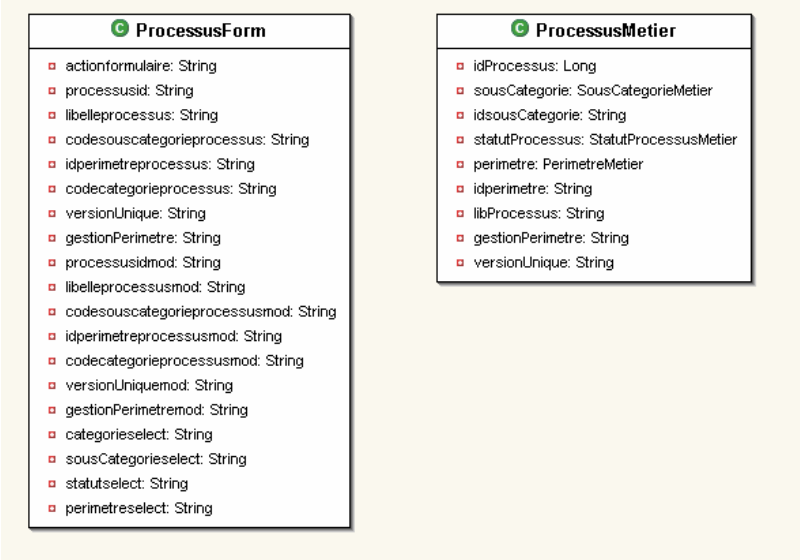


Figure 206 : Diagramme de classe correspondant aux classes métier et ActionForm

Les classes java utilisées par Hibernate qui sont mappées aux tables de la base Oracle ne présentent également pas la même structure *(Figure 125)*. Ceci afin de faciliter la gestion des classes au niveau de la couche Hibernate. La classe ProcessusMetier possède une méthode toProcessusHibernate qui permet, à partir d'un objet de la classe ProcessusMetier, d'obtenir un objet de la classe Processus.

De plus, la classe ProcessusMetier possède un constructeur prenant pour argument un objet de la classe Processus. Nous pouvons ainsi obtenir un objet « métier » à partir des classes Hibernate.

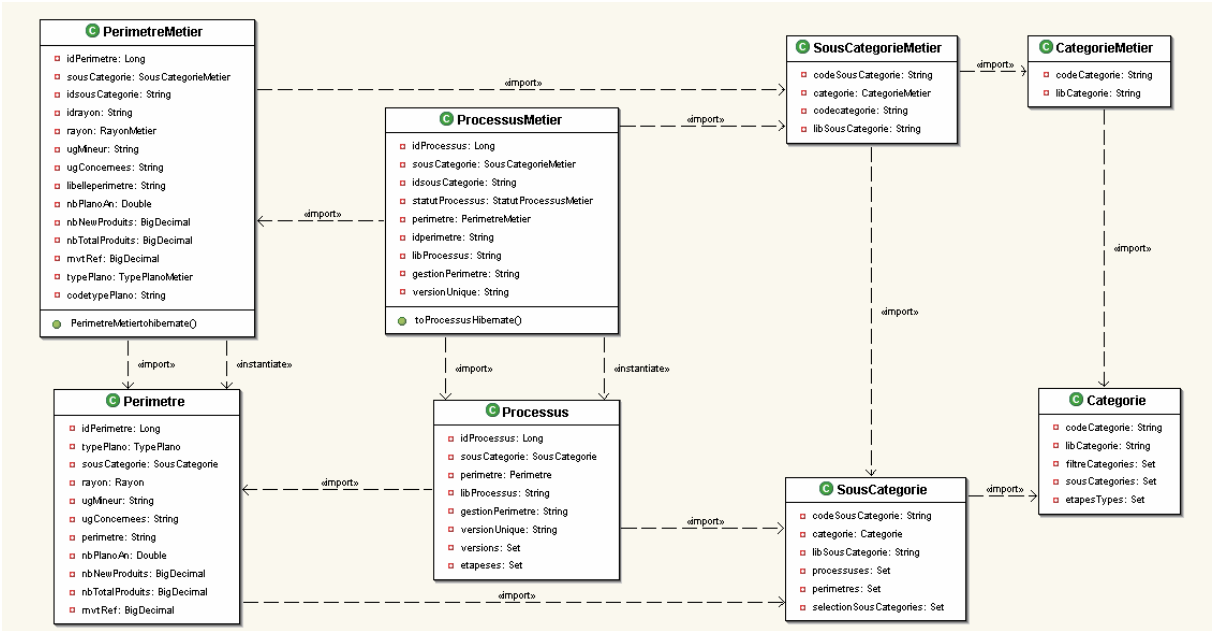
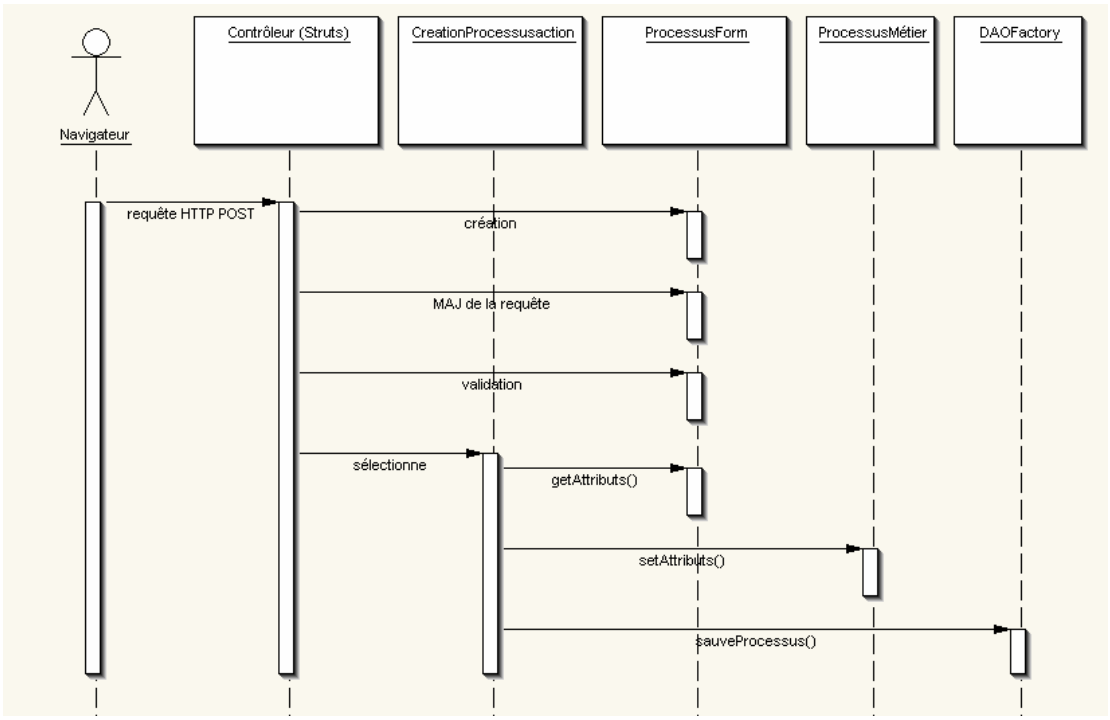


Figure 207 : Diagramme de classe correspondant aux classes métier et aux classes Hibernate

Diagramme de séquence

Ci-dessous le diagramme de séquence correspondant à la création d'un processus. Nous allons le détailler.



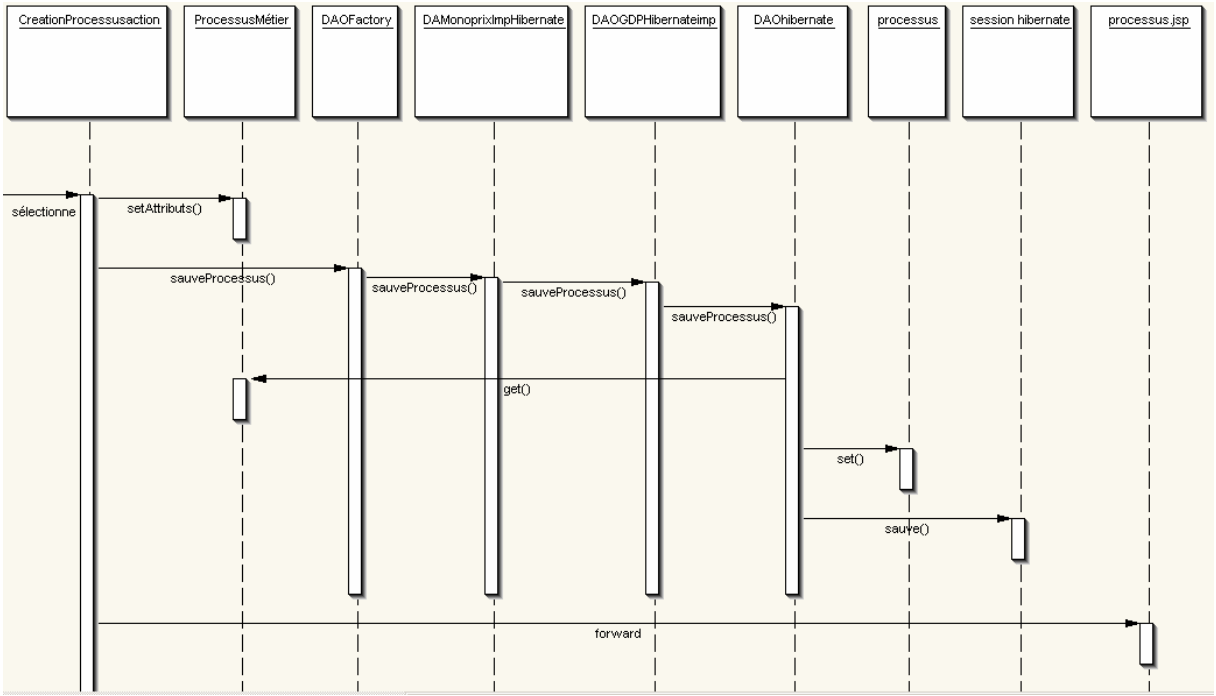


Figure 208 : Diagramme de séquence de la création d'un processus

Le contrôleur

Dans l'écran processus, l'administrateur renseigne les informations caractérisant un processus puis clique sur le bouton Valider.

Figure 209 : Portion de l'écran processus permettant la création d'un processus

La page correspondant à cet écran est Processus.jsp. Elle correspond à la Vue du modèle MVC.

Le fait de cliquer sur le bouton Valider envoie une requête au servlet contrôleur de Struts qui va sélectionner une action.

C'est le fichier struts-config.xml *(Figure 210)* qui permet de faire le lien entre l'action, la vue et l'ActionForm.

La balise form-bean permet de déclarer l'ActionForm processusForm.

La balise action fait le lien entre l'action, l'ActionForm et la page JSP. Elle indique également ce qu'il faut faire en cas de succès, à l'aide de la base forward. Dans notre cas, elle renvoie vers une vue qui effectue une redirection vers la page processus.jsp.

```
<struts-config>
	<form-beans>
		<form-bean name="processusForm" type="com.monoprix.gdpadmin.struts.form.ProcessusForm" />
	</form-beans>
	<action-mappings type="org.apache.struts.action.ActionMapping">
		<action path="/CreationProcessus" name="processusForm"
		type="com.monoprix.gdpadmin.struts.action.processustype.CreationProcessusAction" validate="true"
		input="/jsp/processustype/processus.jsp" >
			<forward  name="success" contextRelative="true" path="/jsp/processustype/redirectprocessus.jsp" />
		</action>
	</action-mappings>
	<message-resources parameter="ApplicationResources" />
</struts-config>
```

Figure 210 : Fichier struts-config.xml

L’action

La Classe CreationProcessusAction convertit l’ActionForm processusForm en objet « métier » processusMetier, sollicite la fabrique abstraite pour sauvegarder le processus et gère les messages de réussite ou d’échec à destination de la vue.

```
package com.monoprix.gdpadmin.struts.action.processustype;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.action.ActionMessages;

import com.monoprix.gdpadmin.Constantes;
import com.monoprix.gdpadmin.DA.factory.DAOFactory;
import com.monoprix.gdpadmin.javabean.ProcessusMetier;
import com.monoprix.gdpadmin.struts.base.BaseMonoprixAction;
import com.monoprix.gdpadmin.struts.form.ProcessusForm;

public class CreationProcessusAction extends BaseMonoprixAction {

	public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request,
			HttpServletResponse response) throws IOException, ServletException {

		//  test authentification
		ActionForward af = super.checkAuthentification(mapping, request);
		if (af != null) {
			return af;
		}

		// conversion de l’ActionForm en object Metier

		ProcessusForm processusform = (ProcessusForm) form;
		ProcessusMetier processusmetier = new ProcessusMetier();
		processusmetier.setLibProcessus(processusform.getLibelleprocessus());

		if(processusform.getIdperimetreprocessus()!=null)
		{
			if(processusform.getIdperimetreprocessus().equals(Constantes.CHAMPS_NR))
			{
				processusmetier.setIdperimetre(null);
			}
			else
			{
				processusmetier.setIdperimetre(processusform.getIdperimetreprocessus());
			}
		}
		else
		{
			processusmetier.setIdperimetre(null);
		}
		processusmetier.setIdsousCategorie(processusform.getCodesouscategorieprocessus());
		processusmetier.setGestionPerimetre(processusform.getGestionPerimetre());
		processusmetier.setVersionUnique(processusform.getVersionUnique());

		// sollicitation de la fabrique abstraite pour sauvegarder le processus
		int codeRetour=DAOFactory.getDAOFactory().sauveProcessus(processusmetier);

		// gestion des messages  à destination de la vue
		ActionMessages messages = new ActionMessages();

		ActionMessage message;
		if(codeRetour==1)
		{
		message= new ActionMessage("message.processus.creation.success");
		messages.add("succes", message);
		}
		else
		{
		message = new ActionMessage("message.processus.creation.echec");
		messages.add("echec", message);
		}
		this.saveMessages(request, messages);
```

```
                        // initialisation du formulaire
                        processusform.init();
                        // indication du lien courant pour la barre de navigation
                        request.setAttribute("CURRENT_LINK", "processus");
                        // on continue le traitement en cas de succès
                        return mapping.findForward(forwardSuccess);
                        }
}
```

Figure 211 : CreationProcessusAction.java

La fabrique abstraite

La fabrique abstraite DAOFactory implémente le pattern Singleton. Le constructeur de la classe n'est pas accessible. On obtient une instance de la fabrique concrète grâce à la méthode getDAOFactory.

Dans notre cas, la fabrique abstraite sélectionne la fabrique concrète DAMonoprixImpHibernate.

```
package com.monoprix.gdpadmin.DA.factory;
import java.util.List;
import org.apache.log4j.Logger;
import com.monoprix.gdpadmin.DA.Impl.DAMonoprixImpHibernate;
import com.monoprix.gdpadmin.javabean;

public abstract class DAOFactory  {
        /**
         * Logger for this class
         */
        private static final Logger logger = Logger.getLogger(DAOFactory.class);

         public abstract int sauveProcessus(ProcessusMetier processusmetier);

        // implémentation du pattern singleton, le constructeur ne la classe n'est pas accessible
        // on obtient une instance de la classe grâce à la méthode getDAOFactory

        protected DAOFactory()
        {

        }

        public static DAOFactory getDAOFactory()
        {
                if (logger.isDebugEnabled()) {
                        logger.debug("getDAOFactory() - start");
                }

                // la fabrique abstraite sélectionne une fabrique concrète
                // dans notre cas DAMonoprixImpHibernate

                DAOFactory returnDAOFactory = DAMonoprixImpHibernate.getInstance();

                if (logger.isDebugEnabled()) {
                        logger.debug("getDAOFactory() - end");
                }
                return returnDAOFactory;
        }
}
```

Figure 212 : DAOFactory.java

La fabrique concrète

La fabrique concrète DAMonoprixImpHibernate implémente le pattern Singleton. Sa méthode sauveProcessus sollicite la méthode sauveProcessus de la classe DAOGDPHibernate.

```
package com.monoprix.gdpadmin.DA.Impl;
import java.util.List;
import org.apache.log4j.Logger;
import com.monoprix.gdpadmin.DA.factory.DAOFactory;
import com.monoprix.gdpadmin.DA.hibernate.DA.DAOGDPHibernate;
import com.monoprix.gdpadmin.javabean;

public class DAMonoprixImpHibernate extends DAOFactory{

	/**
	 * Logger for this class
	 */
	private static final Logger logger = Logger.getLogger(DAMonoprixImpHibernate.class);

	private static DAMonoprixImpHibernate INSTANCE = new DAMonoprixImpHibernate();

	private static DAOGDPHibernate DAOGDPHibernateimp = new DAOGDPHibernate();

	private DAMonoprixImpHibernate()
	{

	}

	/**
	 * Obtenir l'instance
	 *
	 * @return singleton
	 */
	public static DAMonoprixImpHibernate getInstance() {
		if (logger.isDebugEnabled()) {
			logger.debug("getInstance() - start");
		}

		if (logger.isDebugEnabled()) {
			logger.debug("getInstance() - end");
		}
		return INSTANCE;
	}

	public int sauveProcessus(ProcessusMetier processusmetier) {
		int result=-1;
		try {
			result= DAOGDPHibernateimp.sauveProcessus(processusmetier);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return result;
	}

}
```

Figure 213 : DAMonoprixImpHibernate.java

Session hibernate

La classe DAOGDPHibernate permet de gérer une session Hibernate et des transactions à l'intérieur de cette session. Dans la méthode sauvegardeProcessus, une session est ouverte, une transaction est commencée. Ensuite les objets gérant l'accès aux données sont sollicités par l'instruction suivante :

```
DaoHibernate.sauveProcessus(session, processusmetier);
```

Si la sauvegarde ne lève pas d'exception, un commit de la transaction est effectué. Sinon, en cas d'échec, est déclenché un rollback de la transaction. Enfin, la session Hibernate est fermée.

```
package com.monoprix.gdpadmin.DA.hibernate.DA;

import java.sql;

import org.apache.log4j.Logger;
import org.hibernate.Session;
import org.hibernate.Transaction;

import com.monoprix.gdpadmin.DA.hibernate.classe;
import com.monoprix.gdpadmin.DA.hibernate.conf.HibernateJdbcPersister;
import com.monoprix.gdpadmin.javabean;

public class DAOGDPHibernate extends HibernateJdbcPersister {
	/**
	 * Logger for this class
	 */
	private static final Logger logger = Logger.getLogger(DAOGDPHibernate.class);

	private static final long serialVersionUID = 6411058778623475165L;

	public int sauveProcessus(ProcessusMetier processusmetier) throws Exception {
		Transaction tx = null;
		int result = -1;
		try {
			// ouverture d'un session
			Session session = openSession();
			// début de transaction
			tx = session.beginTransaction();
			// accès  aux objets de gestion des accès
			DaoHibernate.sauveProcessus(session, processusmetier);
			// commit de la transaction
			tx.commit();
			result = 1;
			if (logger.isDebugEnabled()) {
				logger.debug("listeElement() - end");
			}
			return result;

		} catch (Exception exc) {
			logger.error("listeElement()", exc);

			if (tx != null)
				try {
					// rollback
					tx.rollback();
				} catch (Exception e) {
					logger.error("listeElement()", e);
				}
			throw exc;
		} finally {

			// fermeture de la session hibernate
			closeSession();
		}
	}
}
```

Figure 214 :DAOGDPHibernate.java

DAO

La classe DaoHibernate, dans la méthode sauveProcessus, convertit l'objet processusMetier en objet processus. Les objets périmètre et sous catégorie correspondant à l'objet « métier » sont récupérés de la session Hibernate et servent à mettre à jour l'objet processus. L'objet processus est ensuite sauvegardé dans la session Hibernate. La table Oracle PROCESSUS est alors mise à jour.

```
	package com.monoprix.gdpadmin.DA.hibernate.DA;

	import java.io.Serializable;
	import java.math.BigDecimal;
	import java.util;

	import org.apache.log4j.Logger;
	import org.hibernate.HibernateException;
```

```
import org.hibernate.Query;
import org.hibernate.SQLQuery;
import org.hibernate.Session;

import com.monoprix.gdpadmin.DA.hibernate.classe;
import com.monoprix.gdpadmin.javabean;
import com.monoprix.gdpadmin.utils.Utils;

/**
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates
 */
public class DaoHibernate implements Serializable {
	/**
	 * Logger for this class
	 */
	private static final Logger logger = Logger.getLogger(DaoHibernate.class);

	private static final long serialVersionUID = -4971975421693384019L;

	public static int sauveProcessus(Session session,
					ProcessusMetier processusmetier) throws HibernateException {

		// l'objet processusMetier est converti en processus

		Processus processushibernate = processusmetier.toProcessusHibernate();

		// l'objet perimetre correspondant est récupére de la session hibernate
		Perimetre perimetre = null;
		if (processusmetier.getIdperimetre() != null) {
			perimetre = (Perimetre) session.load(Perimetre.class, new Long(
						processusmetier.getIdperimetre()));
		}

		// l'objet souscatégorie correspondant est récupére de la session hibernate
		SousCategorie souscategorie = (SousCategorie) session.get(
					SousCategorie.class, processusmetier.getIdsousCategorie());

		// mise à jour de l'objet processus à l'aide des objets précédents
		processushibernate.setPerimetre(perimetre);
		processushibernate.setSousCategorie(souscategorie);

		// sauvegarde en base de donnée de l'objet processus
		session.save(processushibernate);

		return 0;
	}
}
```

Figure 215 : DaoPHibernate.java

Mapping Hibernate

Le lien entre l'objet processus et la table correspondante est fait à l'aide du fichier de mapping Processus.hbm.xml (*Figure 218*). Le modèle physique de données est également fourni dans l'Annexe F.

Pour une meilleure compréhension du mapping, nous rappelons ci-dessous le script (*Figure 216*) correspondant à la table Oracle Processus, ainsi que la portion de code de la classe processus (*Figure 217*) correspondant à la déclaration des attributs.

```
CREATE TABLE PROCESSUS
(
 ID_PROCESSUS       INTEGER              NOT NULL,
 CODE_SOUS_CATEGORIE  CHAR(5 BYTE)          NOT NULL,
 STATUT_PROCESSUS    CHAR(2 BYTE)          NOT NULL,
 ID_PERIMETRE       NUMBER,
 LIB_PROCESSUS      VARCHAR2(100 BYTE)
)
LOGGING
NOCACHE
NOPARALLEL
NOMONITORING;

CREATE UNIQUE INDEX PK_PROCESSUS ON PROCESSUS (ID_PROCESSUS)
LOGGING
NOPARALLEL;

ALTER TABLE PROCESSUS ADD (  CONSTRAINT PK_PROCESSUS  PRIMARY KEY  (ID_PROCESSUS));
```

Figure 216 : Script de la table processus

```
public class Processus  implements java.io.Serializable {

        // champs
        private static final long serialVersionUID = 5113931358703682691L;
      private Long idProcessus;
        private SousCategorie sousCategorie;
      private Perimetre perimetre;
        private String libProcessus;
        private String gestionPerimetre;
        private String versionUnique;
        private Set versions = new HashSet(0);
        private Set etapeses = new HashSet(0);

    // déclaration des constructeurs et méthodes
…
}
```

Figure 217 : Code de la classe processus

Dans le fichier Processus.hbm.xml, la balise class permet de mapper la classe com.monoprix.gdpadmin.DA.hibernate.classe.Processus à la table PROCESSUS

L'attribut idProcessus de cette classe correspond à la colonne ID_PROCESSUS de la table PROCESSUS. Une séquence SQ_PROCESSUS est utilisée pour générer idProcessus.

La balise property est utilisée pour définir le mapping entre les attributs libProcessus, gestionPerimetre versionUnique et les colonnes de la base LIB_PROCESSUS, GESTION_PERIMETRE et VERSION_UNIQUE. C'est une relation 1 pour 1.

Il est possible de définir des relations de type n pour 1 ou 1 pour n.

Les attributs sousCategorie et perimetre sont mis en correspondance avec respectivement les colonnes CODE_SOUS_CATEGORIE et ID_PERIMETRE.

Nous avons utilisé pour déclarer ces mappings la balise « many-to-one » qui permet d'indiquer qu'un ensemble de processus peut correspondre à un même périmètre ou à une même sous catégorie.

La balise set est employée pour indiquer que la classe possède une collection d'objets d'élément unique. La collection set ne permet pas d'avoir des doublons. Si un nouvel objet existe déjà, il ne sera pas ajouté à la collection.

Dans cet exemple, la classe processus comprend deux collections d'objets Version et Etapes. C'est la colonne ID_PROCESSUS qui est utilisée pour faire le lien en base de données. Nous avons également indiqué à l'aide de la balise one-to-many la cardinalité correspondante au lien.

Les classes Version et Etapes sont également mappées aux tables Oracle correspondantes à l'aide de fichiers possédant l'extension .hbm.xml.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 12 juin 2006 16:55:48 by Hibernate Tools 3.1.0.beta4 -->
<hibernate-mapping>
        <class name="com.monoprix.gdpadmin.DA.hibernate.classe.Processus"
                table="PROCESSUS">
                <id name="idProcessus" type="long">
                        <column name="ID_PROCESSUS" precision="22" scale="0"
                                sql-type="long" />
                        <generator class="sequence">
                                <param name="sequence">SQ_PROCESSUS</param>
                        </generator>
                </id>
                <property name="libProcessus" type="string">
                        <column name="LIB_PROCESSUS" length="100" />
                </property>
                 <property name="gestionPerimetre" type="string">
                        <column name="GESTION_PERIMETRE" length="1" />
                </property>
                <property name="versionUnique" type="string">
                        <column name="VERSION_UNIQUE" length="1" />
                </property>
                        many-to-one name="sousCategorie"  class="com.monoprix.gdpadmin.DA.hibernate.classe.SousCategorie"
                            fetch="join">
                            <column name="CODE_SOUS_CATEGORIE" length="5"
                                not-null="true" />
                </many-to-one>
                <many-to-one name="perimetre"
                        class="com.monoprix.gdpadmin.DA.hibernate.classe.Perimetre"
                        fetch="join">
                        <column name="ID_PERIMETRE" length="2" />
                </many-to-one>
                <set name="versions" inverse="true">
                        <key>
                                <column name="ID_PROCESSUS" precision="22" scale="0" not-null="true" />
                        </key>
                        <one-to-many
                                class="com.monoprix.gdpadmin.DA.hibernate.classe.Version" />
                </set>
                <set name="etapeses" inverse="true">
                        <key>
                                <column name="ID_PROCESSUS" precision="22" scale="0" not-null="true" />
                        </key>
                        <one-to-many
                                class="com.monoprix.gdpadmin.DA.hibernate.classe.Etapes" />
                </set>
                </class>
</hibernate-mapping>
```

Figure 218 : Processus.hbm.xml

La vue

Les saisies de l'administrateur ont donc permis de créer un enregistrement processus en base de données. Nous allons maintenant revenir sur la vue processus.jsp et voir comment elle a été mise en œuvre.

Le framework Tiles fournit un template ou modèle qui utilise des étiquettes personnalisées JSP pour décrire l'aspect graphique d'une page. Le modèle permet de définir l'apparence des différentes pages d'une application WEB, sans spécifier le contenu. Le contenu est inséré dans le modèle lors de l'exécution. Plusieurs pages peuvent utiliser le même modèle. La présentation du site est ainsi plus uniforme.

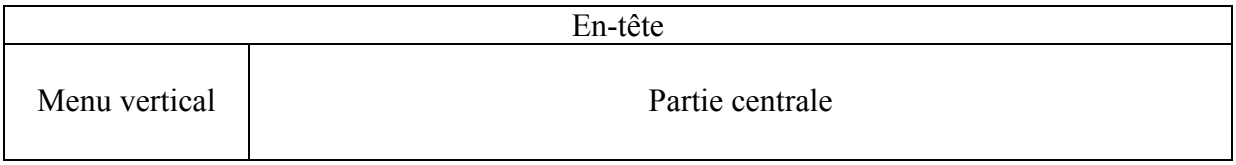


Figure 219 : Modèle de page utilisé dans le site

Le modèle utilise des balises « tiles-insert » pour spécifier les différentes zones constitutives des pages *(Figure 220)*.

```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>
<%@ taglib uri="/WEB-INF/tld/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/tld/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/tld/struts-logic.tld" prefix="logic" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

<html>
 <head>
  <script type="text/javascript" language="javascript" src="<%=request.getContextPath()%><bean:message key="chemin.script"/>menu.js"></script>
  <script type="text/javascript" language="javascript" src="<%=request.getContextPath()%><bean:message key="chemin.script"/>sortable.js"></script>
   <script type="text/javascript" language="javascript" src="<%=request.getContextPath()%><bean:message key="chemin.script"/>popup.js"></script>

  <meta http-equiv=Content-Type content="text/html; charset=iso-8859-1" >
  <link href="<%=request.getContextPath()%><bean:message key="chemin.style"/>accueil.css" rel="stylesheet" >
  <link href="<%=request.getContextPath()%><bean:message key="chemin.style"/>styleMonopTop.css" rel="stylesheet" >
  <title><bean:message key="texte.titre.application" /></title>
  <tiles:insert attribute="javascript" />
 </head>
<body leftmargin="0" topmargin="0" <tiles:insert attribute="onload" ignore="true"/>>

  <table border="0" width="100%" height="100%" cellspacing="0" cellpadding="0" >
   <tbody>
    <tr>
     <td colspan="2"><tiles:insert attribute="header" /></td>
    </tr>
    <tr height="100%">
     <td width="180px"><tiles:insert attribute="menu" /></td>
     <td width="90%" valign="top"><tiles:insert attribute="body" /></td>
    </tr>
   </tbody>
  </table>
 </body>
</html>
```

Figure 220 : Template.jsp

A l'aide de la balise « tiles:insert », nous précisons le modèle que nous allons utiliser.

La balise « tiles:put » permet ensuite d'insérer :

- des sources javascript
- un en-tête
- un menu vertical
- du code javascript dans la page
- le corps du document

Ci-dessous, la partie de la page processus.jsp correspondant à la déclaration du formulaire de création des processus *(Figure 221)*.

```
<%@ taglib uri="/WEB-INF/tld/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tld/struts-html-el.tld" prefix="html-el"%>
<%@ taglib uri="/WEB-INF/tld/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/tld/struts-logic.tld" prefix="logic"%>
<%@ taglib uri="/WEB-INF/tld/struts-tiles.tld" prefix="tiles"%>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c"%>
<%@ page import="com.monoprix.gdpadmin.Constantes"%>
<tiles:insert page="/jsp/common/template.jsp" flush="true">
 <tiles:put name="javascript" type="string">
  <script language=javascript src="<%=request.getContextPath()%><bean:message key="chemin.script"/>commun.js"></script>
  <script language=javascript src="<%=request.getContextPath()%><bean:message key="chemin.script"/>popup.js"></script>
  <script language=javascript src="<%=request.getContextPath()%><bean:message key="chemin.script"/>menu.js"></script>
  <script language=javascript src="<%=request.getContextPath()%><bean:message key="chemin.script"/>formulaire.js"></script>
  <script language=javascript src="<%=request.getContextPath()%><bean:message key="chemin.script"/>controles.js"></script>
 </tiles:put>
 <tiles:put name="header" value="/jsp/common/headerGestionProcessus.jsp" />
 <tiles:put name="menu" value="/jsp/common/menuvertical.jsp" />
 <tiles:put name="javascript" type="string">
  <script language="JavaScript">
   function verif()
   {
            // function javascript permettant de véification des saisis
}
 </script>
 </tiles:put>
 <tiles:put name="body" type="string">
  <logic:notPresent name="user" scope="session">
   <jsp:forward page="/identification.do" />
  </logic:notPresent>
  <html:form action="CreationProcessus" method="post">
   <fieldset style="margin:20px;"><legend><bean:message key="texte.processus.creation.titre" /></legend>
   <table border="0" width="100%">
    <tr>
     <td width="15%"><bean:message key="texte.processus.creation.libelle" /></td>
     <td width="50%"><html:text property="libelleprocessus" size="60" styleClass="saisie" maxlength="60" /></td>
     <td width="25%"><html:errors property="libelle" /></td>
     <td width="10%"><br>         </td>
    </tr>
    <tr>
     <td><bean:message key="texte.processus.creation.categorie" /></td>
     <td><html:select name="processusForm" property="codecategorieprocessus"
onchange="document.processusForm.action='InitProcessusType.do';document.processusForm.submit();">
      <html:options collection="listecategorie" labelProperty="codeLibCategorie" property="codeCategorie" />
     </html:select></td>
     <td><bean:message key="texte.processus.creation.souscategorie" /></td>
     <td><html:select name="processusForm" property="codesouscategorieprocessus">
      <html:options collection="listesscategorie" labelProperty="codeLibSousCategorie" property="codeSousCategorie" />
     </html:select></td>
     <td></td>
    </tr>
    <tr>
     <td><bean:message key="texte.processus.creation.versionUnique" /></td>
     <td colspan="1"><html:select name="processusForm" property="versionUnique">
      <option value="N">N</option>
      <option value="O">O</option>
     </html:select></td>
    </tr>
    <tr>
     <td><bean:message key="texte.processus.creation.gestionPerimetre" /></td>
     <td colspan="1"><html:select name="processusForm" property="gestionPerimetre" onchange="verif();">
      <option value="N">N</option>
      <option value="O">O</option>
     </html:select></td>
    </tr>
    <tr>
     <td><bean:message key="texte.processus.creation.perimetre" /></td>
     <td colspan="2"><html:select name="processusForm" property="idperimetreprocessus" disabled="true">
      <option value="NR"><%=Constantes.CHAMPS_NR%></option>
      <html:options collection="listeperimetre" labelProperty="idLibPerimetre" property="idPerimetre" />
     </html:select></td>
    </tr>
    <tr>
     <td colspan="4" align="center"><html:submit property="btnSubmit" value="Valider" styleClass="btn_cmd" /></td>
    </tr>
   </table>
   </fieldset>
   </tbody>
  </table>
 </tiles:put>
</tiles:insert>
```

Figure 221 : Processus.jsp

Nous pouvons noter que Struts met à disposition de nombreuses balises afin de mettre en place les vues.

Les Tag-lib Struts ou balises personnalisées du framework Struts simplifient beaucoup le développement des pages JSP. Elles permettent également de les relier aux autres composants.

Il est plus rapide de débuguer des pages jsp construites à l'aide de ces balises plutôt qu'avec des scriplets. Ces balises possèdent des noms et des descriptions qui simplifient les tâches de résolution de problème.

Les balises Struts sont regroupées en quatre groupes :

- Html

  Ces balises permettent d'implémenter l'ensemble des balises HTML : champs texte, boutons …

- Logic

  Ces balises permettent de mettre en place la logique des vues : itération, condition …

- Bean

  Ces balises gèrent l'accès aux beans de l'application : page, request, session … Elles permettent également de manipuler les propriétés des beans.

- Nested

  Ces balises ajoutent une arborescence permettant d'accéder aux beans. Elles regroupent les trois premiers groupes de balises.

Framework log4j et Ant

Nous allons finir par la présentation des deux fichiers de configuration correspondant à log4j et Ant.

Le framework log4j se base sur le fichier de configuration log4j.properties. Ce fichier permet de configurer les caractéristiques des traces correspondant aux appels de log.debug() ou log.info() dans le code java.

Il permet de préciser les sorties utilisées telles que la sortie standard ou un fichier de log.

Il précise la forme des messages affichés. Dans le cas des fichiers, il permet également de préciser de quelle façon ils sont gérés. L'exemple présente la mise en place des fichiers de log rotatifs par taille.

Enfin, nous précisons le niveau de log désiré, ici DEBUG.

```
# Fichier de configuration pour tracer les appels du style log.debug(...) ou
# log.info(...). Il faut que le projet ait acces a la librairie log4j-1.2.8.jar
#
# Pour avoir le niveau de trace INFORMATION, positionner 'info'
#		exemple : log4j.rootLogger=info, R
#
# Pour avoir le niveau de trace DEBUG, positionner 'debug'
#		exemple : log4j.rootLogger=debug, R
#
# Hierarchie des niveaux : FATAL < ERROR < WARN < INFO < DEBUG
################################################################################
# Utilisation de 2 logger : sortie standard (stdout) et fichier de log (R)
log4j.rootLogger=ERROR, stdout, R
################################################################################
# Definition du logger stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
# Pattern to output the caller's file name and line number.
log4j.appender.stdout.layout.ConversionPattern=%d %5p - %F:%L - %m%n
################################################################################
# Definition du logger R (fichier de log rotatif par taille)
log4j.appender.R=org.apache.log4j.RollingFileAppender
# Fichier
log4j.appender.R.File=${catalina.home}/logs/visuadmin.log
# Taille du fichier
log4j.appender.R.MaxFileSize=1000KB
# Nombre de fichier backup à conserver
log4j.appender.R.MaxBackupIndex=1000
# Layout
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d %5p - %F:%L - %m%n
################################################################################
# Definition des niveaux de log
# Log de l'application
log4j.logger.com.monoprix=DEBUG
```

Figure 222 : Fichier log4j.properties

Le framework Ant s'appuie sur le fichier de configuration build.xml *(Figure 223)*. Ce fichier comporte un ensemble de cibles (target) nommées :

- clean

  Elle permet de supprimer les classes compilées précédemment, ainsi que les jars correspondants.

- init

  Elle permet de créer l'arborescence nommée build qui va être utilisée lors de la construction de l'application. Et elle copie les différentes ressources qui vont être utilisées.

- compile

  Elle permet de compiler le code java dans le répertoire build. L'exécution de cette cible est conditionnée par la bonne réalisation de la cible init.

- war

  Elle permet la création de l'archive WEB java. Elle est conditionnée par les cibles init et compile.

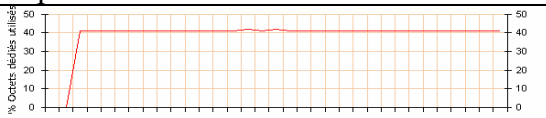
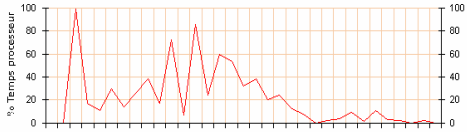
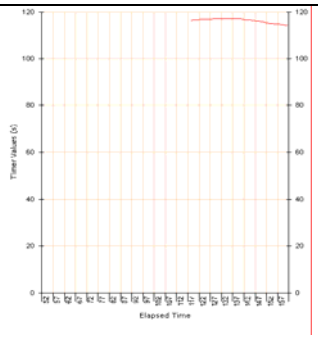
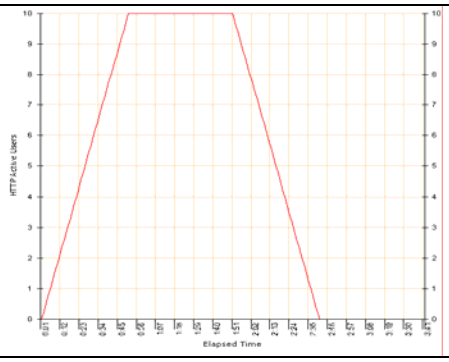
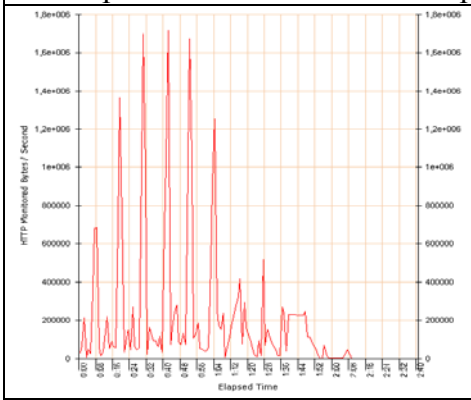
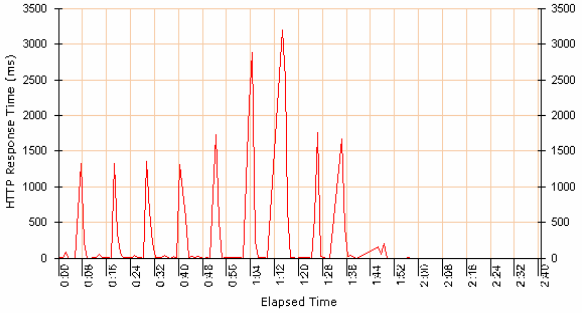
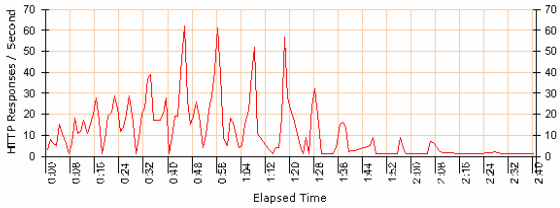
```
<?xml version="1.0" encoding="UTF-8"?>
<project name="gdpadmin" default="war">
<description> Génération de l'application GDPAdmin   </description>
<!-- Set global properties for this build -->
<property name="src.dir" value="./WEB-INF/src"/>
<property name="build.dir" value="./WEB-INF/buildir"/>
<property name="tomcat.lib" value="d:/dev/Tomcat 5.0/common/lib"/>

<!-- target qui supprime les classes compilé et le JAR -->
<target name="clean">
        <delete dir="${build.dir}"/>
        <delete file="${ant.project.name}.war"/>
</target>

<target name="init">
<!-- création de l'arborescence du build-->
<!-- copie des differentes ressources utilisées -->
<copy todir="${build.dir}/WEB-INF/src">
        <fileset dir="${src.dir}"/>
</copy>
<copy todir="${build.dir}/jsp">
        <fileset dir="./jsp" />
</copy>
<copy todir="${build.dir}/WEB-INF/lib">
        <fileset dir="./WEB-INF/lib" />
</copy>
<copy todir="${build.dir}/WEB-INF/tld">
        <fileset dir="./WEB-INF/tld" />
</copy>
<copy todir="${build.dir}/WEB-INF/" includeEmptyDirs="no" >
        <fileset dir="./WEB-INF/" >
                <patternset>
                <include name="*.conf"/>
                <include name="**.properties"/>
                <include name="**.xml"/>
                </patternset>
        </fileset>
</copy>
</target>

<!-- target qui compile le code Java dans le répertoire build -->
<target name="compile" depends="init">
<!-- création d'un répertoire pour la compilation -->
        <mkdir dir="${build.dir}/WEB-INF/classes"/>

<!-- compilation des sources Java -->
        <javac srcdir="${build.dir}/WEB-INF/src" destdir="${build.dir}/WEB-INF/classes">
        <classpath>
        <!-- Include all jar files  -->
        <fileset dir="${build.dir}/WEB-INF/lib">
        <include name="*.jar"/>
        </fileset>
        <fileset dir="${tomcat.lib}">
        <include name="*.jar"/>
        </fileset>
        </classpath>
        </javac>
        <copy todir="${build.dir}/WEB-INF/classes" includeEmptyDirs="no" >
                <fileset dir="${build.dir}/WEB-INF/src" >
                <patternset>
                        <include name="*.conf"/>
                        <include name="**/*.properties"/>
                        <include name="**/*.xml"/>
                </patternset>
                </fileset>
        </copy>
</target>


<!-- target qui crée le WAR -->
<target name="war" depends="clean,compile">
        <war warfile="${ant.project.name}.war" webxml="${build.dir}/WEB-INF/web.xml" excludes="build.xml">
        <fileset dir="${build.dir}/" includes="**/*.*" excludes="build.xml"/>
        <classes dir="${build.dir}/WEB-INF/classes" includes="**/*.properties" />
        </war>
        <delete dir="${build.dir}"/>
        </target>
</project>
```

Figure 223 : Fichier build.xml

Annexe D : Résultats des tests de montée en charge

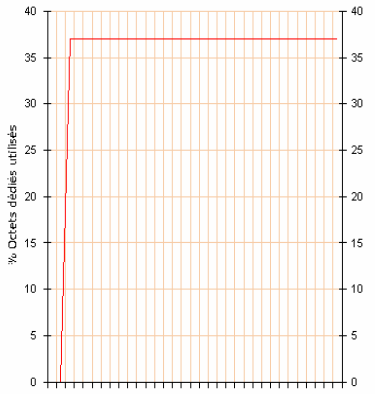
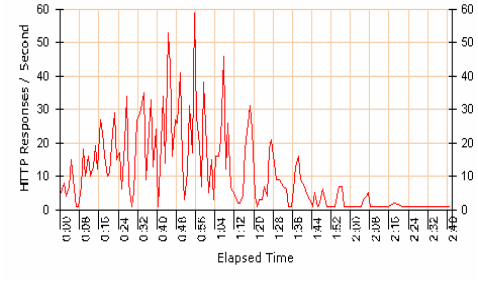
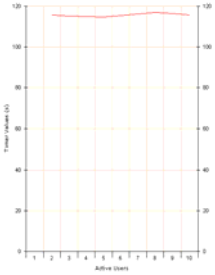
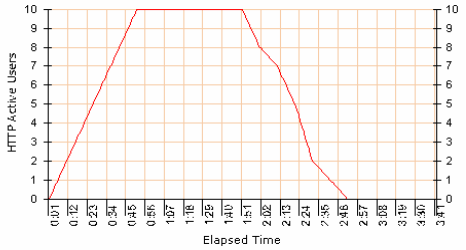
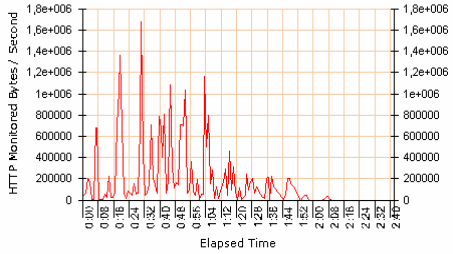
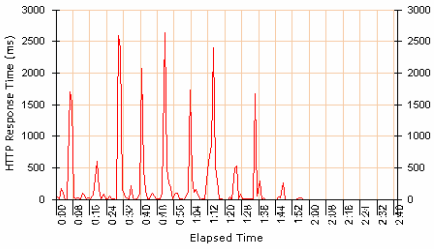
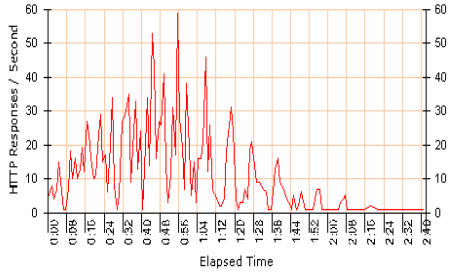
Test 1

Tableau 37 : Résultats obtenus pour le Test 1.

Occupation Ram serveur	Occupation CPU serveur
	
Durée du scénario en fonction du temps	**Nombre d'utilisateurs en fonction du temps**
	
Bande passante en fonction du temps	**Temps de réponse en fonction du temps**
	
Nombre de réponses en fonction du temps	
	

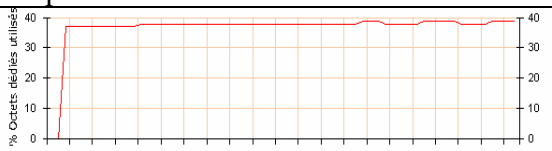
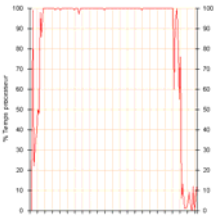
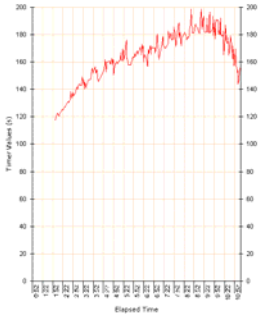
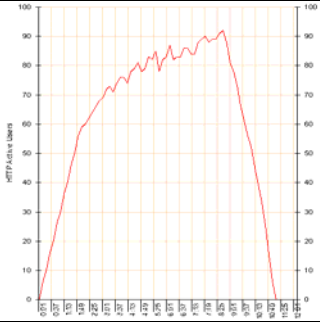
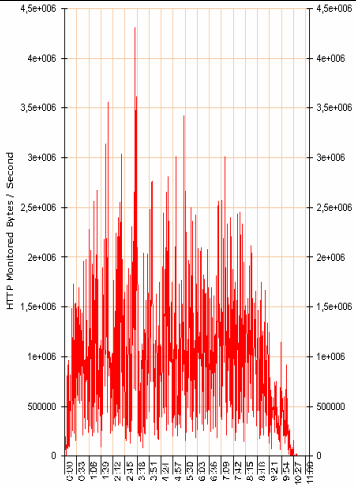
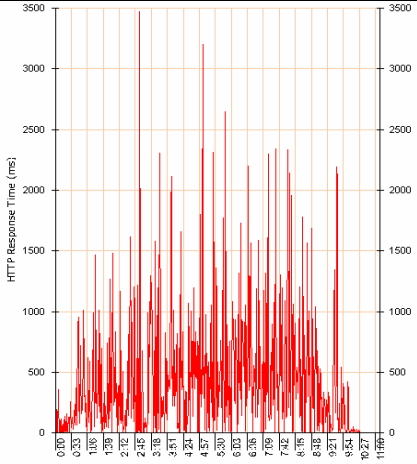
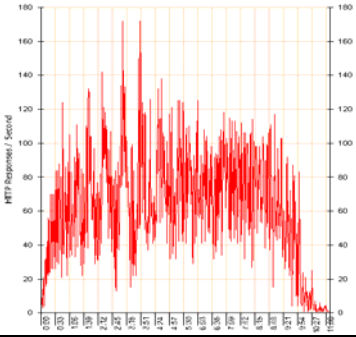
Test 2

Tableau 38 : Résultats obtenus pour le Test 2.

Occupation Ram serveur	Occupation CPU serveur
	
Durée du scénario en fonction du nombre d'utilisateurs actifs	Nombre d'utilisateurs en fonction du temps
	
Bande passante en fonction du temps	Temps de réponse en fonction du temps
	
Nombre de réponses en fonction du temps	
	

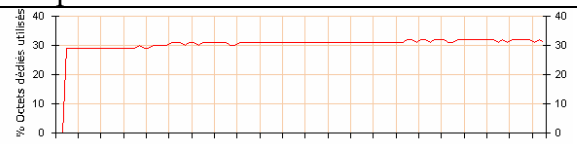
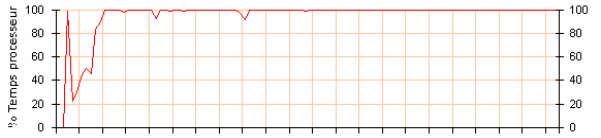
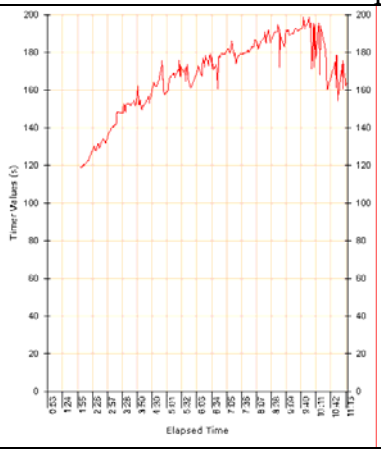
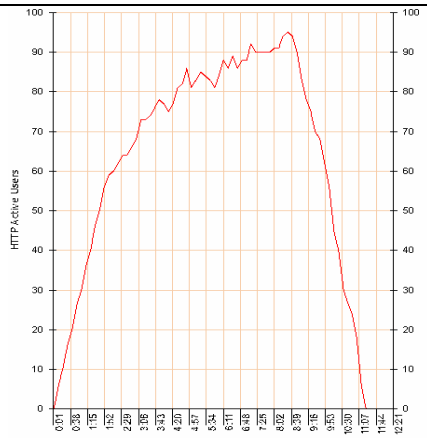
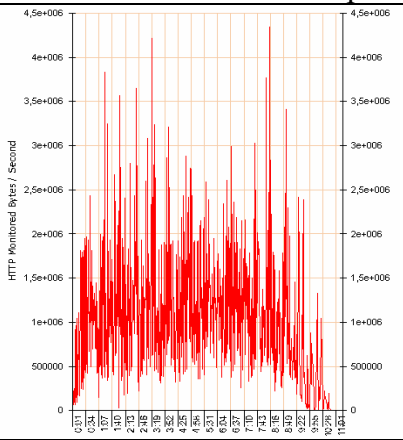
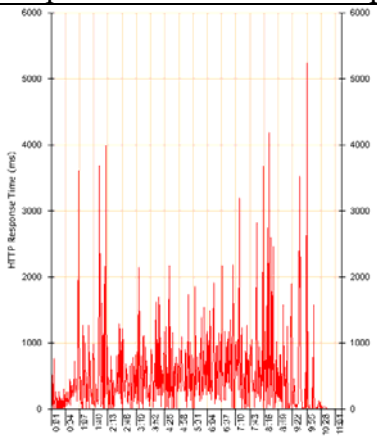
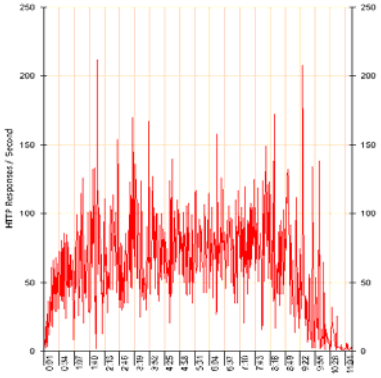
Test 3

Tableau 39 : Résultats obtenus pour le Test 3.

Occupation Ram serveur	Occupation CPU serveur
	
Durée du scénario en fonction du temps	**Nombre d'utilisateurs en fonction du temps**
	
Bande passante en fonction du temps	**Temps de réponse en fonction du temps**
	
Nombre de réponses en fonction du temps	
	

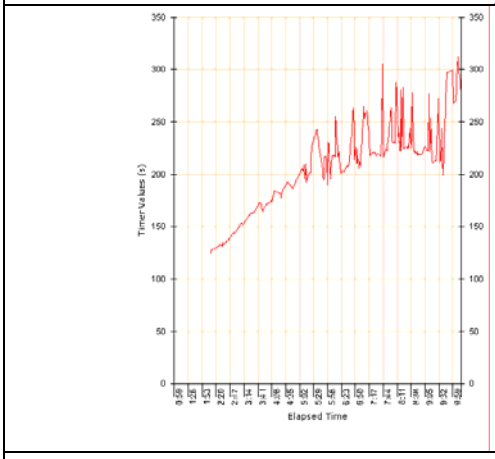
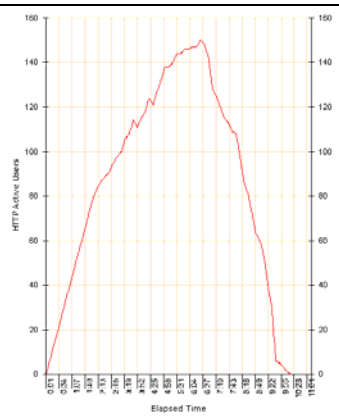
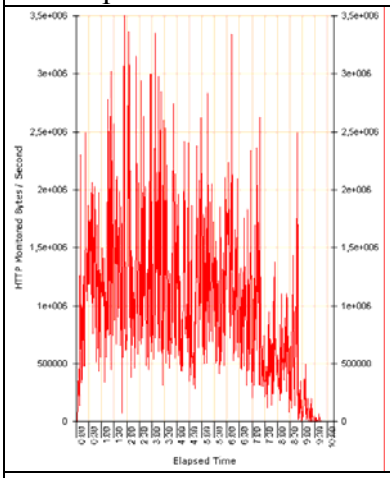
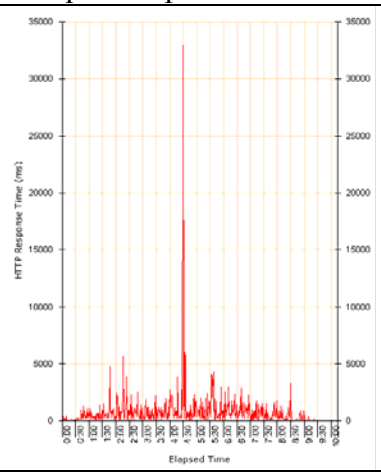
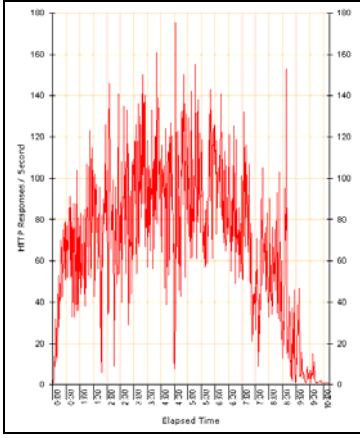
Test 4

Tableau 40: Résultats obtenus pour le Test 4.

Occupation Ram serveur	Occupation CPU serveur
	
Durée du scénario en fonction du temps	**Nombre d'utilisateurs en fonction du temps**
	
Bande passante en fonction du temps	**Temps de réponse en fonction du temps**
	
Nombre de réponses en fonction du temps	
	

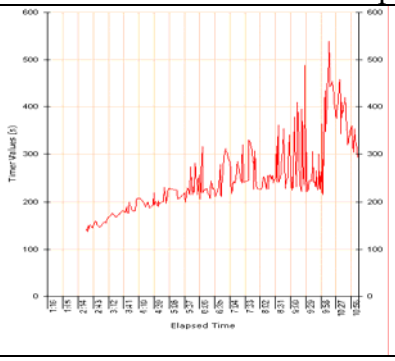
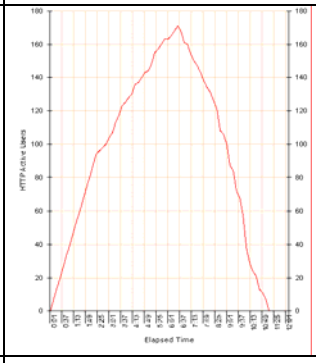
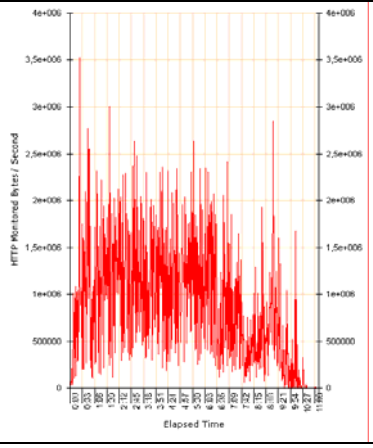
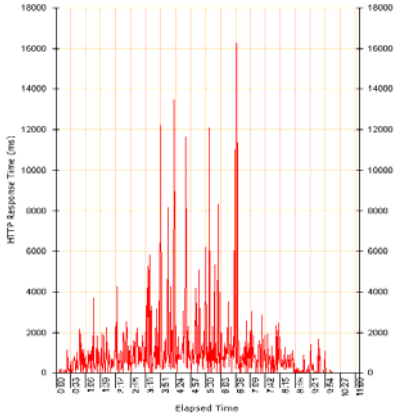
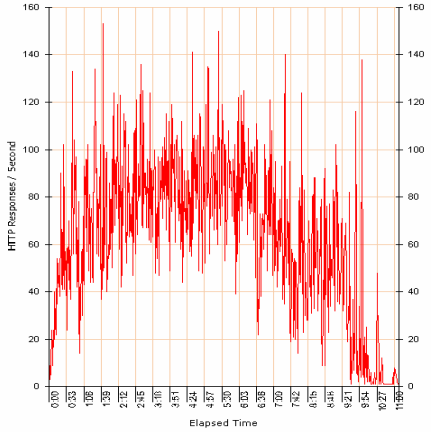
Test 5

Tableau 41 : Résultats obtenus pour le Test 5.

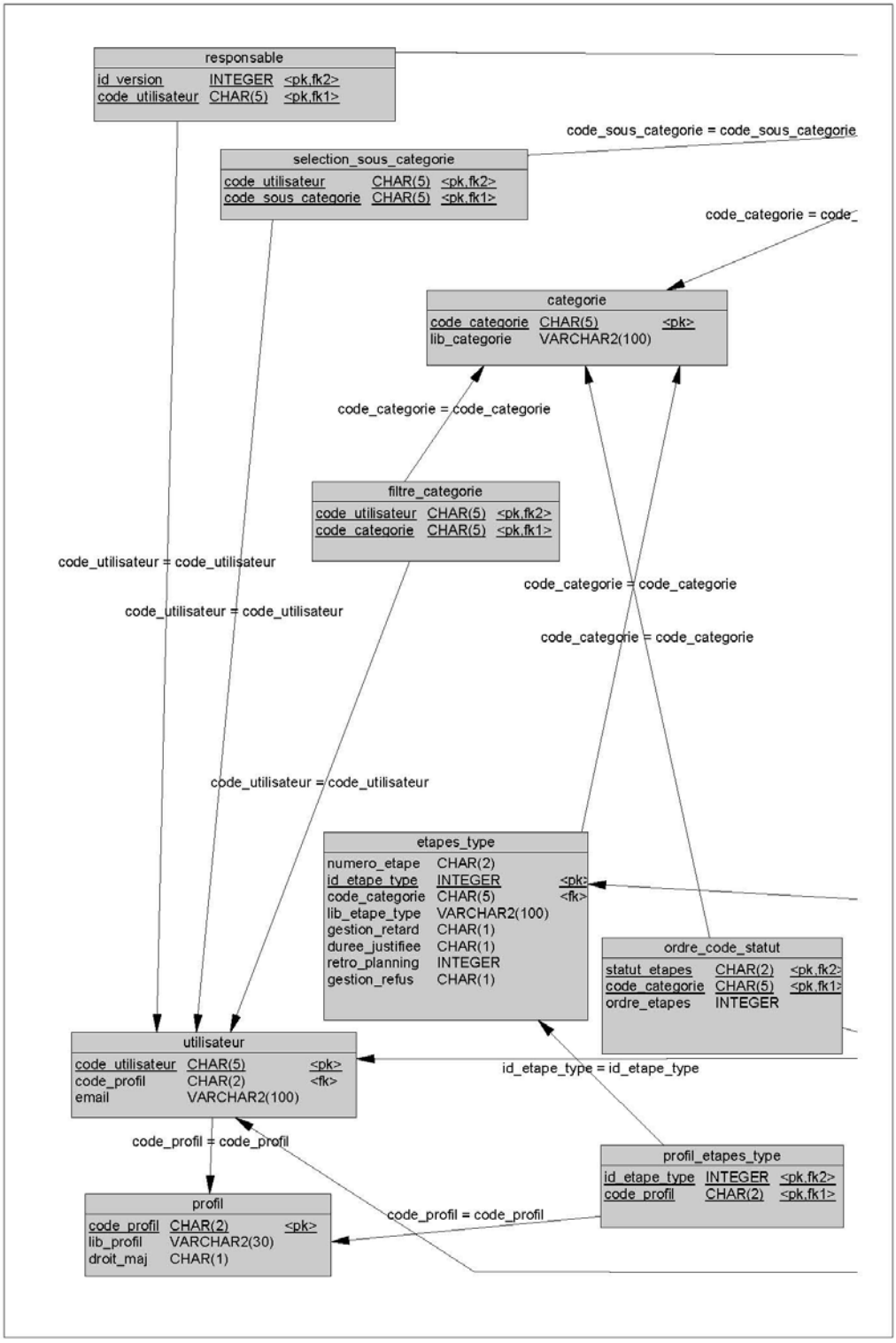
Durée du scénario en fonction du temps	Nombre d'utilisateurs en fonction du temps
	
Bande passante en fonction du temps	**Temps de réponse en fonction du temps**
	
Nombre de réponses en fonction du temps	
	

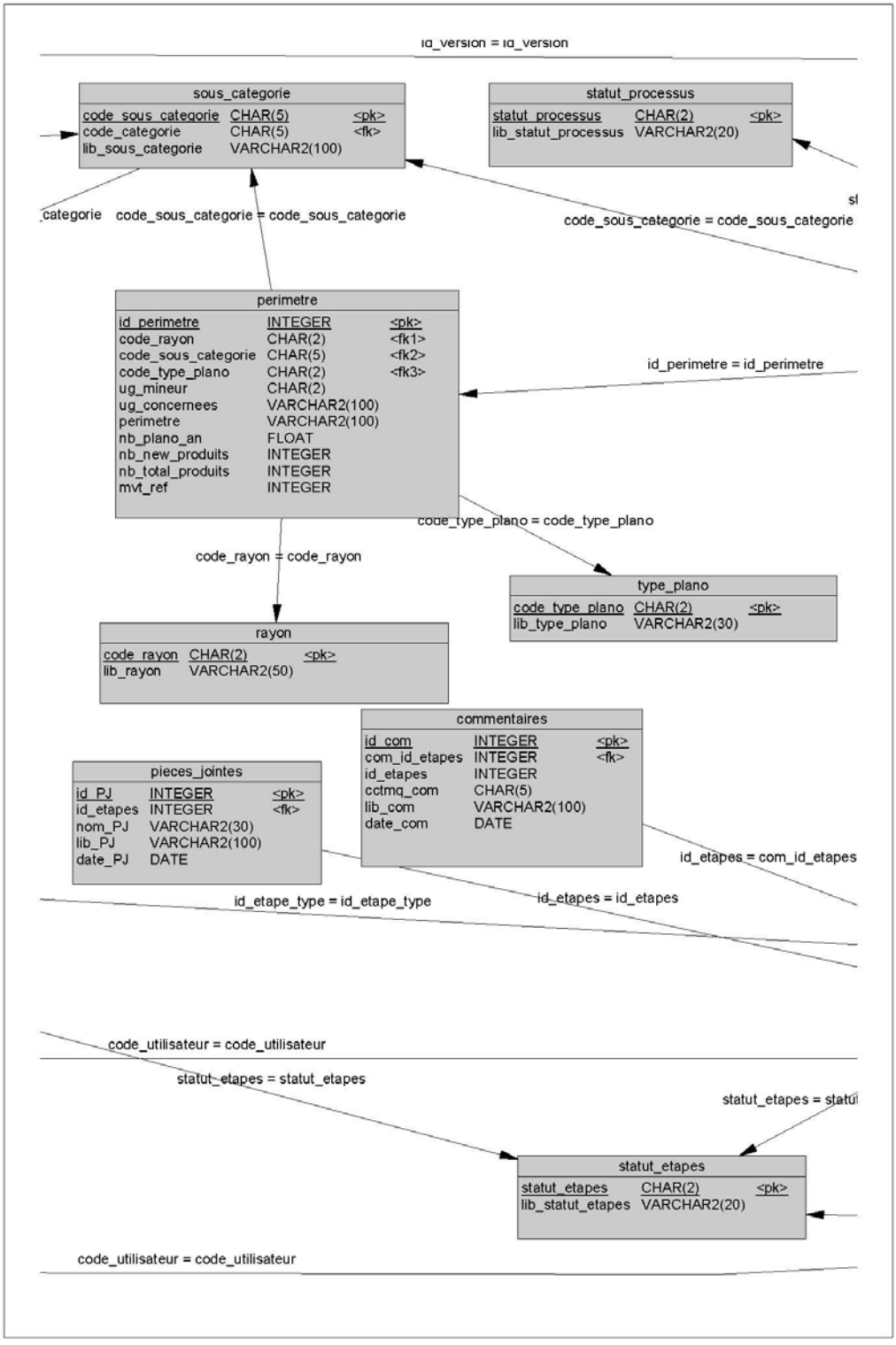
Test 6

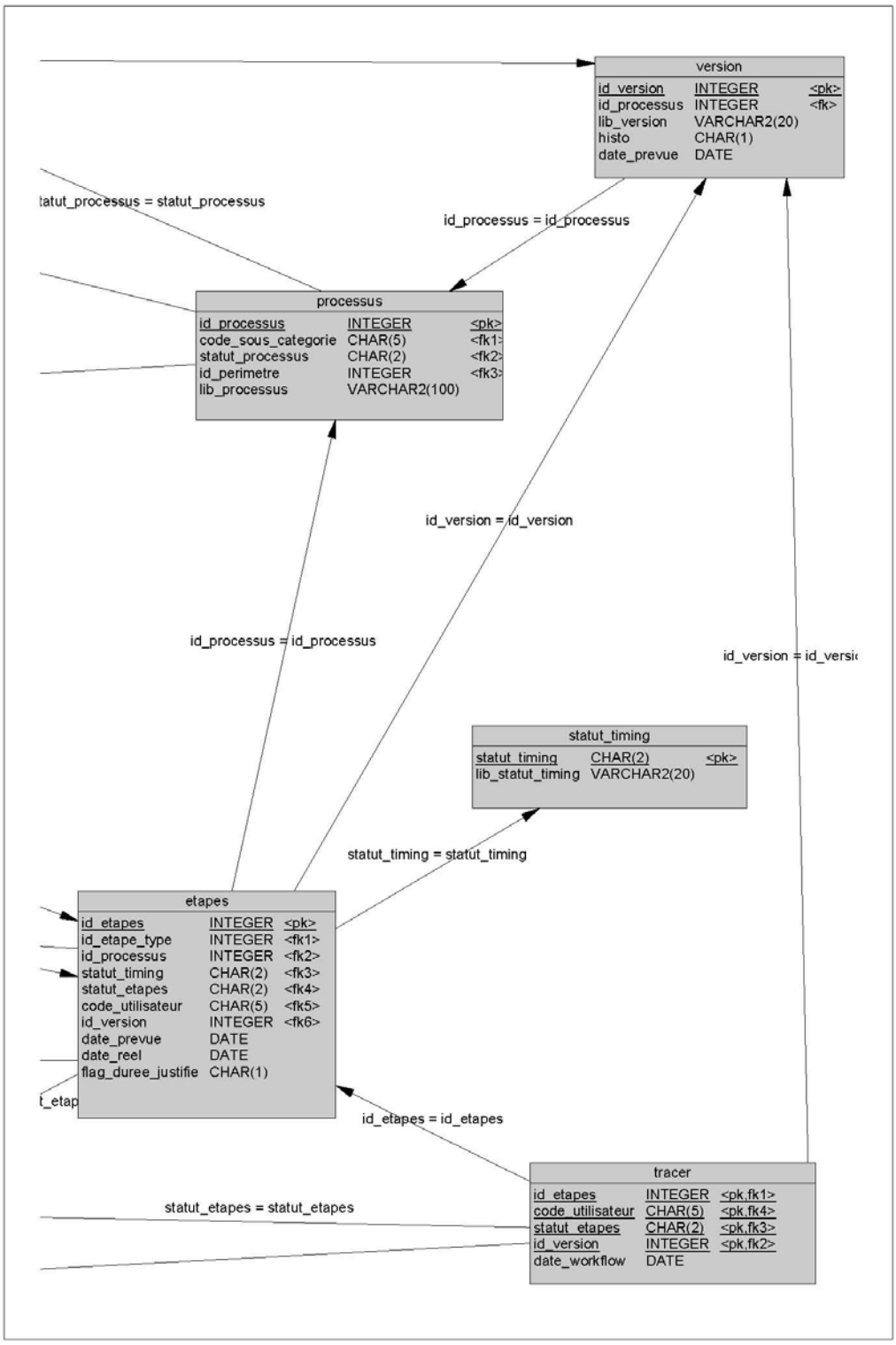
Tableau 42 : Résultats obtenus pour le Test 6.

Durée du scénario en fonction du temps	Nombre d'utilisateurs en fonction du temps
	
Bande passante en fonction du temps	**Temps de réponse en fonction du temps**
	
Nombre de réponses en fonction du temps	
	

Annexe F : MLD de GDP




id_version = id_version
sous_categorie
code_sous_categorie CHAR(5) <pk>
code_categorie CHAR(5) <fk>
lib_sous_categorie VARCHAR2(100)
statut_processus
statut_processus CHAR(2) <pk>
lib_statut_processus VARCHAR2(20)
categorie
code_sous_categorie = code_sous_categorie
code_sous_categorie = code_sous_categorie
perimetre
id_perimetre INTEGER <pk>
code_rayon CHAR(2) <fk1>
code_sous_categorie CHAR(5) <fk2>
code_type_plano CHAR(2) <fk3>
ug_mineur CHAR(2)
ug_concernees VARCHAR2(100)
perimetre VARCHAR2(100)
nb_plano_an FLOAT
nb_new_produits INTEGER
nb_total_produits INTEGER
mvt_ref INTEGER
id_perimetre = id_perimetre
code_type_plano = code_type_plano
code_rayon = code_rayon
type_plano
code_type_plano CHAR(2) <pk>
lib_type_plano VARCHAR2(30)
rayon
code_rayon CHAR(2) <pk>
lib_rayon VARCHAR2(50)
commentaires
id_com INTEGER <pk>
com_id_etapes INTEGER <fk>
id_etapes INTEGER
cctmq_com CHAR(5)
lib_com VARCHAR2(100)
date_com DATE
pieces_jointes
id_PJ INTEGER <pk>
id_etapes INTEGER <fk>
nom_PJ VARCHAR2(30)
lib_PJ VARCHAR2(100)
date_PJ DATE
id_etapes = com_id_etapes
id_etapes = id_etapes
id_etape_type = id_etape_type
code_utilisateur = code_utilisateur
statut_etapes = statut_etapes
statut_etapes = statut
statut_etapes
statut_etapes CHAR(2) <pk>
lib_statut_etapes VARCHAR2(20)
code_utilisateur = code_utilisateur

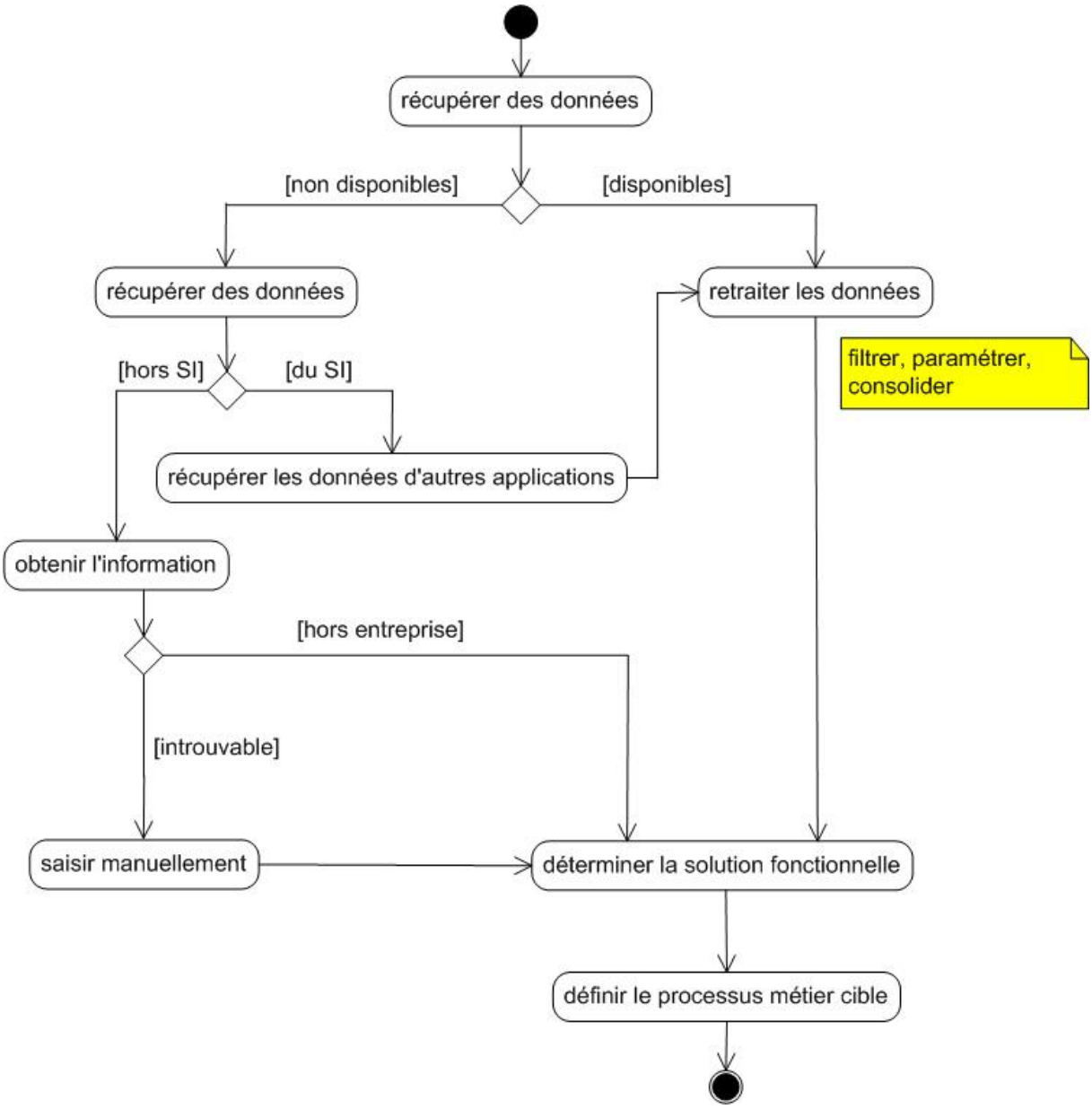

version
id_version INTEGER <pk>
id_processus INTEGER <fk>
lib_version VARCHAR2(20)
histo CHAR(1)
date_prevue DATE
tatut_processus = statut_processus
id_processus = id_processus
processus
id_processus INTEGER <pk>
code_sous_categorie CHAR(5) <fk1>
statut_processus CHAR(2) <fk2>
id_perimetre INTEGER <fk3>
lib_processus VARCHAR2(100)
id_version = id_version
id_processus = id_processus
id_version = id_versi
statut_timing
statut_timing CHAR(2) <pk>
lib_statut_timing VARCHAR2(20)
statut_timing = statut_timing
etapes
id_etapes INTEGER <pk>
id_etape_type INTEGER <fk1>
id_processus INTEGER <fk2>
statut_timing CHAR(2) <fk3>
statut_etapes CHAR(2) <fk4>
code_utilisateur CHAR(5) <fk5>
id_version INTEGER <fk6>
date_prevue DATE
date_reel DATE
flag_duree_justifie CHAR(1)
t_etap
id_etapes = id_etapes
tracer
id_etapes INTEGER <pk,fk1>
code_utilisateur CHAR(5) <pk,fk4>
statut_etapes CHAR(2) <pk,fk3>
id_version INTEGER <pk,fk2>
date_workflow DATE
statut_etapes = statut_etapes

Annexe G : Versions des logiciels

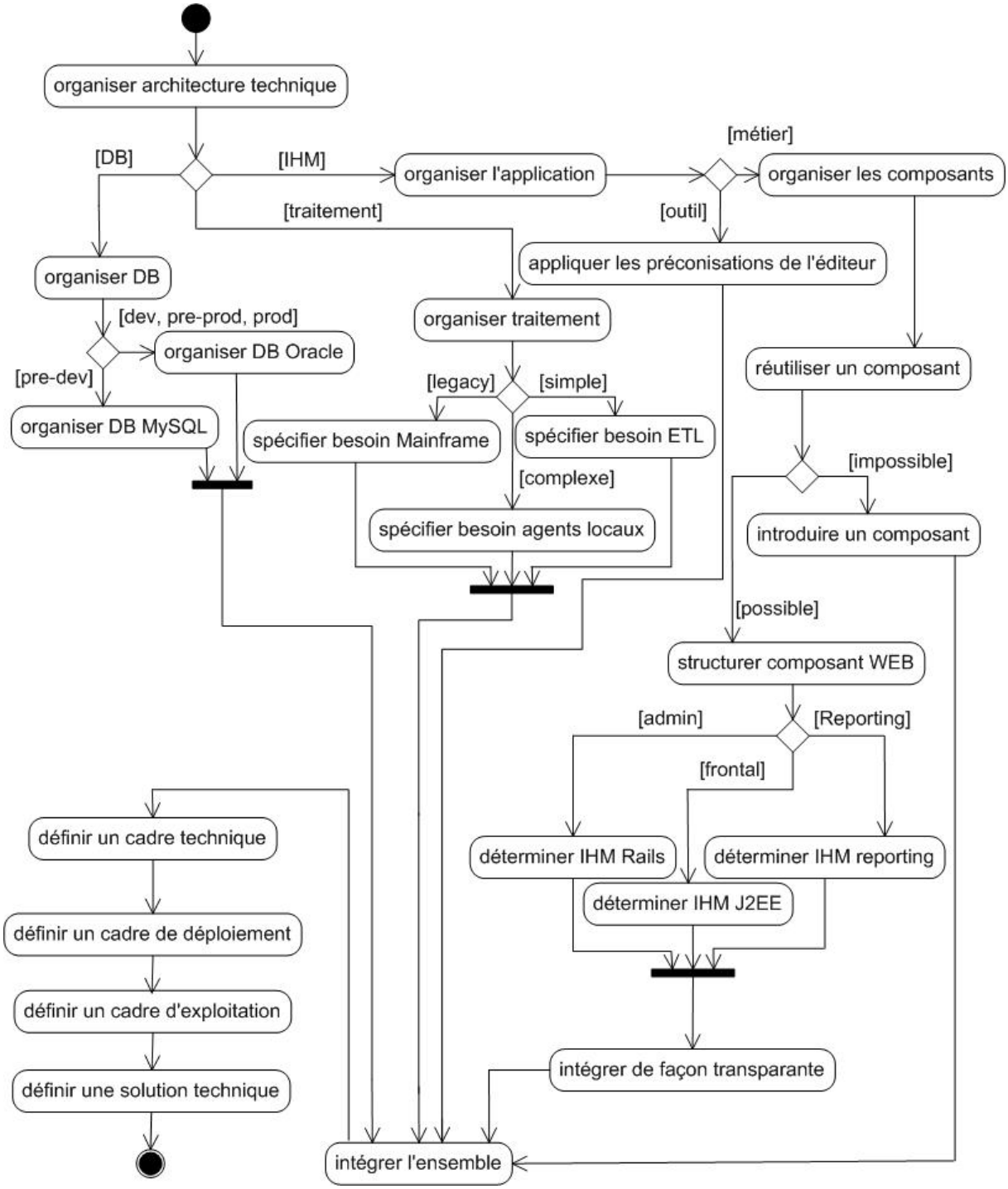
- Apache 2.0.49
- GanttProject 2.0.4
- GENIO 4.6
- JdK 1.4.2_09 et 1.5
- MySQL 5.0.37
- Oracle 8.7.1 et 9.2
- Oracle Raptor 1.0.0.07
- PowerAmc 8
- Powercenter 7.1.4
- SQL-view 1.9
- Toad 8.5
- Tomcat 5.0.28
- VBIS 7 .3
- Microsoft visio 2003

Annexe R : Processus du projet – diagrammes d'activité

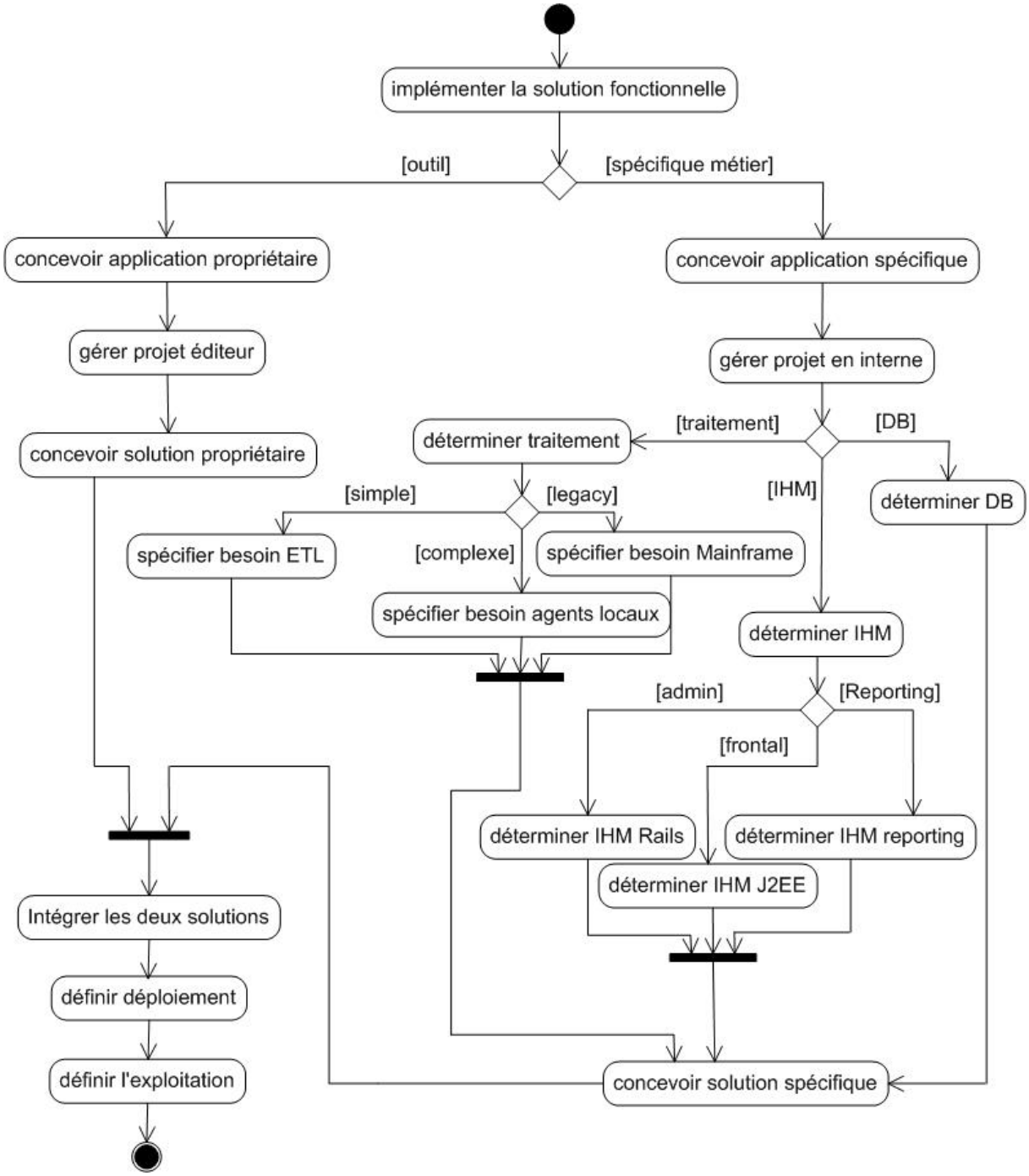
Analyse



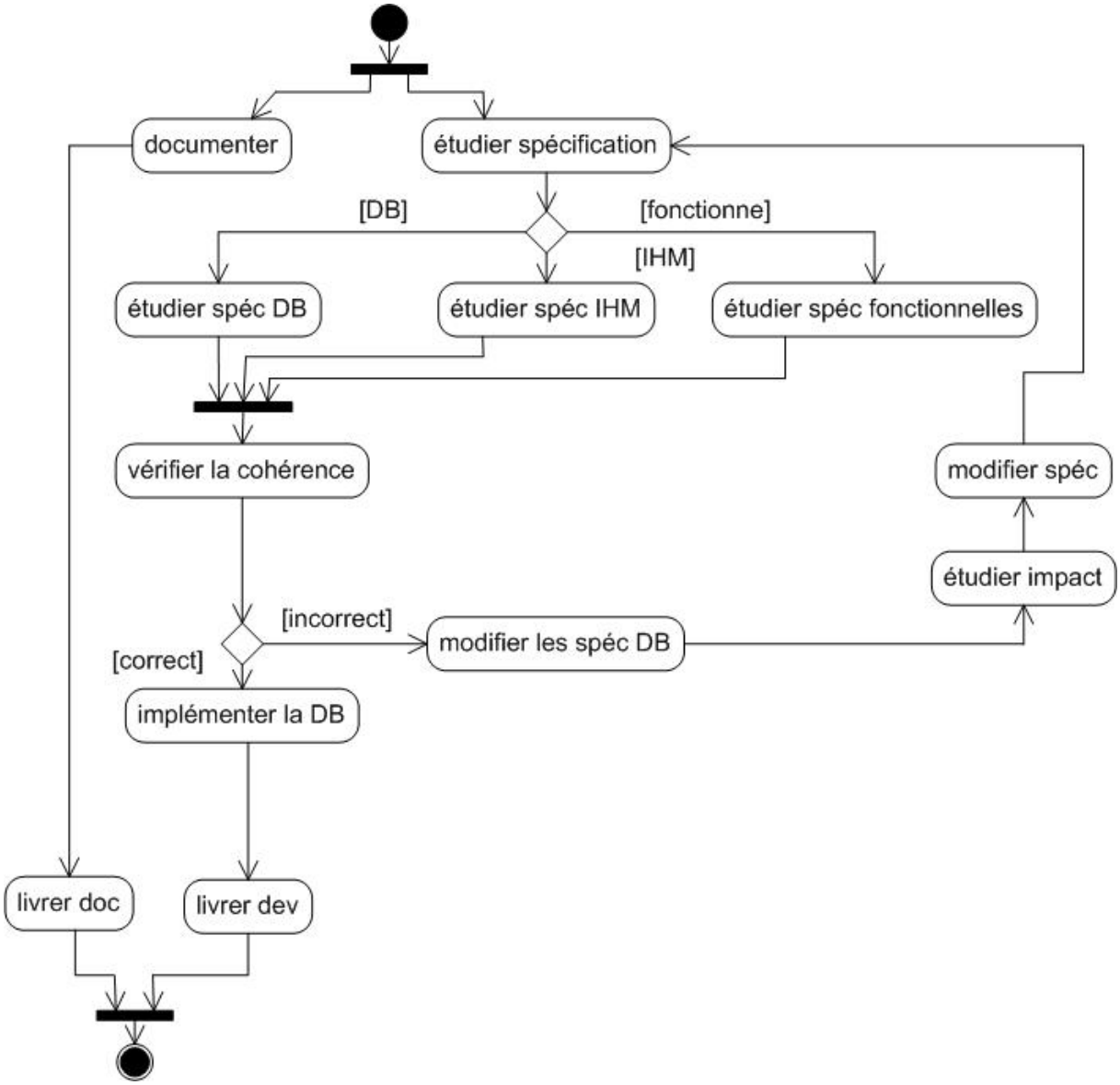
Conception générique



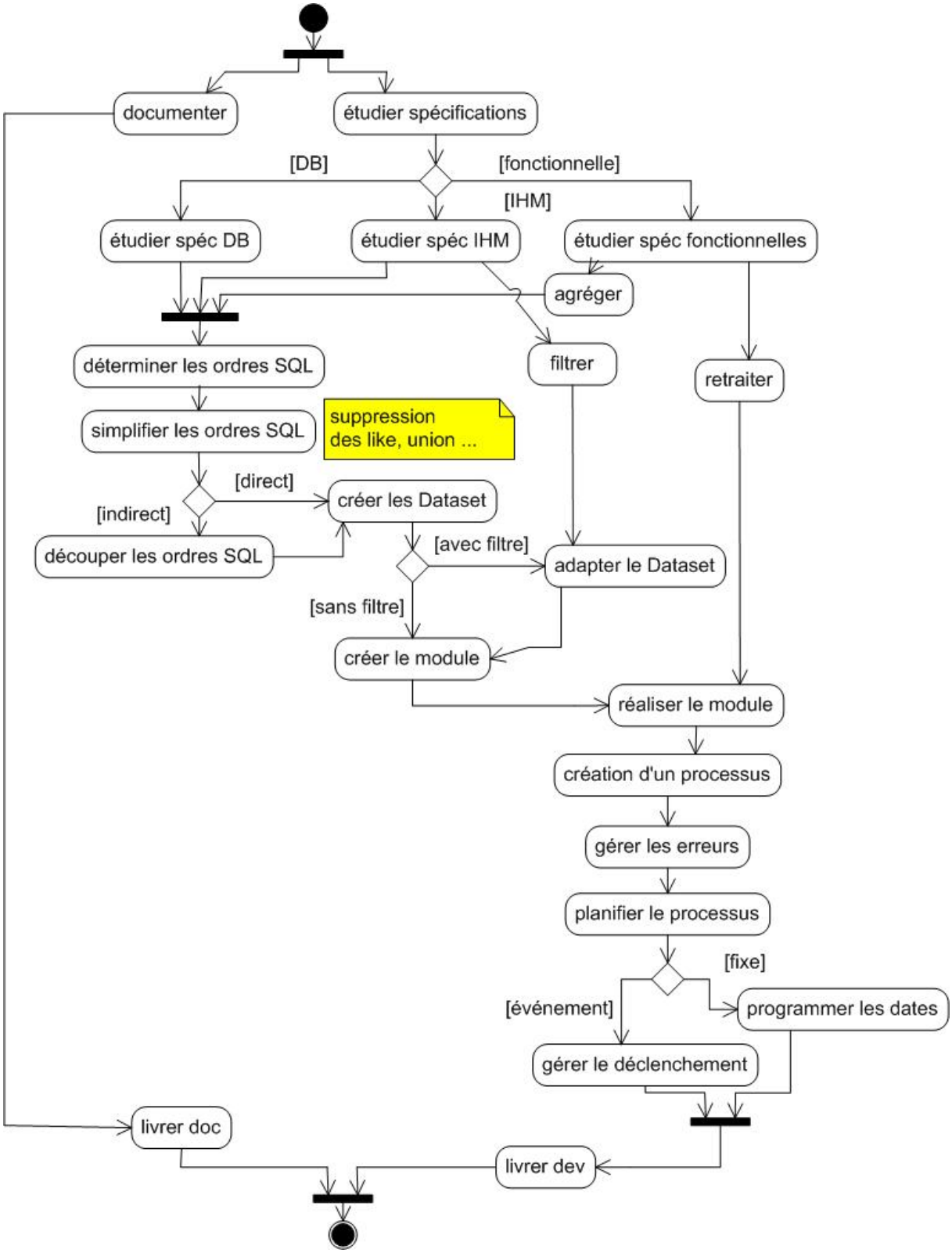
Conception préliminaire



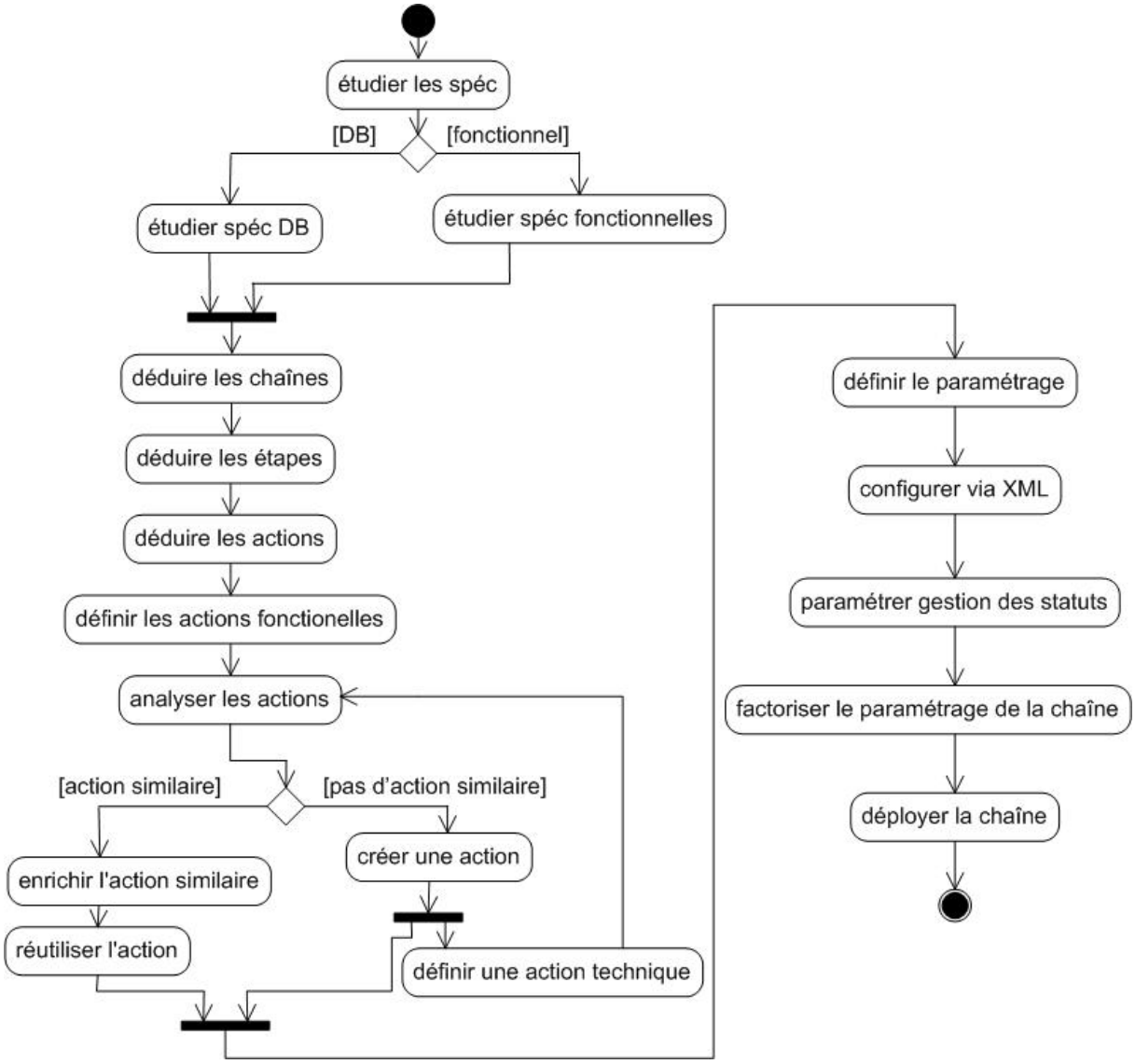
Développement base de données



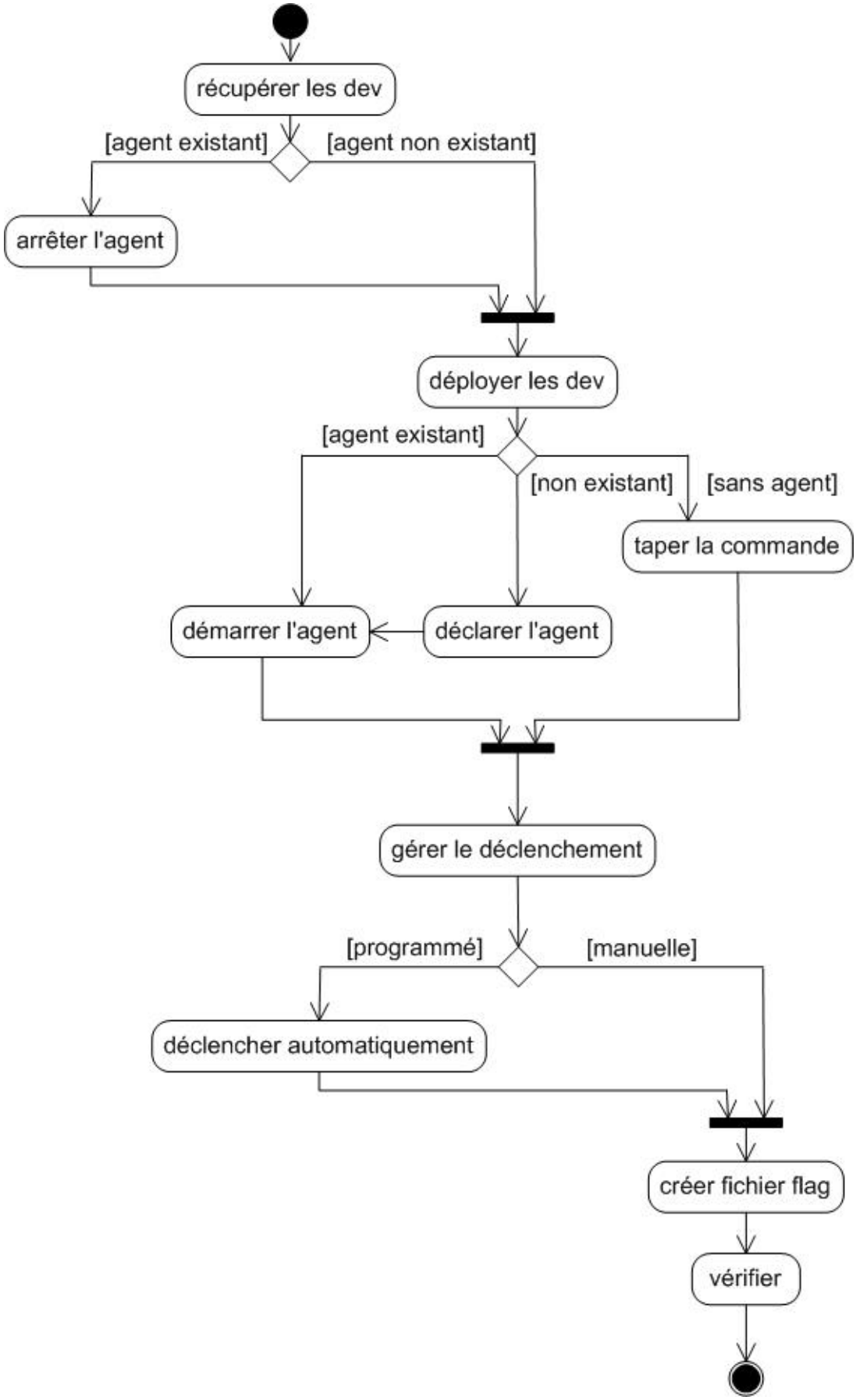
Développement ETL

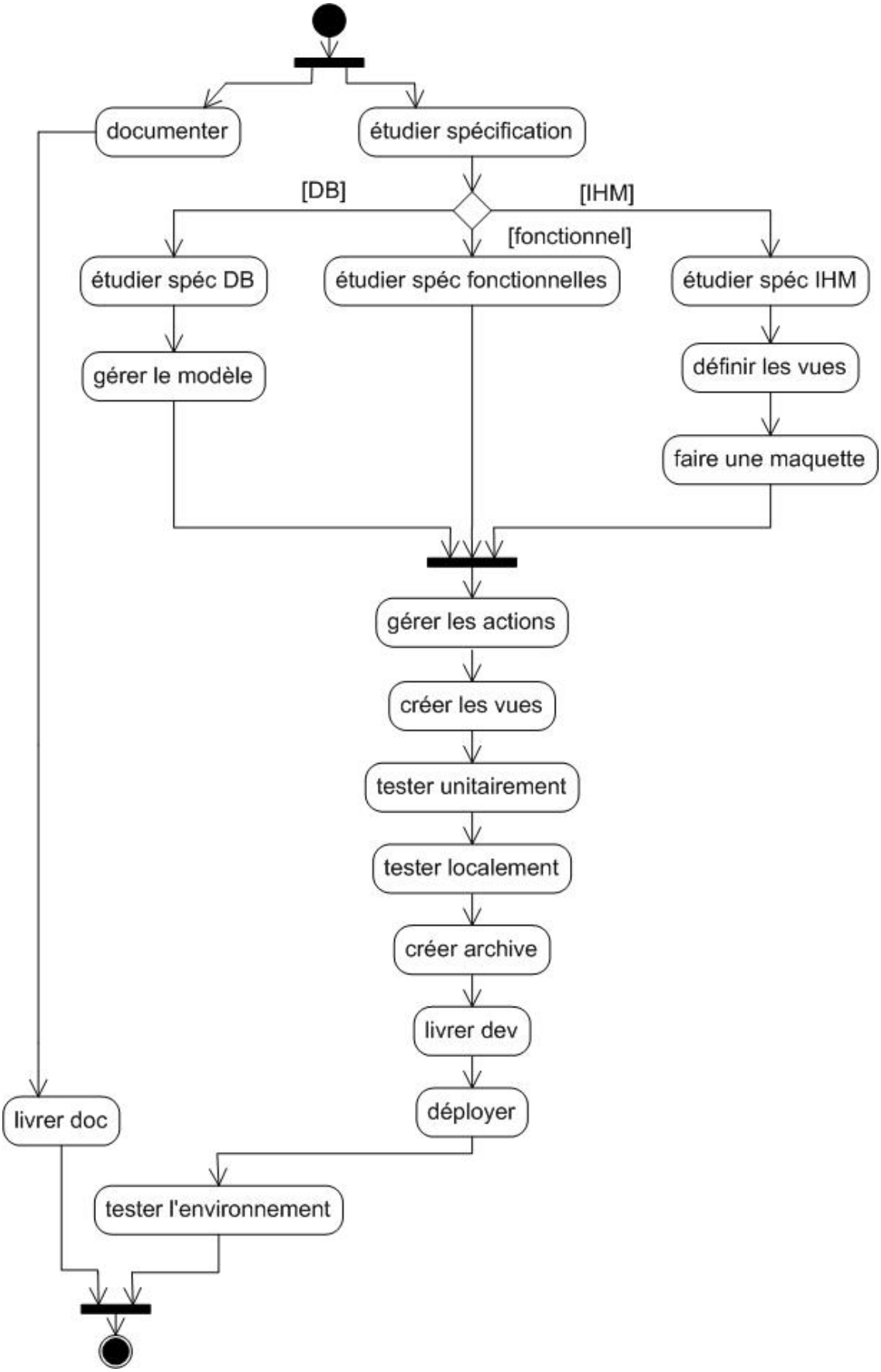


Développement java framework spécifique

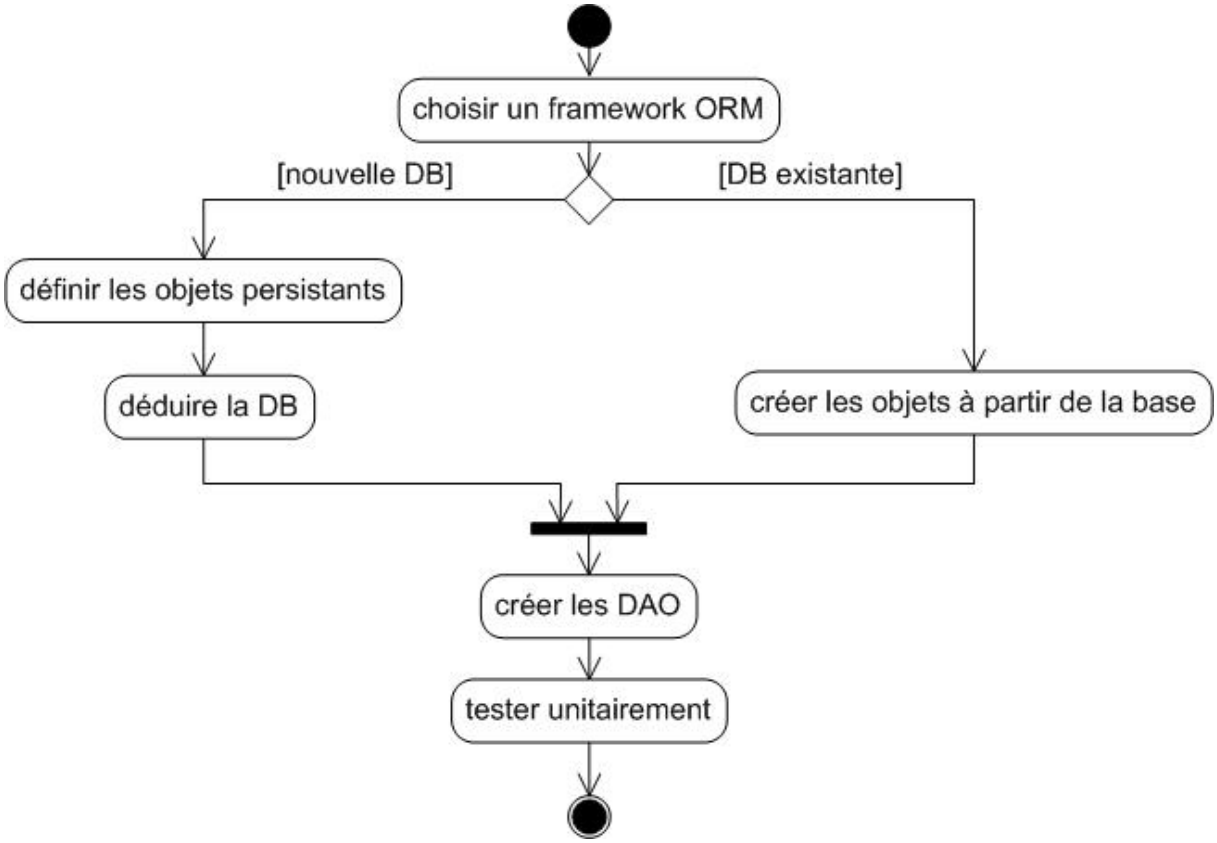


Développement java framework spécifique – déploiement

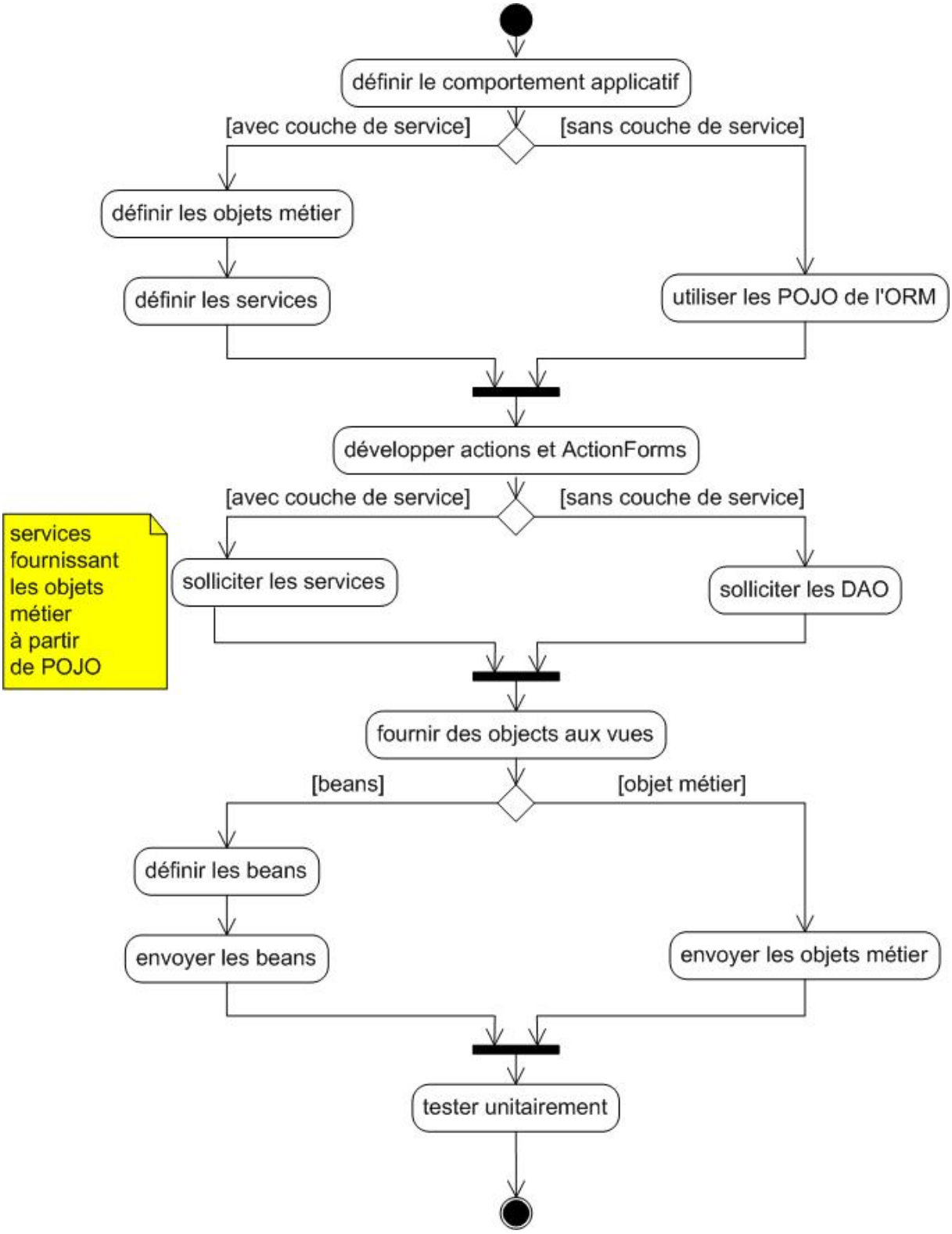


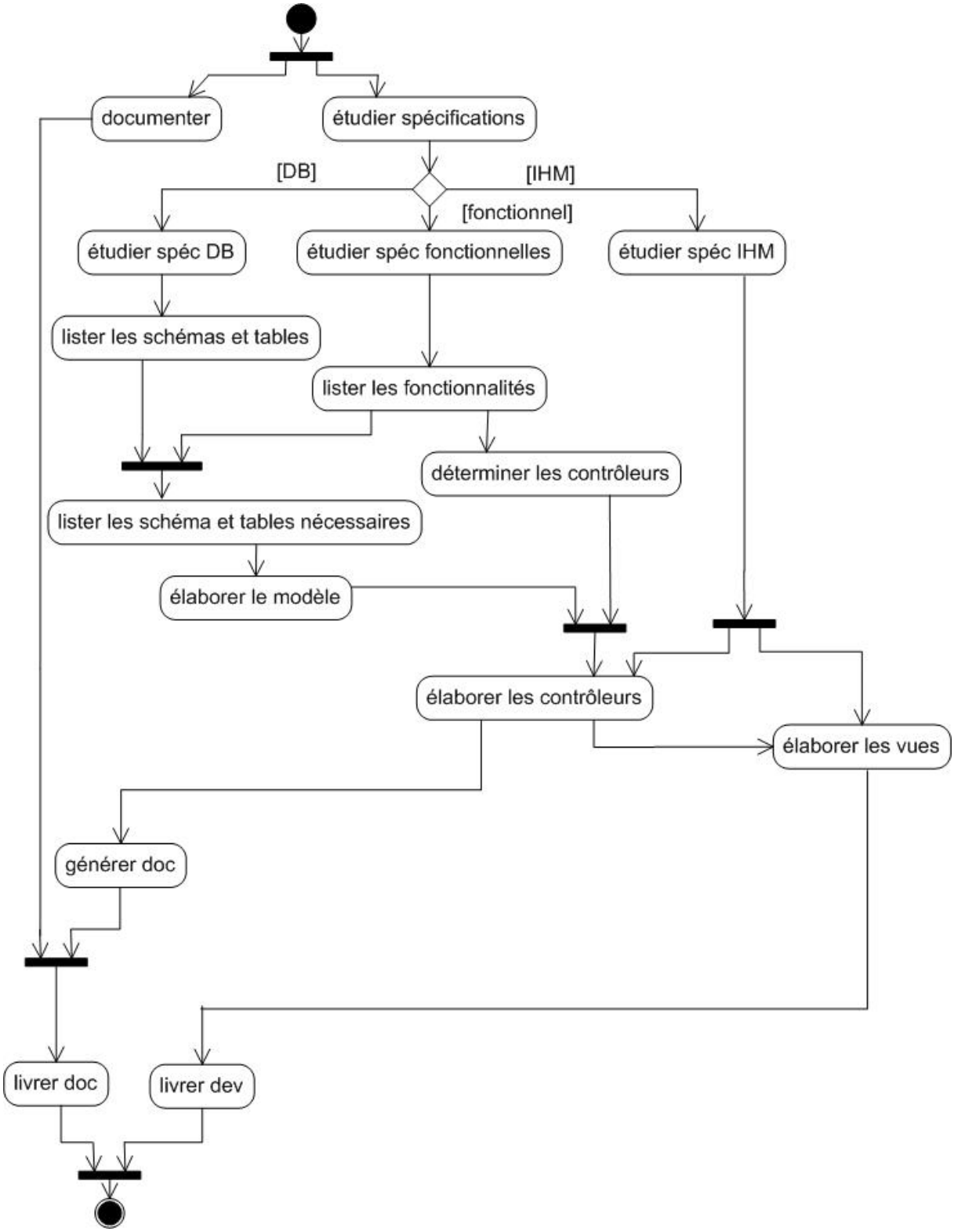

documenter
étudier spécification
[DB]
[IHM]
[fonctionnel]
étudier spéc DB
étudier spéc fonctionnelles
étudier spéc IHM
gérer le modèle
définir les vues
faire une maquette
gérer les actions
créer les vues
tester unitairement
tester localement
créer archive
livrer dev
déployer
livrer doc
tester l'environnement

Développement J2EE – modèle

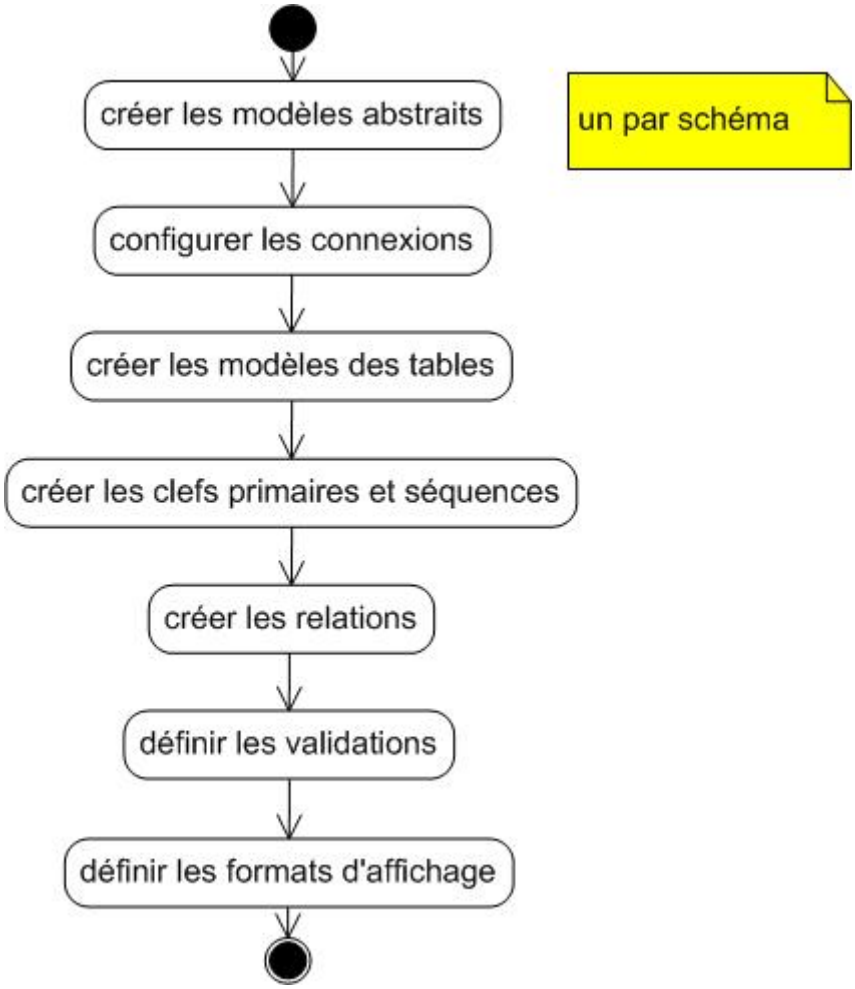


Développement J2EE – actions

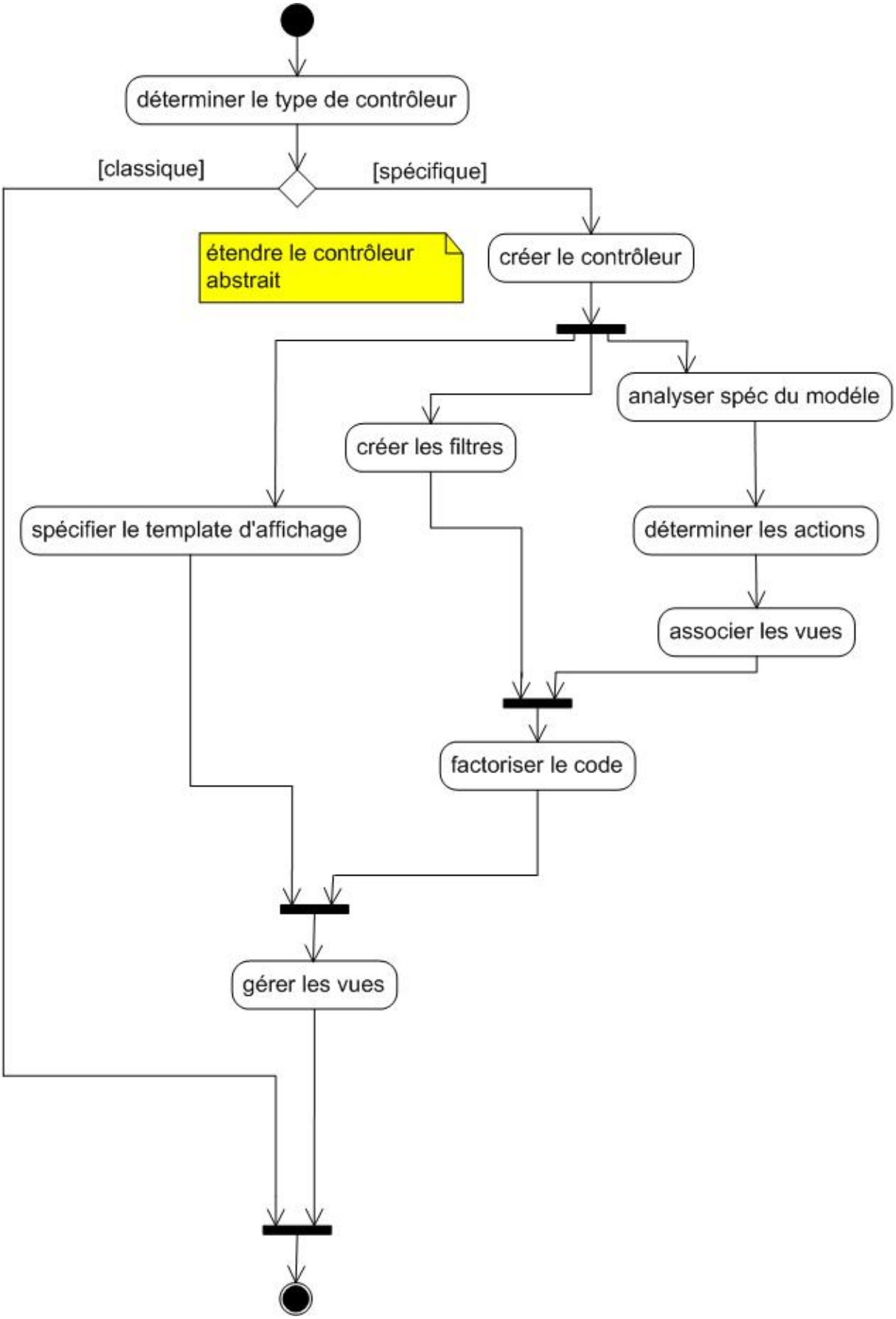



documenter
étudier spécifications
[DB]
[IHM]
[fonctionnel]
étudier spéc DB
étudier spéc fonctionnelles
étudier spéc IHM
lister les schémas et tables
lister les fonctionnalités
déterminer les contrôleurs
lister les schéma et tables nécessaires
élaborer le modèle
élaborer les contrôleurs
élaborer les vues
générer doc
livrer doc
livrer dev

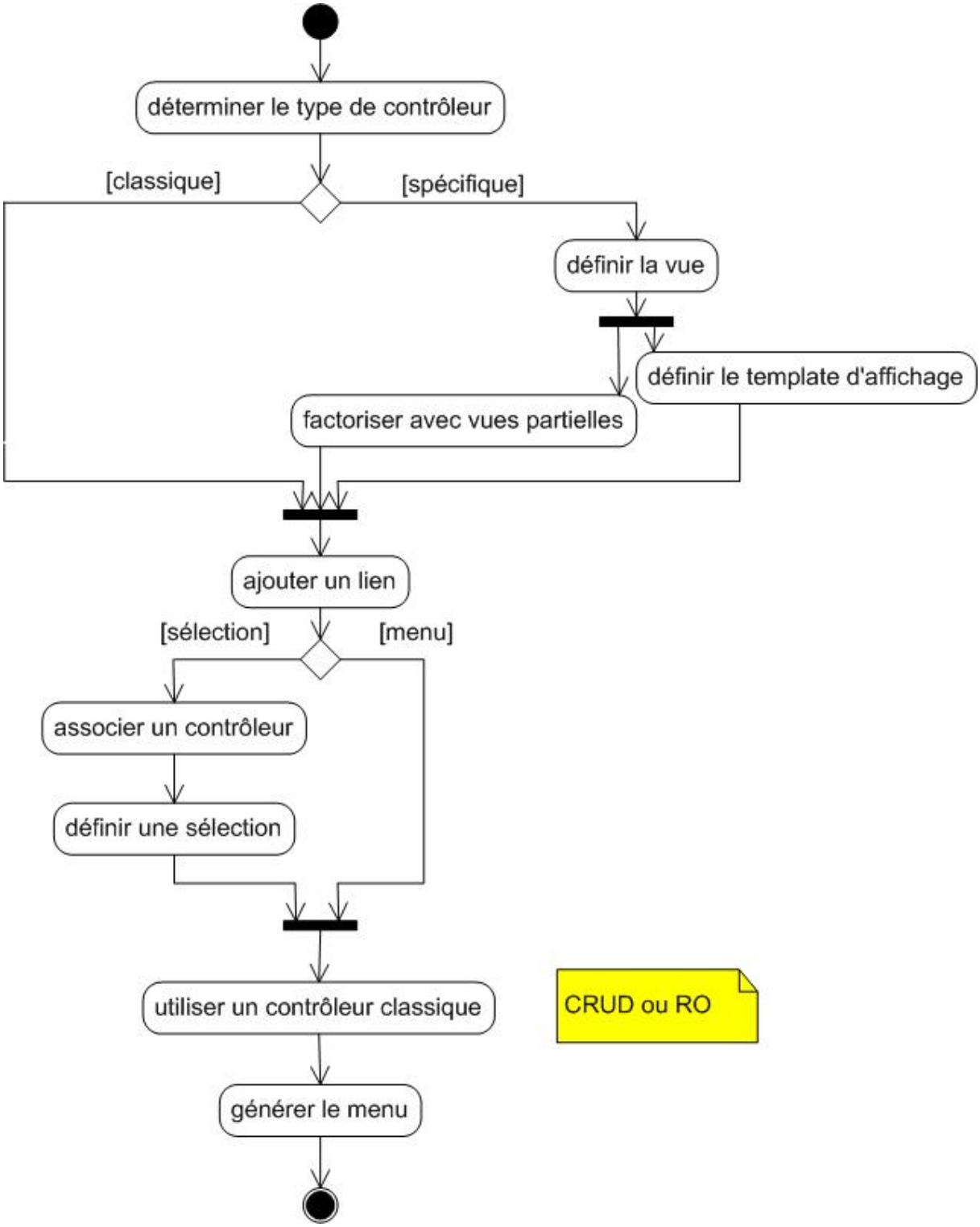
Développement RAILS – modèle



Développement RAILS – contrôleur



Développement RAILS – vues



Développement report