



**HAL**  
open science

## La supervision des systèmes d'information

Jérôme Poussineau

► **To cite this version:**

Jérôme Poussineau. La supervision des systèmes d'information. Architectures Matérielles [cs.AR]. 2010. dumas-00530233

**HAL Id: dumas-00530233**

**<https://dumas.ccsd.cnrs.fr/dumas-00530233>**

Submitted on 28 Oct 2010

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

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

**CONSERVATOIRE NATIONAL DES ARTS ET METIERS**

**CENTRE REGIONAL ASSOCIE DE LYON**

---

**MEMOIRE**

**présenté en vue d'obtenir**

**le DIPLOME D'INGENIEUR CNAM**

**SPECIALITE : informatique**

**OPTION : Systèmes d'information (ISI)**

**par**

**Jérôme POUSSINEAU**

---

**La supervision des systèmes d'information**

**Soutenu le 17 Septembre 2010**

---

**JURY**

**PRESIDENT : Christophe PICOULEAU**

**MEMBRES : Bertrand DAVID  
Claude GENIER  
Philippe AUBERTIN  
Edouard DUVERGER**



## Remerciements

Mes premiers remerciements s'adressent à Monsieur Claude GENIER, Enseignant au CNAM et ingénieur au CERN, pour m'avoir accompagné lors de la rédaction de ce mémoire.

Je remercie Monsieur Philippe AUBERTIN, PDG de la société Axopen, ainsi que Monsieur Pierre-Denis VANDUYNSLAGER, associé chez Axopen pour avoir tout deux cru en la réussite de ce projet.

Je remercie le CNAM, cette grande institution qui donne la chance à tous de suivre des études supérieures parallèlement à la vie active. Je remercie Monsieur Bertrand DAVID, responsable de la filière Informatique du CNAM à Lyon.

Je remercie aussi tous les membres de ce jury, dont Monsieur Christophe PICOULEAU qui me fait l'honneur de le présider.

Je remercie Messieurs Philippe AUBERTIN, Pierre-Denis VANDUYNSLAGER et Edouard DUVERGER, associés chez Axopen, pour leurs conseils avisés tout au long de la réalisation de mon travail.

A tous les permanents et stagiaires de la société Axopen, je leur adresse une salve de remerciements pour leurs contacts chaleureux.

Une pensée très particulière va à ma compagne Elisa pour son soutien total durant tant d'années où j'ai suivi les cours du soir, malgré les sacrifices que ceux-ci ont représenté pour elle.

Je tiens enfin à remercier mon entourage pour leur encouragement, notamment mes parents Jean-Claude et Valentine.

## Liste des abréviations

Abréviation	Description
SI	Système d'information
IHM	Interface Homme Machine
API	(Anglais) Application Programming Interface
JMS	(Anglais) Java Message Service
SLA	(Anglais) Service Level Agreement
ISM	(Anglais) Information System Manager
BAM	(Anglais) Business Activity Monitoring
BSM	(Anglais) Business Service Management
NSM	(Anglais) Network Systems Management
KPI	(Anglais) Key Performance Indicators
SDK	(Anglais) Software Development Kit
SOA	(Anglais) Service Oriented Architecture
MTBF	(Anglais) Mean Time Between Failures
AFR	(Anglais) Annualized Failure Rate
MTTR	(Anglais) Mean Time To Repair
AST	(Anglais) Agreed Service Time
WS	(Anglais) Web Service
SOAP	(Anglais) Simple Object Access Protocol
ESB	(Anglais) Entreprise Service Bus
MOM	(Anglais) Message Oriented Middleware
AJAX	(Anglais) Asynchronous JavaScript and XML
XML	(Anglais) eXtended Markup Language
JAR	(Anglais) Java ARchive
MOE	(Français) Maîtrise d'œuvre
MOA	(Français) Maîtrise d'ouvrage
SWOT	(Anglais) Strengths, Weaknesses, Opportunities and Threats
JVM	(Anglais) Java Virtual Machine

**Tableau 1 : Liste des abréviations**

## Glossaire

Objet	Définition
Green computing	Le <b>Green computing</b> (ou Green IT), que l'on pourrait traduire par informatique écologique, est un concept marketing mais aussi une tendance technologique réelle qui consiste à tenir compte des contraintes et des coûts en énergie (alimentation électrique et climatisation) des matériels informatiques.
SOAP	Protocole applicatif indépendant des modalités de transport, permettant l'appel synchrone ou asynchrone d'un service.
Web Service	Un <b>web service</b> (en Français service web) est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.
Middleware	Un <b>middleware</b> (en Français intergiciel) est un logiciel servant d'intermédiaire de communication entre plusieurs applications, généralement complexes ou distribuées sur un réseau informatique.
Cluster	On parle de <b>cluster</b> (en Français grappe de serveurs ou ferme de calcul) pour désigner des techniques consistant à regrouper plusieurs ordinateurs indépendants (appelés nœuds, node en anglais) pour permettre une gestion globale et dépasser les limitations d'un ordinateur.
Sniffer	Un <b>sniffer</b> (en Français renifleurs) est un composant logiciel de récupération des informations circulant sur le réseau.
XML	Langage permettant de représenter toute structure arborescente de données, chaque nœud ayant une sémantique et une syntaxe définie par une balise.

**Tableau 2 : Glossaire**

## Table des matières

Remerciements .....	3
Liste des abréviations .....	4
Glossaire .....	5
Table des matières .....	6
Introduction .....	9
1 – INTRODUCTION DU THEME .....	9
2 – CONTEXTE DU PROJET .....	9
3 – PRESENTATION DE L’ENTREPRISE.....	10
1 - Création et organisation.....	10
2 - Les offres proposées par Axopen.....	11
IT Consulting .....	12
Software engineering .....	13
System integration .....	13
<b>PARTIE 1 LA GOUVERNANCE ET SUPERVISION DES SYSTEMES D’INFORMATION.....</b>	<b>15</b>
CHAPITRE 1 – LE ROLE DU SI DANS LA PERFORMANCE DES ENTREPRISES .....	16
Sous-chapitre 1 – Le SI vecteur de valeur pour l’entreprise.....	16
Sous-chapitre 2 – L’agilité nécessaire du SI.....	16
Sous-chapitre 3 – Le SI : un ensemble d’applications hétérogènes .....	17
Complexité trop importante du SI .....	18
Maîtrise de la constitution du SI.....	18
Organisation métier et SI en silos .....	19
CHAPITRE 2 – LA NECESSITE DE CONSTRUIRE UN SI TRANSVERSE .....	20
Sous-chapitre 1 – L’intérêt de la mutualisation .....	20
Périmètre de la mutualisation .....	21
Sous-chapitre 2 – Le besoin de supervision de l’ensemble .....	22
Sous-chapitre 3 – Les enjeux de la gouvernance du SI .....	23
CHAPITRE 3 – QU’EST CE QUE LA PERFORMANCE DANS UN SI ?.....	25
Sous-chapitre 1 – Définition de la notion de performance dans un SI .....	25
Les temps de réponse .....	25
La disponibilité du système .....	25
La robustesse .....	26
La capacité à monter en charge .....	26
Sous-chapitre 2 – La mesure des performances d’un SI.....	27
Sous-chapitre 3 – Les coûts induits par la recherche de la performance.....	28
<b>PARTIE 2 LA DEFINITION D’UNE PLATEFORME DE SUPERVISION .....</b>	<b>31</b>
CHAPITRE 1 – LES DIFFERENTS TYPES DE SOLUTION DE SUPERVISION .....	32
Sous-chapitre 1 – Les outils de type BAM .....	32
Définition et objectifs.....	32
Utilisateurs finaux.....	33
Fonctionnalités couvertes.....	33
Sous-chapitre 2 – Les outils de type BSM.....	34
Définition et objectifs.....	34
Sous-chapitre 3 – Les outils de type NSM.....	34
Définition et objectifs.....	34

Sous-chapitre 4 – Les solutions du marché.....	34
Offre Progress Software® « Actional » .....	34
Offre de supervision de Systar®.....	36
CHAPITRE 2 – LA SOLUTION CONÇUE PAR AXOPEN .....	37
Sous-chapitre 1 – Cahier des charges de la solution .....	37
Liste des exigences .....	37
Définition du périmètre de la solution .....	38
Ma contribution au projet.....	38
Sous-chapitre 2 – Maquette des écrans de la solution.....	39
Ecrans de gouvernance.....	39
Ecrans d'étude métier.....	41
Ecrans de conception .....	44
Ecrans d'exploitation .....	45
<b>PARTIE 3 LA CONSTRUCTION D'UNE PLATEFORME DE SUPERVISION.....</b>	<b>49</b>
CHAPITRE 1 – ARCHITECTURE DE LA SOLUTION .....	50
Sous-chapitre 1 – Composition du Framework.....	52
Serveur d'applications.....	52
Base de données.....	53
Bus de messagerie JMS.....	54
Framework technique et langage .....	55
Le processus de développement d'applications Web à l'aide du Framework GWT peut être matérialisé de la sorte : .....	56
Architecture n-tiers .....	56
La figure suivante illustre le principe : .....	57
Plateforme d'intégration continue .....	57
CHAPITRE 2 – DEVELOPPEMENT DE LA SOLUTION .....	60
Sous-chapitre 1 – Processus de monitoring .....	60
Sous-chapitre 2 – Structure de la base de données.....	61
Stockage des événements .....	61
Stockage du référentiel logique .....	62
Stockage du référentiel technique.....	63
Stockage des statistiques .....	64
Stockage des paramètres .....	66
Généralisation des objets du référentiel .....	67
Génération des ordres DDL et des entity Java .....	67
Sous-chapitre 3 – Collecte et traitements des évènements .....	71
Sondes complexes.....	71
Interfaces de communication.....	71
Moteur de collecte des évènements .....	72
Moteur de corrélation des évènements.....	77
Moteur d'orchestration des tâches et de remontée des alertes .....	78
Sous-chapitre 4 – Application de supervision.....	79
Conclusion.....	81
Bibliographie .....	83
Table des annexes.....	84
Annexe 1 Planning de développement des lots 1 et 2 de l'application .....	85
Annexe 2 Application de supervision développée pour l'entreprise ST Ericsson .....	86
Annexe 3 Application de supervision développée pour l'entreprise Boiron.....	89
Liste des figures.....	91
Liste des tableaux .....	93



# Introduction

## 1 – Introduction du thème

Mes deux dernières missions de mise en place d'une plateforme d'échanges pour les entreprises ST Ericsson et Boiron (annexes 2 et 3) m'ont amené à développer pour ces deux entreprises une interface Web dédiée à la supervision de ces plateformes d'échange.

Il m'a alors paru intéressant en partant de ces travaux de construire une solution de supervision des applications ainsi que des interfaces d'échange sans être fortement lié aux technologies sous-jacentes.

Ce projet consiste donc avec l'appui du service R&D de l'entreprise Axopen de développer une application de supervision indépendante de la technologie et contexte client pour offrir aux utilisateurs :

- ▶ une vision de l'état de marche du Système d'Information (SI)
- ▶ une vision de la cartographie applicative du SI
- ▶ une vision des interdépendances entre les composants
- ▶ des statistiques sur les performances et l'utilisation du SI

L'application devra offrir des sondes complexes pouvant être insérées dans les programmes Java et .NET ainsi qu'au niveau des bases de données Oracle et IBM DB2. Cette application devra collecter les informations de manière asynchrone pour limiter l'impact sur les composants supervisés.

## 2 – Contexte du projet

Le développement de cette application s'inscrit dans le cadre d'un programme de recherche et développement phare de l'entreprise informatique Axopen nommé *A world of innovation*.

Ce programme initié par la direction de l'entreprise vise à construire une gamme complète de produits et de services permettant aux entreprises fortement liées à leur système d'information d'améliorer leur capacité à :

- ▶ Piloter le développement de nouveaux composants.
- ▶ Réutiliser des composants existants dans leur SI.
- ▶ Superviser le cycle de vie des instances des composants déployés sur leur SI
- ▶ Contrôler l'exécution des instances de ces composants
- ▶ Superviser l'état des serveurs logiques et physiques
- ▶ Fournir des indicateurs de qualités sur les composants logiciels et matériels
- ▶ Fournir des indicateurs d'utilisation des composants logiciels et matériels

### 3 – Présentation de l'entreprise

L'entreprise Axopen dans laquelle cette étude a été réalisée est une société de service en ingénierie informatique. Cette entreprise a réussi en un temps très court à faire sa place dans ce domaine d'activité très concurrentiel.

#### 1 - Création et organisation

Créée le 22 Janvier 2007 par Monsieur Philippe AUBERTIN, la société Inforésolutions s'est destinée dans ses débuts à assister ses clients dans la mise en place de réseaux d'entreprise, le développement sur mesure de sites Web et de petites solutions de gestion.

Suite notamment à la montée au capital de Monsieur Jérôme POUSSINEAU et à la volonté de Monsieur Philippe Aubertin, la société prend à partir du 18 Décembre 2009 un nouveau tournant en recentrant son activité sur les métiers du conseil en solutions informatiques, développements sur mesure et intégration de systèmes. Rapidement, deux nouveaux associés Monsieur Pierre-Denis VANDUYNSLAGER et Monsieur Edouard DUVERGER rejoignent la structure pour apporter leur contribution durant cette phase de repositionnement de l'activité.

Cette nouvelle orientation est notamment marquée par le changement du nom de la société pour Axopen – Innovative Solutions. Ce changement de nom est accompagné par la création d'un nouveau logo et d'une nouvelle chartre graphique s'articulant autour du bleu, du blanc et du gris.



Figure 1 : Logo de l'entreprise Axopen

A l'heure actuelle, l'entreprise Axopen est composée de quatre employés actionnaires ainsi que 3 employés non actionnaires. Les responsabilités sont réparties selon le diagramme ci-dessous :

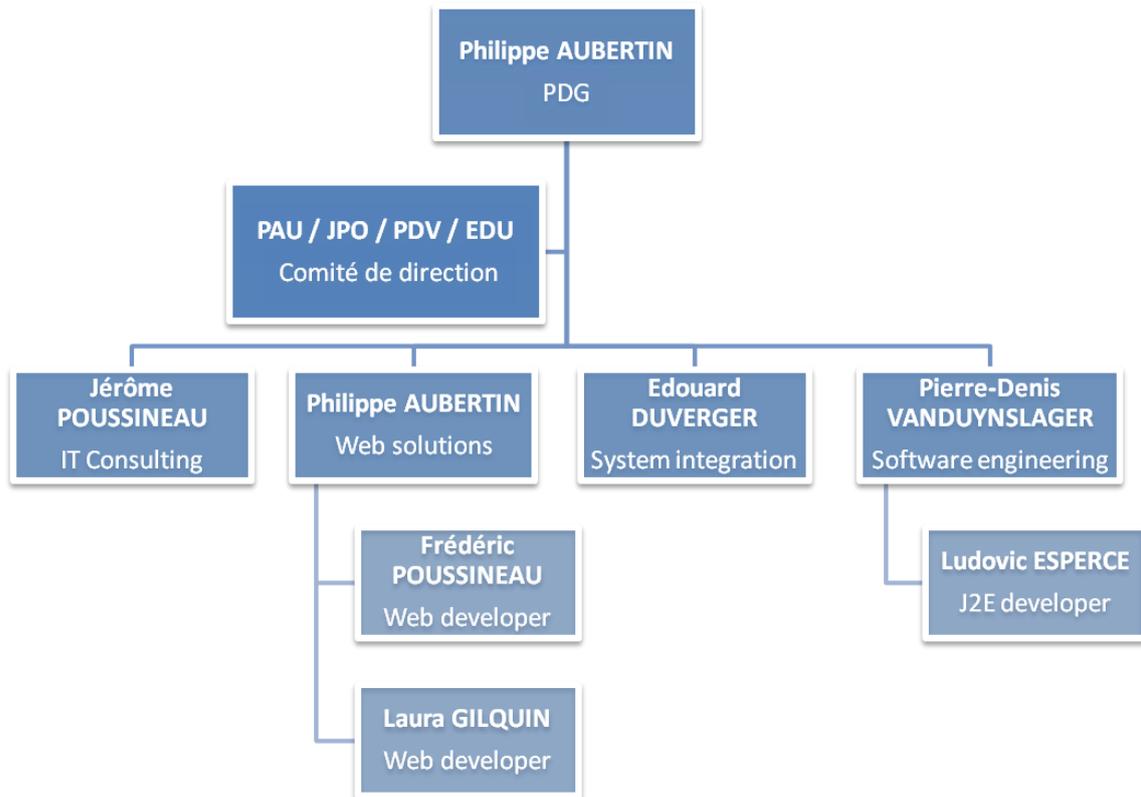


Figure 2 : Organisation Axopen

## 2 - Les offres proposées par Axopen

L'entreprise Axopen propose toute une gamme de produits et services informatiques allant du conseil en système d'informations, développement de solutions sur mesure et intégration de systèmes.

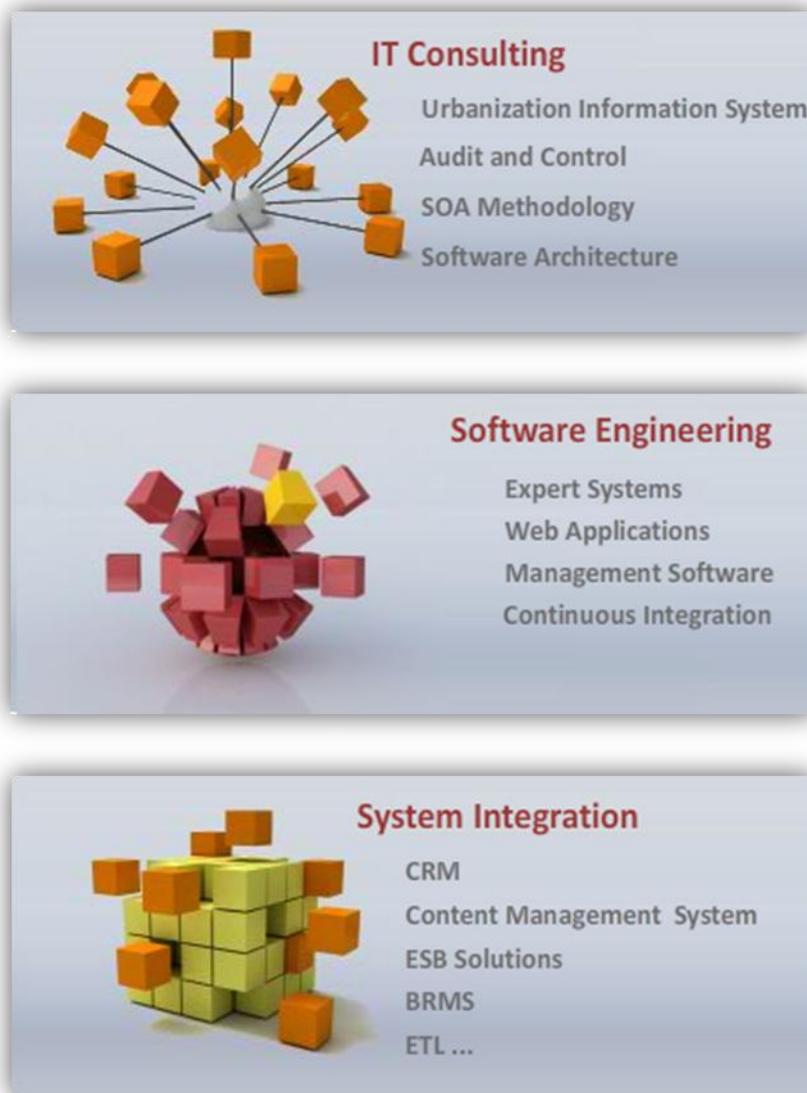


Figure 3 : Offres de l'entreprise Axopen

### **IT Consulting**

Le système d'informations, vecteur de performances dans les organisations modernes, est basé le plus souvent pour une entreprise sur un ensemble de technologies, d'architectures et de méthodes acquises au fil du temps.

Cette hétérogénéité limite in fine les capacités du système d'informations à s'adapter aux évolutions de la stratégie de l'entreprise concernée.

Dans un monde où la capacité d'adaptation d'une entreprise face aux fluctuations de son marché devient un élément différenciant majeur, il est nécessaire de remettre l'architecture transverse au centre du débat pour ainsi permettre d'augmenter l'agilité des systèmes d'informations des entreprises et par la même occasion réduire le délai nécessaire à l'alignement entre les besoins métiers et l'outil informatique.

Conscient de cet enjeu majeur pour les entreprises, Axopen propose un ensemble de prestations de conseil en organisation et en technologies permettant d'atteindre cet objectif.

Conseil en organisation :

- ▶ Diagnostic sur une organisation du domaine Informatique.
- ▶ Assistance à la définition d'un schéma directeur.
- ▶ Assistance à l'urbanisation d'un système d'informations.

Conseil en technologies :

- ▶ Assistance à la définition d'une architecture d'entreprise.
- ▶ Assistance à la définition d'une architecture applicative.
- ▶ Méthodologie de mise en place d'une architecture orientée services.
- ▶ Aide aux choix technique et technologique.
- ▶ Audit de survol d'une application.

### **Software engineering**

Puisque chaque client est unique et qu'une solution informatique n'a de sens que pour un contexte donné, Axopen propose le développement de solutions sur mesure.

Afin de vous offrir le meilleur de la technologie au meilleur coût Axopen a constitué un ensemble d'accélérateurs de développement permettant la génération d'une partie du code source ainsi que la composition d'interfaces Web à partir d'objets graphiques entièrement personnalisables.

Pour vous offrir des prestations de qualité, Axopen a su investir dans des solutions permettant l'intégration et le test en continu pendant tout le cycle de développement ainsi que le contrôle automatique de la qualité du code source.

Axopen s'est ainsi spécialisé dans le développement d'applications, de type interface client riche ou interface Web, à partir de technologies reconnues sur le marché :

- ▶ Java / J2E
- ▶ Microsoft .NET
- ▶ PHP
- ▶ Adobe ColdFusion, FLEX et Air

### **System integration**

Pourquoi réinventer la roue alors que des solutions existent ?

Pour certains besoins métiers ou techniques, tels la gestion de la relation client, la gestion de contenus ou le transport de données ; les solutions Open Source ou propriétaires entièrement paramétrables offrent un coût de mise en place inférieur au développement d'une solution sur mesure.

Fort de ce constat, Axopen vous propose de prendre en charge l'intégration de ces solutions, depuis la phase de choix de l'outil à l'accompagnement au changement des équipes.

Solutions fonctionnelles proposées par Axopen :

- ▶ Web et gestion de contenu (Content Management System)
- ▶ Gestion de la relation client (Customer Relationship Management)
- ▶ Gestion électronique des documents (Enterprise Content Management)
- ▶ Progiciel de gestion d'entreprise (Enterprise Resource Planning)
- ▶ Aide à la décision (Business Intelligence).
- ▶ Supervision de l'activité en temps réel (Business Activity Monitoring)
- ▶ Gestion des données référentielles (Master Data Management)
- ▶ Gestion des catalogues produits (Product Information Management)

Solutions techniques proposées par Axopen :

- ▶ Automatisation des tâches (Continuous Integration System)
- ▶ Bus de services (Enterprise Service Bus)
- ▶ Transport et transformation de données (Extract Transform & Load)
- ▶ Gestion et exécution des règles métiers (Business Rules Management System)
- ▶ Gestion et exécution des processus métiers (Business Process Management)
- ▶ Intégration d'applications (Enterprise Application Integration)
- ▶ Corrélation d'événements (Complex Event Processing)
- ▶ Transport de messages (Message Oriented Middleware)

## **Partie 1**

# **La gouvernance et supervision des systèmes d'information**

# Chapitre 1 – Le rôle du SI dans la performance des entreprises

Comme toute nouveauté le SI des entreprises a tout d'abord été utilisé comme un élément stratégique permettant d'améliorer la productivité des entreprises. Au fil du temps, dans la majorité des secteurs d'activités, le SI est devenu un élément critique indispensable au fonctionnement des entreprises en question (système de paiement, réservation des trains, gestion des stocks...).

## Sous-chapitre 1 – Le SI vecteur de valeur pour l'entreprise

La compétition de plus en plus féroce entre les entreprises que connaît le 21<sup>ème</sup> siècle s'accompagne d'une montée en puissance de l'informatisation des processus. Cette montée en puissance est provoquée par le besoin d'accélérer les traitements et les échanges :

- ▶ Amélioration de la productivité grâce à l'assistance informatique.
- ▶ Remplacement des tâches récurrentes par des traitements automatisés.
- ▶ Echanges de données informatisés entre les entreprises.

De fait, l'outil informatique devient aujourd'hui indispensable dans la majorité des secteurs d'activité (banque, assurance, services...).

Un SI performant permet alors à une entreprise de :

- ▶ Mettre sur le marché rapidement des produits ou services innovants
- ▶ Optimiser les processus afin de réduire les coûts
- ▶ Sous-traiter au maximum les tâches à faible valeur ajoutée
- ▶ Améliorer la relation client grâce à une vision unique et une cohérence d'image et de comportement sur les différents canaux de vente

Un système d'information est jugé performant par les entreprises notamment par :

- ▶ Son niveau d'agilité : capacité à s'adapter aux changements métiers et/ou technologiques.
- ▶ Sa performance technique : temps de réponse, capacité à monter en charge, robustesse et fiabilité.
- ▶ Son architecture : centralisée, client /serveur ou orientée services.

Nous définirons dans les prochains chapitres ces notions ainsi que les difficultés auxquelles les entreprises font face pour les appliquer. Enfin, dans une dernière partie nous aborderons la capacité à répondre à ces enjeux grâce à notre solution de supervision.

## Sous-chapitre 2 – L'agilité nécessaire du SI

Dans un siècle où l'économie tend à se mondialiser et où le changement permanent est devenu la norme, le système d'information des entreprises doit s'adapter à ces nouvelles contraintes. Il doit être plus réactif pour aider les métiers à mettre sur le marché le plus rapidement possible de nouveaux produits tout en limitant la consommation de ressources

La capacité d'adaptation d'une organisation n'est alors plus un luxe mais une absolue nécessité. Le SI comme vecteur de valeur se doit de contribuer efficacement à la capacité d'évolution d'une organisation.

Cette évolutivité doit être vue comme la facilité avec laquelle un système d'information va pouvoir s'aligner face à de nouveaux besoins et à de nouvelles demandes. Cette agilité va permettre d'influencer le *time to market* (délai de mise sur le marché) nécessaire au niveau informatique lors de la création d'un nouveau produit ou bien la capacité à optimiser le fonctionnement d'une organisation en un temps très court.

### Sous-chapitre 3 – Le SI : un ensemble d'applications hétérogènes

Le problème que rencontrent les sociétés à l'heure actuelle avec leur système d'information est que celui-ci dispose d'un nombre d'applications de plus en plus important fonctionnant sur des systèmes différents, destinés à des utilisateurs de plus en plus variés, qu'ils utilisent depuis des plateformes matérielles de plus en plus diverses.

En plus de ces applications, l'entreprise dispose souvent d'une multitude de bases de données fonctionnant avec des technologies différentes, plusieurs référentiels sur des plateformes hétérogènes.

Cette hétérogénéité est mise en exergue lorsque l'entreprise concernée souhaite développer de nouvelles applications en réutilisant au maximum les composants existants au sein du SI.

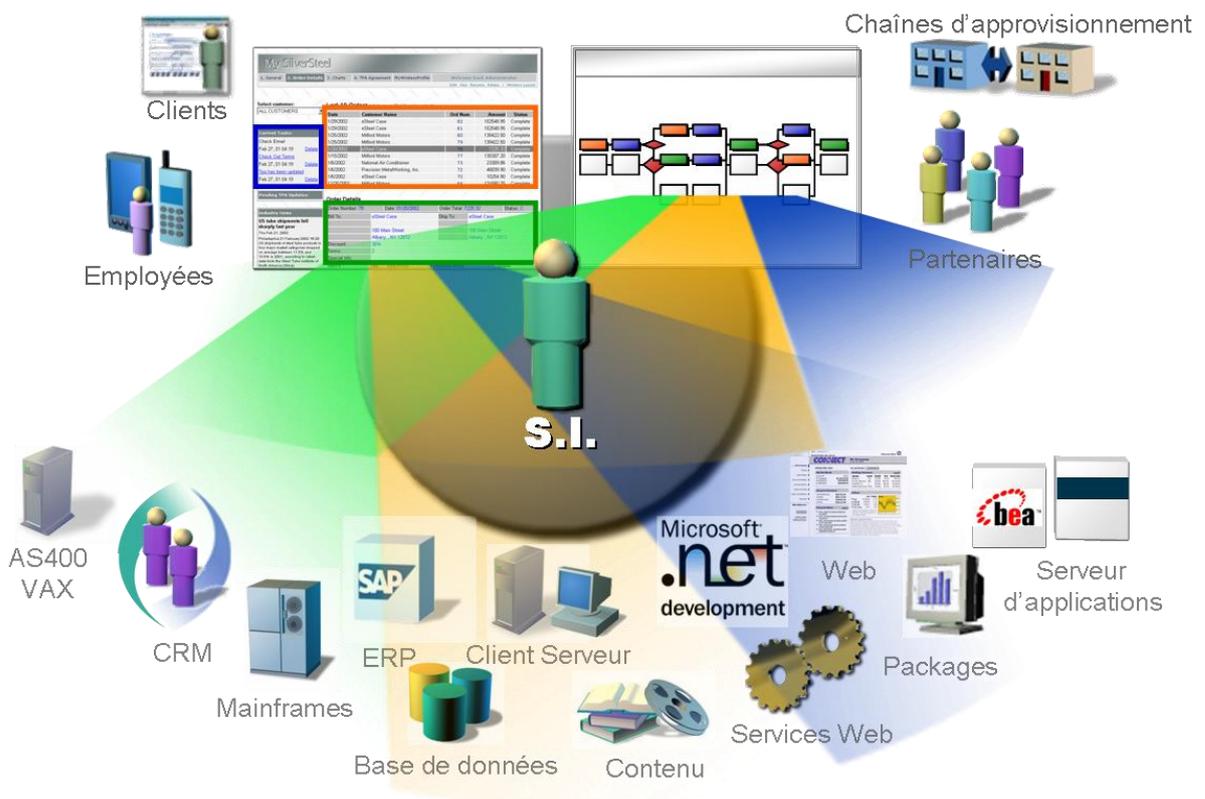


Figure 4 : Illustration de l'hétérogénéité des systèmes d'information

Afin de permettre tout de même la supervision de systèmes aussi hétérogènes il est nécessaire de disposer d'une solution ouverte pouvant s'adapter aux différentes technologies en présence.

### **Complexité trop importante du SI**

Plusieurs facteurs ont entraîné dans la majorité des entreprises une complexification non justifiée de leur système d'information, on citera notamment :

- ▶ L'observation dans les systèmes d'information actuels d'une duplication par 3 de la logique métier et/ou des informations métier. Cette duplication entraînant des surcoûts en termes de développement mais surtout de maintenance.
- ▶ Le fait qu'un système d'information soit bâti sur une période de plusieurs dizaines d'années provoquant in fine, dans la majorité des cas, un empilement des applications avec stratification au fil du temps.
- ▶ La logique de mise en œuvre de *quick-win*<sup>1</sup> sans réorganisation des démontages à chaque nouveau projet entraîne bien souvent une hétérogénéité importante entre les applications d'un même domaine.
- ▶ L'introduction de plus en plus fréquente de progiciels basés sur des plateformes techniques totalement différentes renforçant d'avantage cette complexité.

### **Maîtrise de la constitution du SI**

On observe dans bon nombre d'entreprises des difficultés grandissantes dans la mesure de l'impact de tout changement, ces difficultés étant principalement dues aux facteurs suivants :

- ▶ La globalité du système d'information n'est connue que de quelques personnes clés, le papy boom devient alors un risque de perte de la connaissance du SI.
- ▶ Une cartographie du système d'information existe, mais elle n'est plus à jour ou mise à jour par des urbanistes qui ne sont pas assez proches des projets.

---

<sup>1</sup> **Quick-win** : Pour un consultant en informatique, un *quick-win* est une action rapide permettant un gain substantiel.

## Organisation métier et SI en silos

L'organisation des métiers (MOA<sup>2</sup>) et projets informatiques (MOE<sup>3</sup>) en silos a entraîné au niveau des systèmes d'information un certain nombre de difficultés :

- ▶ Difficulté à réutiliser la logique métier déjà mise en œuvre dans un autre domaine de l'entreprise.
- ▶ Manque d'objectifs et de budgets partagés entre les différents domaines du SI.
- ▶ La conception des applications réalisée par domaine fonctionnel, aucune mutualisation n'étant alors possible.
- ▶ La logique transverse du système d'information (logique métier ou technique) diluée dans l'ensemble des applications.

---

<sup>2</sup> **MOA** (Maîtrise d'ouvrage) : En informatique le maître d'ouvrage intervient en amont du projet. Il doit avoir une très bonne connaissance fonctionnelle du domaine. Il doit fixer les objectifs du projet, proposer et élaborer les solutions. Il doit aussi définir les coûts et les délais du projet.

<sup>3</sup> **MOE** (Maîtrise d'œuvre) : En informatique le maître d'œuvre intervient au niveau de la réalisation du projet. Il établit les spécifications techniques à partir des spécifications fonctionnelles en tenant compte des contraintes techniques. Il doit donc avoir une bonne connaissance technique.

## Chapitre 2 – La nécessité de construire un SI transverse

La gestion du SI des entreprises représente un budget conséquent pour celle-ci. Les banques et assurances par exemple y consacrent 4 à 5% de leur chiffre d'affaire. La construction d'un SI transverse permet aux entreprises de réaliser des économies sur l'évolution et la maintenance de leur système en facilitant la réutilisation et en diminuant la complexité de celui-ci.

### Sous-chapitre 1 – L'intérêt de la mutualisation

Le système d'information reste en soi un ensemble homogène d'applications, seulement si les applications ont été développées ou sont en tant que telles développées sur la base de standards permettant la réutilisation. Il est possible alors pour tout nouveau développement de se baser sur des composants existants exposés par les applications qui couvrent un besoin métier proche.

Le développement d'une application devient alors plus proche de l'assemblage d'appels à des composants existants ou nouveaux, plutôt qu'à l'écriture de traitements sombres et obscurs.

L'entreprise retrouve par la même occasion une meilleure vision de son système d'information. Elle retrouve des métriques utilisables en comptant le nombre de services de types règles métiers, interfaces, données de références...

La mise en place d'un SI transverse facilite grandement cette réutilisation des composants en offrant un socle technique sur lequel s'appuyer. Outre l'aspect financier qui devient une évidence pour bon nombre de décideurs, la mutualisation de composants du SI permet notamment :

- ▶ **Une rationalisation du SI :** La mutualisation des fonctions redondantes sous forme de composants réutilisables entraîne in fine une réduction des coûts de développement puisque l'on développera une seule fois le composant pour qu'il soit utilisé par plusieurs applications mais surtout une réduction des coûts de maintenance évolutive et/ou corrective puisque lors d'un besoin de maintenance sur cette fonctionnalité, seul le composant mutualisé sera modifié contre ses  $x$  versions auparavant. Par ailleurs, la gestion du SI à une maille plus petite (découlant du fait de travailler au niveau composant et non application lors des études sur le système d'information) entraîne une diminution de la complexité globale de gestion, puisqu'il sera possible de versionner désormais les composants sans toucher aux applications lors de tout nouveau besoin métier.
- ▶ **Une augmentation de sa souplesse :** Cette gestion du changement, réalisée par composant permet de diminuer les temps de cycle projet puisque l'on s'occupera de modifier uniquement les composants dont les règles métiers évoluent et ainsi cibler notre action (contrairement à une gestion du changement par application). Cette agilité permettra alors d'obtenir un alignement plus rapide sur le métier lors d'une évolution de celui-ci (avantages concurrentiels, évolutions de la réglementation...). Lors de l'appel à un composant existant depuis une nouvelle application les bonnes pratiques impliquent l'externalisation des aspects organisationnels du système dans un fichier de configuration prévu à cet effet. Il s'agit principalement de l'adresse du

composant dans le système d'information (exemple : URL). Cette bonne pratique permet alors de modifier facilement l'adresse d'un composant lors de son déplacement ou de sa duplication ou bien d'utiliser une version différente de celui-ci lorsque cela est nécessaire. La standardisation des interfaces d'échange apportée par certains standards (HTTP, SOAP, XML...) permet d'accélérer grandement la mise en place de tout nouveaux canaux de communication (interne, e-commerce...). Cette augmentation de l'interopérabilité est visible aussi lors de tout changement dans le système d'information (ajout d'un nouveau progiciel, développement d'une nouvelle application...).

- ▶ **Une pérennisation des investissements :** La facilité d'exposer des fonctions existantes sous forme de composants permet aux entreprises, dont le système d'information est majoritairement constitué d'applications issues de leur patrimoine applicatif (legacy, mainframes...), de pérenniser au maximum ce patrimoine. On peut citer notamment les banques pour lesquelles on estime à l'heure actuelle que leur système d'information est constitué pour 60% d'applications antérieures aux années 1980. Dans ces sociétés, cette capacité permet de réutiliser une partie de ce patrimoine existant dans de nouveaux contextes d'utilisation (site de gestion, e-commerce...).

En conclusion, la mutualisation des composants du SI permet aux entreprises qui la mettent en pratique d'augmenter l'agilité de leur système d'information, et donc sa capacité à évoluer rapidement et à faible coût tout en réduisant les coûts de développement et de maintenance. Et ce, grâce à la rationalisation du SI, rendue possible par la réutilisation de composants développés récemment mais aussi de programmes issus du patrimoine de l'entreprise.

### Périmètre de la mutualisation

Une application informatique peut être vue comme un ensemble de traitements ayant pour rôles :

- ▶ De proposer une couche de présentation aux utilisateurs (IHM pour Interface Homme Machine).
- ▶ De permettre l'accès à du contenu non structuré (vidéos, images, documents...).
- ▶ De permettre l'accès à des données structurées (données transactionnelles comme par exemple les commandes et données référentielles telles que le référentiel des produits).
- ▶ D'exécuter des règles métiers et retourner le résultat en fonction du contexte (si *age\_client* < 21 et *conduite\_accompagnee* == faux alors *tarif* = *maximum*).
- ▶ De réaliser l'orchestration entre plusieurs traitements et/ou services de l'application et/ou du SI de l'entreprise (par exemple processus de commande).
- ▶ De permettre l'interfaçage avec le reste du SI ainsi que le SI des clients, partenaires ou fournisseurs (par exemple un lien est souvent nécessaire entre l'application de gestion des clients et celle de gestion des commandes).

Forts de ces différents rôles, les traitements contenus au sein d'une application doivent être de préférence développés avec des outils et des normes permettant de faciliter la réutilisation et le couplage lâche (limitation de l'impact d'un changement grâce au contrat d'interface standardisé).

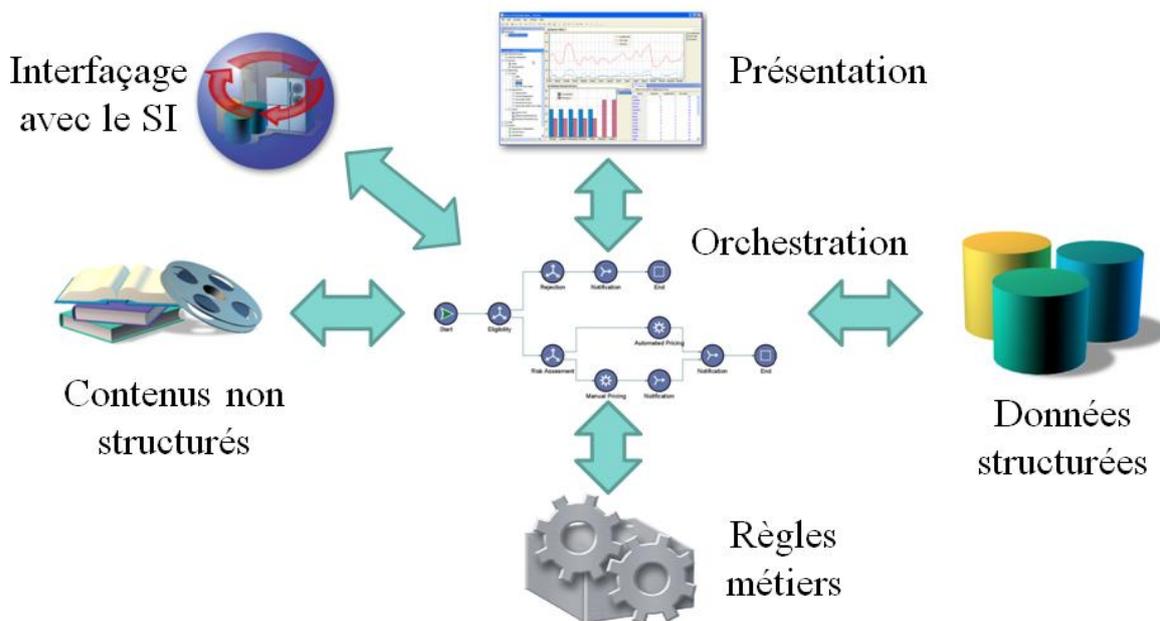


Figure 5 : Liste des traitements mutualisables dans un SI

Le nombre de composants mutualisables augmentant de manière importante, au sein des entreprises mettant en œuvre une démarche de mutualisation, il devient alors primordial de disposer d'un système de supervision / gouvernance du SI capable d'offrir une vision exhaustive des différents composants réutilisables et de leur rôle au sein du SI.

## Sous-chapitre 2 – Le besoin de supervision de l'ensemble

Piloter un véhicule sans tableau de bord s'avère être un jeu risqué, en effet tout dysfonctionnement de la voiture non connu par le chauffeur (manque d'huile, température anormale du moteur...) pourra entraîner une panne dont il faudra déterminer l'origine et assumer les conséquences très coûteuses face aux actions qui auraient permis de l'éviter (ajout d'huile, de liquide de refroidissement...).

La supervision d'un système d'information suit ce principe, en effet il est souvent plus rapide et moins coûteux d'éviter une panne que de la réparer lorsque celle-ci se produit (écroulement d'un serveur devant la charge, espace disque insuffisant pour une base de données...).

### Sous-chapitre 3 – Les enjeux de la gouvernance du SI

La gouvernance du système d'information consiste en la mise en œuvre d'un plan nommé schéma directeur visant à absorber au niveau du SI les impacts de la stratégie de l'entreprise.

La gouvernance d'un SI se décline en trois phases :

- ▶ Définir la stratégie : élaboration du schéma directeur.
- ▶ Piloter : mise en œuvre du schéma directeur.
- ▶ Auditer : vérifier que les objectifs ont bien été atteints.

Ces trois phases étant réalisées de manière cyclique (cf. illustration ci-dessous), un schéma directeur a donc une durée de vie limitée dans une entreprise. L'entreprise doit alors disposer d'un référentiel lui permettant de réaliser plus simplement le prochain schéma directeur à appliquer.

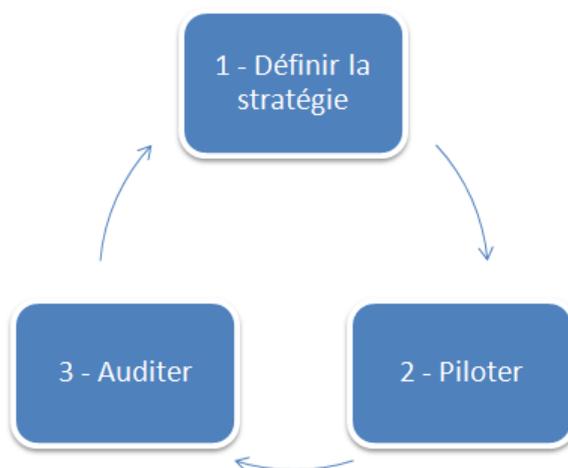


Figure 6 : Illustration du caractère cyclique de la gouvernance du SI

La rédaction d'un schéma directeur est issue de l'application d'une méthodologie permettant de confronter les objectifs stratégiques de l'entreprise avec la réalité du SI.

Pour ma part, la méthodologie que j'utilise pour permettre la définition d'un schéma directeur pour une entreprise de taille moyenne est constituée des étapes suivantes :

- ▶ Définition ou collecte des objectifs stratégiques de l'entreprise (diagramme d'Ishikawa<sup>4</sup>).
- ▶ Définition des objectifs stratégiques du SI.
- ▶ Mesure de l'alignement entre les objectifs stratégiques de l'entreprise et le SI en réalisant un audit de survol du SI ainsi qu'une analyse SWOT<sup>5</sup>.

---

<sup>4</sup> **Diagramme d'Ishikawa** : Ce diagramme aussi nommé diagramme de causes et effets ou diagramme en arêtes de poisson est le fruit des travaux de Kaoru Ishikawa.

- ▶ La cartographie des processus (si elle existe) doit être mise à jour avec les processus ajoutés, modifiés ou supprimés en privilégiant les processus cœur de métier. Ces processus sont identifiés en réalisant une analyse « Chaîne de valeur de Porter ».
- ▶ L'interview des utilisateurs clés du SI permet le plus souvent de dégager des axes d'amélioration du SI en conformité avec les objectifs stratégiques.
- ▶ A partir des éléments précédents il est alors possible d'élaborer plusieurs scénarios (de 2 à 4) en intégrant les facteurs humain, organisationnel, technologique et financier.

Le schéma directeur consiste à l'élaboration d'un plan permettant de mettre en place le scénario retenu sur une période de 2 à 5 ans.

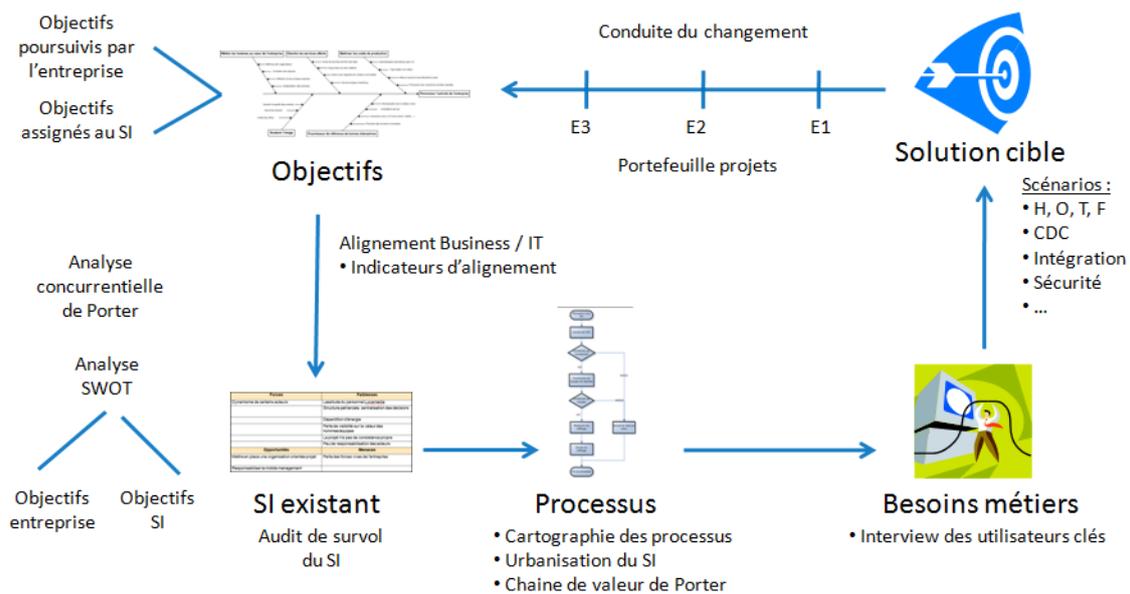


Figure 7 : Aperçu d'une méthodologie permettant de définir un schéma directeur

Un des objectifs de l'application de supervision sera de faciliter ce travail en assistant l'utilisateur durant l'audit de survol du SI ainsi que la cartographie des processus (cf. maquettes de la solution au Chapitre 2 de la Partie 2).

<sup>5</sup> **SWOT** : L'analyse SWOT ou matrice SWOT, de l'anglais *Strengths* (forces), *Weaknesses* (faiblesses), *Opportunities* (opportunités), *Threats* (menaces), est un outil de stratégie d'entreprise permettant de déterminer les options stratégiques envisageables au niveau d'un domaine d'activité.

## Chapitre 3 – Qu’est ce que la performance dans un SI ?

Une application offrant des performances médiocres peut ralentir la productivité des collaborateurs et entraîner frustrations et stress jusqu’au rejet total de l’application par les utilisateurs. Un projet informatique est réussi uniquement lorsque l’application qui en découle est réellement utilisée par les équipes et permet de répondre aux enjeux métiers.

### Sous-chapitre 1 – Définition de la notion de performance dans un SI

Au niveau d’un système informatique la performance ne se définit pas uniquement par les temps de réponse résultants des applications aux utilisateurs, cette notion est plus vaste et comprend les aspects suivants :

- ▶ Les temps de réponse (*respond time*)
- ▶ La disponibilité du système (*availability*)
- ▶ La robustesse (*robustness*)
- ▶ La capacité de montée en charge (*scalability*)

#### Les temps de réponse

Un temps de réponse désigne la durée d’exécution d’une opération sur le système informatique. Cette opération, par exemple l’affichage d’une page Web de présentation d’un article, peut recouvrir l’invocation de plusieurs composants logiciels (serveur web, serveur d’application, serveur de base de données etc...).

Des temps de réponse non conformes aux attentes impliquant des ralentissements visibles par les utilisateurs peuvent entraîner une mauvaise acceptation voire un rejet dans certains cas de l’application ou du site Web.

#### La disponibilité du système

La disponibilité d’un composant du système informatique désigne le ratio de temps pendant lequel il est en état de fonctionner correctement sur une période de temps donnée (figure ci-dessous). La disponibilité s’exprime en pourcentage, elle est notée A ou HA en Anglais (pour *Availability* ou *High Availability*).

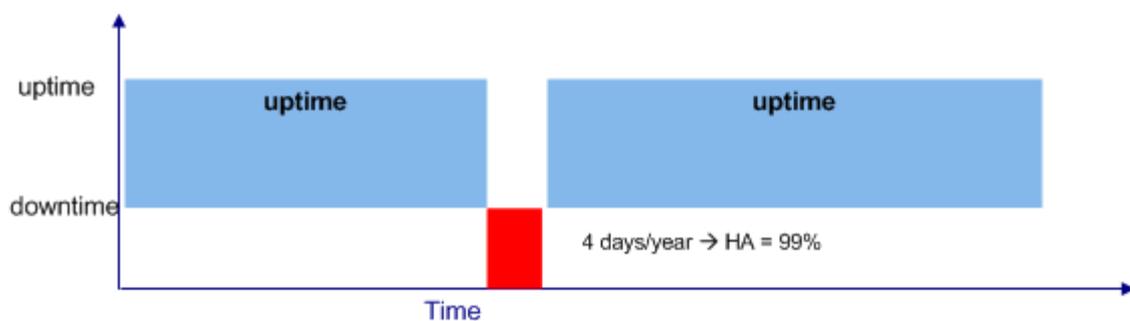


Figure 8 : Illustration de la notion de disponibilité

Les applications d'entreprise ont généralement une disponibilité de 90% à 95%. A partir de 99% on parle d'architecture à « haute disponibilité ».

La disponibilité peut concerner :

- ▶ Une application ou un service ;
- ▶ Un équipement matériel (serveur, baie de stockage, etc...) ;
- ▶ Un équipement réseau (routeur, répartiteur de charges, etc...) ;
- ▶ Un îlot applicatif ou une plateforme complète ;

On parle de « tolérance aux pannes » (*fail over*) pour un système qui peut fonctionner lorsqu'un de ses composants est défaillant et peut être remplacé à chaud, c'est-à-dire sans arrêt du service.

### **La robustesse**

La robustesse désigne la capacité d'un système à ne pas « planter » et « perdre ou corrompre » des données ou des messages lorsqu'il est soumis à des sollicitations inhabituelles. Il s'agit donc d'une mesure de la disponibilité des systèmes et de l'intégrité des informations en situation de stress.

L'intégrité peut se porter sur :

- ▶ Des données échangées avec d'autres systèmes (messages, fichiers, etc.) ;
- ▶ Des données persistantes stockées dans des bases de données, annuaires, serveurs de fichiers, etc.

La robustesse s'exprime par une grandeur scalaire sans unité, par exemple :

- ▶ Pas plus d'un message perdu pour 100 000 messages traités ;
- ▶ Au maximum trois redémarrages d'un serveur de base de données par mois ;

### **La capacité à monter en charge**

La capacité à monter en charge désigne l'aptitude d'une application ou d'un service à offrir des temps de réponse « raisonnables » quand la quantité d'utilisateurs simultanés augmente.

La capacité à monter en charge s'exprime par « une durée maximum de traitements pour un niveau de charge donné ». C'est-à-dire un temps de réponse pour un volume de charge. On parle aussi de débit ou *throughput*, mesuré en nombre de tâches simultanées par unité de temps.

Par exemple :

- ▶ 3 secondes maximum pour afficher une page Web avec 100 utilisateurs simultanés sur le site ;
- ▶ 3h30 maximum pour traiter les factures des 10 000 clients en base de données.

La capacité à monter en charge est une caractéristique tout aussi importante que les temps de réponse car elle précise le contexte dans lequel ceux-ci doivent être atteints.

## Sous-chapitre 2 – La mesure des performances d'un SI

Un test de performance est un test dont l'objectif est de mesurer la performance d'un système informatique par rapport aux contraintes et exigences identifiées au début du projet (temps de réponse maximum admissible par page, nombre d'utilisateurs simultanés...).

Pour mesurer les performances des applications constituant un SI, il est possible d'effectuer les tests suivants :

- **Test de charge** : Lors de ce type de test on simule l'utilisation progressive de l'application par un nombre d'utilisateurs jusqu'à arriver à  $x$  fois le nombre d'utilisateurs cibles et ainsi valider l'application pour une charge attendue d'utilisateurs en ayant tout de même une certaine marge de sécurité. Ce type de test permet de mettre en évidence les points sensibles et critiques de l'architecture technique. Il permet en outre de calculer le dimensionnement (*sizing*) des serveurs ainsi que de la bande passante réseau nécessaire pour la mise en production de l'application.
- **Test de performance** : A l'instar du test de charge, lors des tests de performances nous allons simuler un certain nombre d'utilisateurs et mesurer chacune des étapes constituant le temps de réponse final constaté par l'utilisateur grâce à des sondes techniques positionnées sur chacun des composants de l'architecture (base de données, serveur d'application, accès disque, réseau...). Ces tests permettent de ventiler le temps total de traitement d'une page entre les différents composants de l'architecture et ainsi proposer des axes d'amélioration de l'application lorsque l'on n'obtient pas de celle-ci les performances escomptées.
- **Test de dégradations des transactions** : il s'agit d'un test technique primordial au cours duquel on ne va simuler que l'activité transactionnelle d'un seul scénario fonctionnel parmi tous les scénarios du périmètre des tests, de manière à déterminer quelle charge limite simultanée le système est capable de supporter pour chaque scénario fonctionnel et d'isoler éventuellement les transactions qui dégradent le plus l'ensemble du système.
- **Test de stress** : il s'agit d'un test au cours duquel on va simuler l'activité maximale attendue tous scénarios fonctionnels confondus en heures de pointe de l'application, pour voir comment le système réagit au maximum de l'activité attendue des utilisateurs.
- **Test de robustesse, d'endurance et de fiabilité** : il s'agit de tests au cours duquel on va simuler une charge importante d'utilisateurs sur une durée relativement longue, pour voir si le système testé est capable de supporter une activité intense sur une longue période sans dégradations des performances et des ressources applicatives ou système.
- **Test de capacité, test de montée en charge** : il s'agit d'un test au cours duquel on va simuler un nombre d'utilisateurs sans cesse croissant de manière à déterminer quelle charge limite le système est capable de supporter.

- **Test aux limites** : il s'agit d'un test au cours duquel on va simuler en général une activité bien supérieure à l'activité normale, pour voir comment le système réagit aux limites du modèle d'usage de l'application.
- Il existe d'autres types de tests, plus ciblés en fonction des objectifs à atteindre dans la campagne de test.

Pour pouvoir calculer la disponibilité d'un système, il est nécessaire d'utiliser les notions suivantes :

- L'**uptime** désigne le temps de bon fonctionnement d'un système ou temps écoulé depuis le dernier démarrage ou le dernier « plantage ».
- Le **MTBF** (*Mean Time Between Failure*) est le temps moyen entre deux « plantages » pour un composant d'architecture.
- L'**AFR** (*Annualized Failure Rate*) est l'inverse du MTBF rapporté à une année, c'est-à-dire qu'il représente la proportion de composants à changer chaque année.
- Le **downtime** désigne le temps d'arrêt lié à un dysfonctionnement.
- Le **MTTR** (*Mean Time To Repair*) désigne le temps moyen nécessaire au rétablissement du service.
- L'**AST** (*Agreed Service Time*) désigne l'exigence de continuité de service convenue avec les utilisateurs du système.

Sur la base de ces définitions, on peut calculer la disponibilité de trois manières :

$$HA = \frac{MTBF}{MTBF + MTTR}$$

$$HA = \frac{Uptime}{Uptime + Downtime}$$

$$HA = \frac{AST - Downtime}{AST}$$

Figure 9 : Méthodes de calcul de la disponibilité

Quand à la robustesse d'un système, elle est constatée en production en réalisant un ratio entre le nombre de sollicitations traitées par un composant (requêtes) et le nombre d'erreurs obtenues.

### Sous-chapitre 3 – Les coûts induits par la recherche de la performance

Plus l'on souhaite obtenir un système informatique performant, plus le coût engendré par cette exigence devient important. On estime en informatique que le coût de la performance d'un système répond à une fonction exponentielle.

Il est donc primordial avant tout de déterminer quel est le niveau d'exigence d'une application en terme de performance avant de démarrer tout projet informatique, et d'autant plus, toute campagne de test de performance.

Sur le plan de la disponibilité des systèmes, l'atteinte d'un très haut niveau de disponibilité est très coûteuse en termes de matériel, de logiciels et de surveillance humaine. Le surcoût engendré par cette recherche du meilleur taux de disponibilité doit être mise en relation avec les coûts engendrés par une indisponibilité du système (cf. tableau ci-dessous).

Secteur d'activité	Coût moyen par heure
Banque d'investissement	6,48 M\$
Télécom	2,00 M\$
Web marchand	1,1 M\$
Pharmacie	1,0 M\$
Chimie	0,7 M\$
Réservation aérienne	90 k\$

**Tableau 3 : Exemples de coûts d'indisponibilité pour des applications critiques**

L'arbitrage doit alors être fait sur des critères rationnels de coût, en effet la recherche d'une disponibilité idéale sans réelle contrainte métier n'aurait que peu de sens devant les coûts engendrés (cf. courbe ci-dessous).

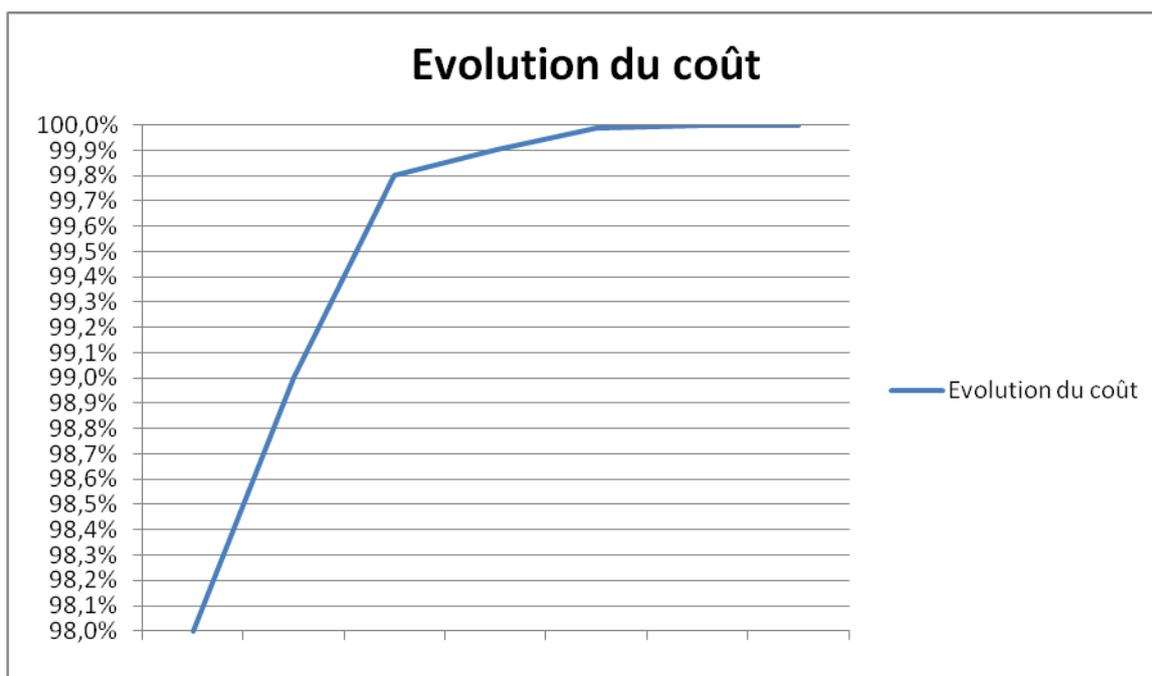


Figure 10 : Caractère exponentiel des coûts de haute disponibilité

La mise en place d'une plateforme de supervision / gouvernance du SI permet alors d'augmenter la disponibilité du système en améliorant la connaissance de celui-ci par les équipes de maintenance (cartographie des composants) et en avertissant au plus tôt lors de toute indisponibilité de l'un des composants.



## **Partie 2**

### **La définition d'une plateforme de supervision**

## Chapitre 1 – Les différents types de solution de supervision

A l'heure actuelle de nombreux éditeurs logiciels proposent des solutions de supervision ne couvrant pour la plupart qu'une partie des enjeux de la supervision du système d'information. Ces différents types de solutions logiciels sont notamment représentés par les sigles :

- ▶ **BAM** pour **B**usiness **A**ctivity **M**onitoring (Supervision de l'activité métier)
- ▶ **BSM** pour **B**usiness **S**ervice **M**anagement (Gestion des services métiers)
- ▶ **NSM** pour **N**etwork **S**ystems **M**anagement (Gestion des systèmes techniques et réseaux)

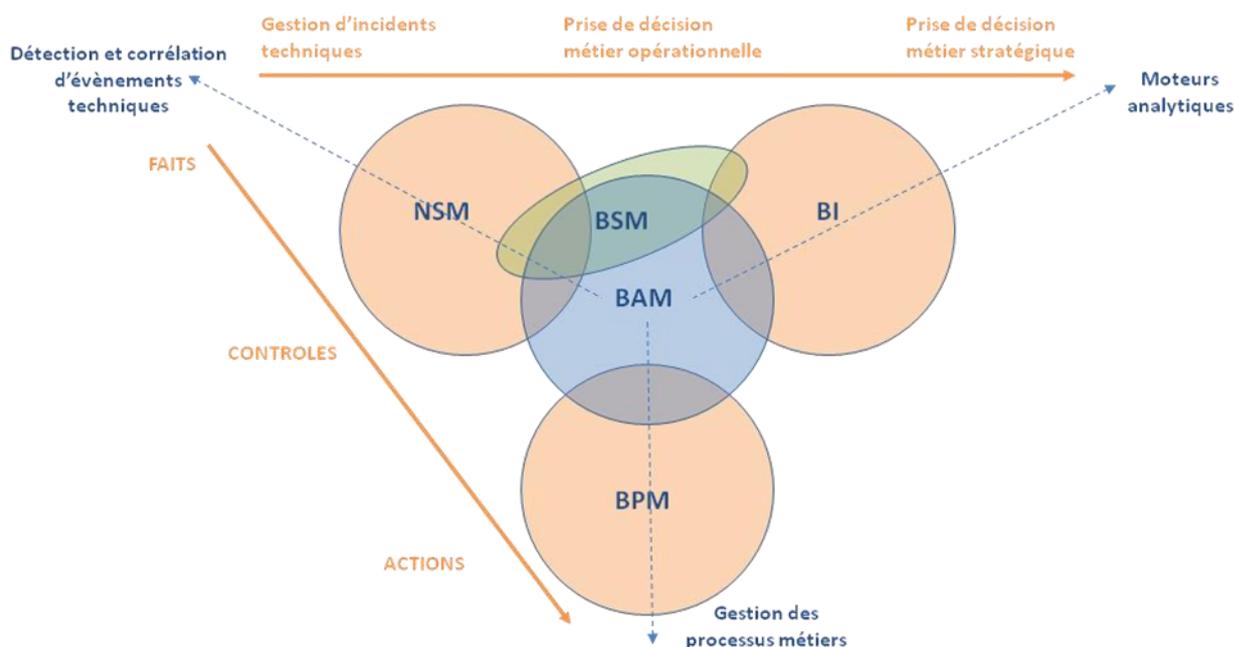


Figure 11 : Périmètre des différentes solutions de supervision

### Sous-chapitre 1 – Les outils de type BAM

#### Définition et objectifs

Un système de type BAM permet la mesure en quasi-temps réel des indicateurs de performance et des métriques permettant de juger du bon fonctionnement général des processus de l'entreprise et de l'efficacité de la gestion des activités métier.

La mise en place d'une solution de type BAM s'inscrit le plus souvent dans une démarche d'amélioration continue en permettant d'utiliser les indicateurs fournis pour être réactif face aux dysfonctionnements, mais aussi de tirer des enseignements sur leur origine afin de mettre en place des actions d'optimisation sur le long terme (amélioration d'un processus métier, automatisation d'une tâche...).

Quatre objectifs supplémentaires sont attribués aux solutions de type BAM en fonction de leur couverture du besoin :

- ▶ **Permettre une vision d'ensemble** sur le déroulement d'activités métier impliquant clients et partenaires de l'entreprise.
- ▶ **Faciliter le contrôle du bon déroulement des activités métiers** en mesurant en quasi temps réel les indicateurs de performance (KPI).
- ▶ **Assister les équipes de supervision dans le support et la qualification des incidents** observés sur les processus métiers en permettant le suivi du déroulement de chaque instance d'un processus métier.
- ▶ **Permettre l'amélioration continue des processus métiers** en offrant la capacité de suivre en quasi temps réel les impacts d'une politique d'amélioration continue.

### Utilisateurs finaux

Au sein d'une direction métier, un outil de type BAM est destiné aux :

- ▶ **Responsables métier** souhaitant avoir une vision de l'efficacité des activités et processus métiers sous leur responsabilité.
- ▶ **Responsables opérationnels** souhaitant identifier rapidement une situation de dysfonctionnement métier.

Au sein d'une direction informatique, un outil de type BAM est destiné aux :

- ▶ **Groupes de support informatique** ayant à leur charge d'identifier la cause de problèmes métiers constatés.
- ▶ **Equipes d'exploitation / production** ayant sous responsabilité d'évaluer l'impact de problèmes techniques liés aux applications et flux d'échanges.

### Fonctionnalités couvertes

Afin de remplir pleinement son rôle, une solution de type BAM doit proposer les fonctionnalités suivantes :

- ▶ **Supervision temps-réel des processus métiers**, permettant de suivre leur bonne exécution de manière massive ou bien unitairement pour chaque objet du SI mis sous surveillance (processus métier, service, flux de données...).
- ▶ **Pilotage temps-réel des indicateurs de performance**, ces indicateurs pouvant être de formes multiples : durée d'exécution d'une ou plusieurs étapes d'un processus, volumes d'objets dans un statut donné, taux d'erreur, agrégation de montants financiers traités par un processus métier...
- ▶ **Fonction d'aide à l'identification, à la qualification et à la résolution des dysfonctionnements**, il s'agit par exemple de fonctionnalités d'analyse de cause, d'identification des instances de processus en erreur, d'analyse d'impact d'un dysfonctionnement sur un processus donné.

- ▶ **Fonction d'analyse de données et de restitution graphique**, ces fonctionnalités doivent permettre la composition de graphiques de présentation de données métiers (taux d'erreur, montant des commandes traitées par le processus dans la journée...). Ces graphiques doivent être facilement intégrables au sein du portail de l'entreprise.

## **Sous-chapitre 2 – Les outils de type BSM**

### **Définition et objectifs**

Un système de type BSM permet la supervision et la gestion de la qualité des services métiers délivrés par un système d'information.

L'objectif d'un système de type BSM consiste alors à proposer aux exploitants des interfaces de reporting temps réel permettant de suivre la qualité de service des différentes composantes du système d'information de l'entreprise concernée.

## **Sous-chapitre 3 – Les outils de type NSM**

### **Définition et objectifs**

Un système de type NSM permet la gestion et le suivi de l'état des composants réseaux constituant un système informatique.

Ces outils principalement basés sur le protocole SNMP (Simple Network Management Protocol) permettent de diagnostiquer au plus tôt tout problème lié aux équipements réseaux ainsi que de superviser l'utilisation de ceux-ci.

Les outils de type NSM sont donc limités à la couche technique d'un système d'information.

## **Sous-chapitre 4 – Les solutions du marché**

De nombreuses solutions éditeurs existent sur le marché, elles sont portées principalement par les éditeurs de plateformes SOA (IBM, Oracle, BEA, Software AG, Progress Software...).

### **Offre Progress Software® « Actional »**

Couverture de offre « Actional » commercialisée par Progress Software :

- ▶ **Actional Diagnostics** : Améliore la performance, qualité et fiabilité des services que vous développez (SOA, REST et POX).

- ▶ **Actional for Active Policy Enforcement** : Permet une gestion centralisée des politiques SOA en termes de norme de sécurité et de conformité, tout en assurant l'exécution des politiques distribuées.
- ▶ **Actional for Continuous Service Optimization** : Permet à l'informatique d'aligner les opérations SOA sur les besoins métiers en assurant une qualité de service pour les clients et les utilisateurs finaux SOA. Il fournit un aperçu des opérations SOA pour réaliser et prioriser les décisions et permettre l'intégration de contrôles en temps réel pour permettre l'optimisation en continu des résultats métiers.
- ▶ **Actional for SOA Operations** : Permet pour des opérations de bout en bout, la visibilité au sein de votre architecture SOA, d'accélérer la résolution des problèmes et d'atténuer les risques de non adoption de la SOA.
- ▶ **Actional for Sonic ESB Management** : Permet d'étendre les capacités de gestion opérationnelle de l'ESB Sonic de Progress en fournissant de bout en bout, la visibilité et le contrôle des déploiements complexes ESB et les ressources de connexion.

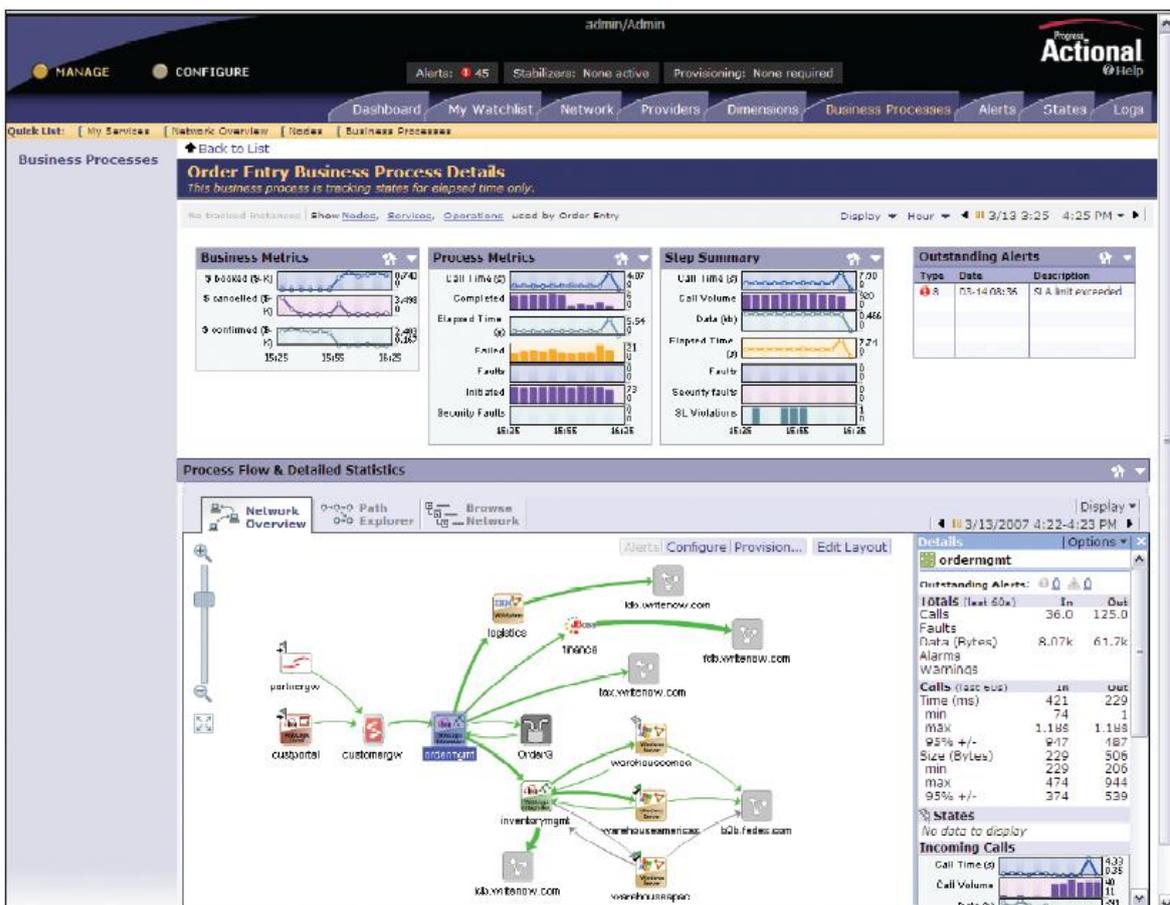


Figure 12 : Aperçu de l'outil Actional commercialisé par Progress Software

## Offre de supervision de Systar®

### Couverture de l'offre de supervision de Systar :

- ▶ **ServiceVision** : Pour chaque service ou processus, les tableaux de bord de ServiceVision permettent de suivre en temps réel et de bout en bout la bonne exécution du service et de ses étapes clés en fonction des résultats attendus.
  - **Dynamique Applicative** : Tableau de bord surveillant la progression des flux d'objets techniques (fichiers, messages) supportant le service à travers les différentes applications.
  - **Infrastructure** : Tableau de bord indiquant la disponibilité des composants techniques supportant le processus. Il consolide les informations hétérogènes en provenance des outils de monitoring techniques déjà en place pour donner une visibilité transversale.
  - **Service délivré** : Tableau de bord évaluant la qualité de service délivré par l'informatique aux utilisateurs métiers conformément aux engagements.
  
- ▶ **WideVision Alert** : Solution de supervision en temps réel qui analyse les alertes et événements significatifs en provenance de votre système d'information et identifie leur impact sur les services métiers et les conventions de service (SLA). Il s'intègre sans effort à l'infrastructure existante en capturant l'ensemble des événements en provenance des outils de gestion de systèmes et réseaux (NSM) ou de gestion de performance des applications (APM) et en interagissant avec les solutions de gestion de configuration, de gestion de parcs et de gestion d'incidents.
  
- ▶ **WideVision Météo des services** : Cette solution permet de contrôler en permanence le fonctionnement de tous les services à travers une analyse « Top-Down » grâce à une librairie d'indicateurs (disponibilité des composants, temps de réponse, réception de fichiers, bonne fin des sauvegardes)

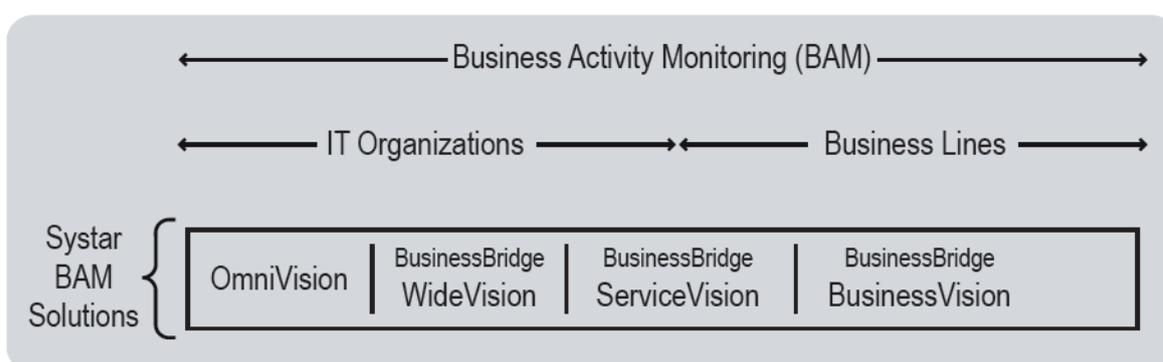


Figure 13 : Couverture de l'offre de supervision de Systar

## Chapitre 2 – La solution conçue par Axopen

Nous avons vu dans les chapitres précédents le rôle du SI dans la performance des entreprises ainsi que le besoin de supervision et de gouvernance de l'ensemble. Des solutions propriétaires ou libres existent actuellement sur le marché, mais aucune d'entre elles ne proposent de faire le lien entre ces trois domaines de la supervision.

Pour répondre à ce besoin, nous avons décidé de développer notre propre solution logicielle qui permet d'assister les équipes informatiques dans leur travail quotidien de supervision, maintenance, évolution et gouvernance du SI.

### Sous-chapitre 1 – Cahier des charges de la solution

#### Liste des exigences

Afin de permettre à la solution de s'adapter facilement au contexte des futurs clients, la solution de supervision et de gouvernance du SI devra être conforme aux exigences suivantes :

- ▶ **Performances** : la solution de supervision ne devra pas entraîner un surcoût en termes de performances lors de la mise en place au sein d'un SI (Asynchronisme via JMS).
- ▶ **Indépendance** : la solution devra être aussi indépendante que possible des technologies mises en œuvre au sein du SI des clients afin de permettre une adaptation aisée aux différents contextes clients.
- ▶ **Coûts de réalisation** : les coûts de développements devront être limités en utilisant un Framework et des outils de développement permettant une productivité maximale.
- ▶ **Portabilité OS** : la portabilité OS (Operating System) de la solution devra permettre une adaptation aisée aux différents contextes clients.
- ▶ **Approche modulaire** : une approche modulaire permettrait une mise en place progressive et adaptée de cette solution en fonction des priorités et niveau des enjeux métiers de chaque client.
- ▶ **Généricité** : la solution devra être complète et autosuffisante, respecter les standards et les normes en vigueur et être adaptée au suivi de toute l'infrastructure d'un SI (ESB, ETL, Web Services, Brokers JMS, Base de données, Serveur d'applications...).
- ▶ **Ouverture** : la solution devra proposer des interfaces d'échanges normalisées permettant l'échange avec différents types de flux (Web Services, messages JMS, fichiers plats...).
- ▶ **Expérience utilisateur** : la solution devra disposer d'interfaces de gestion intuitives et adaptées aux différentes populations utilisatrices.

## Définition du périmètre de la solution

Par rapport au périmètre « classique » d'une solution de type BAM, la solution devra proposer des écrans de supervision de l'infrastructure physique, d'analyse métier de l'utilisation des applications ainsi que de gouvernance des actifs immatériels du système d'information (services, processus, flux d'échanges, applications, vues...).

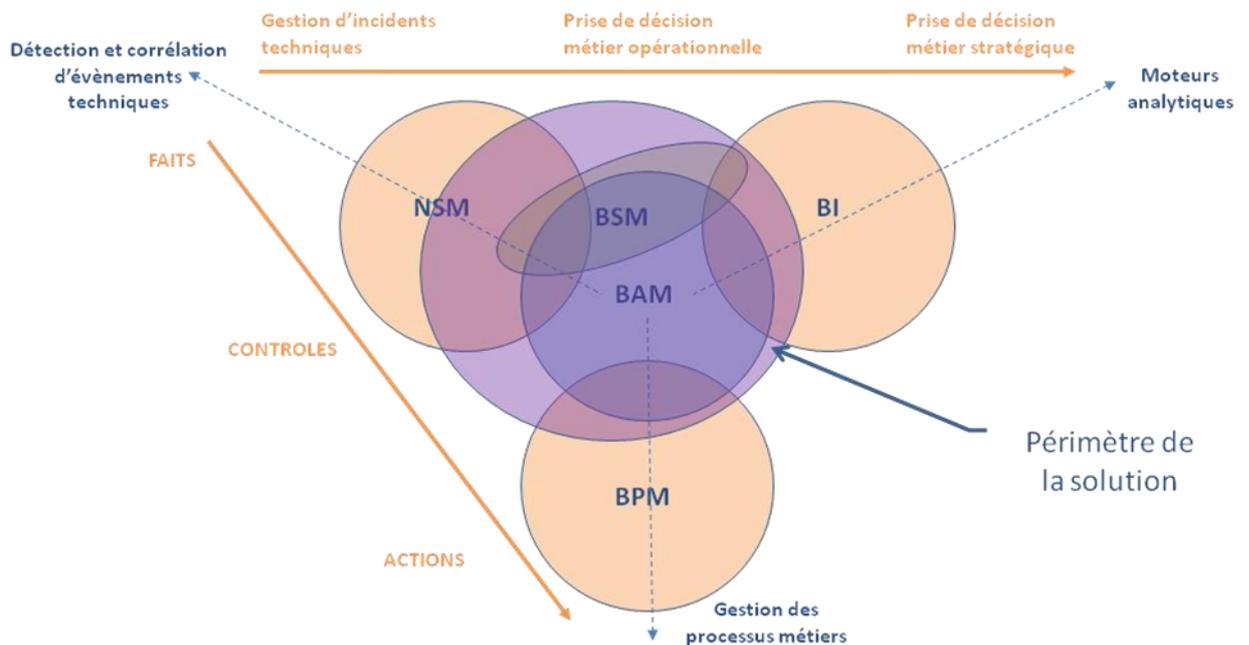


Figure 14 : Périmètre de la solution

## Ma contribution au projet

Outre le sponsoring de ce projet auprès de ma hiérarchie, mon rôle inclut la responsabilité des tâches suivantes :

- ▶ Conception des écrans et maquettage.
- ▶ Conception du schéma de base de données.
- ▶ Architecture de la solution en partenariat avec Philippe AUBERTIN et Pierre-Denis VANDUYNSLAGER.
- ▶ Ecriture des scripts de génération des tables Oracle (projet Java *ISM.Generator.SQL*).
- ▶ Développement des entités métiers Java (projets Java *ISM.Generator.Entity* et *ISM.Java.Entity*).
- ▶ Développement des managers Java (projets Java *ISM.Generator.Manager* et *ISM.Java.Manager*).
- ▶ Développement de la solution de collecte des évènements (projet Java *ISM.Event.Collector*).
- ▶ Développement du Framework technique Java comprenant les classes utilitaires permettant la gestion de la base de données, des fichiers, du bus JMS, de la sérialisation XML... (projet Java *FWK.Java.Framework*).
- ▶ Développement du moteur de lancement des tâches comme par exemple l'envoi des alertes par e-mail ou SMS ou la purge des bases de données (projet Java *ISM.Engine.Task*).

Le développement du moteur de corrélation ainsi que l'interface de l'application sont confiés à d'autres membres de l'équipe.

## Sous-chapitre 2 – Maquette des écrans de la solution

Afin d'adapter le contenu à l'utilisateur, l'application devra afficher un tableau de bord spécifique en fonction du profil connecté à l'application. Par défaut, nous proposerons les profils suivants :

- ▶ Administration
- ▶ Conception
- ▶ Exploitation
- ▶ Gouvernance
- ▶ Etude métier

Le choix du profil constitue l'écran d'accueil à l'application, dans cet écran (cf. aperçu ci-dessous) l'utilisateur en fonction de ses droits peut accéder à une des facettes de l'application.



Figure 15 : Maquette de l'écran de choix d'un profil

En cliquant sur l'un des profils de l'application l'utilisateur accède alors au tableau de bord de ce profil. Le tableau de bord regroupe pour chaque profil les statistiques clés les concernant en offrant la possibilité de modifier les critères d'affichage de ces statistiques.

### Ecrans de gouvernance

Le tableau de bord de gouvernance permet d'offrir une vue synthétique des modifications survenues sur le système depuis la dernière connexion de l'utilisateur sur cet écran (cf. aperçu ci-dessous).

Ainsi l'utilisateur peut observer depuis sa dernière connexion à l'application :

- ▶ Le nombre de composants logiques (processus, écran...) créés, modifiés ou supprimés.
- ▶ Le nombre d'instances de composants logiques créés, modifiés ou supprimés.
- ▶ Le nombre de composants techniques (serveur, base de données...) créés, modifiés ou supprimés.
- ▶ Le nombre d'évènements survenus sur le système.
- ▶ Le nombre d'alertes remontées par le système.
- ▶ La période d'indisponibilité cumulée du système.
- ▶ Le nombre d'utilisateurs de l'application créés, modifiés ou supprimés.

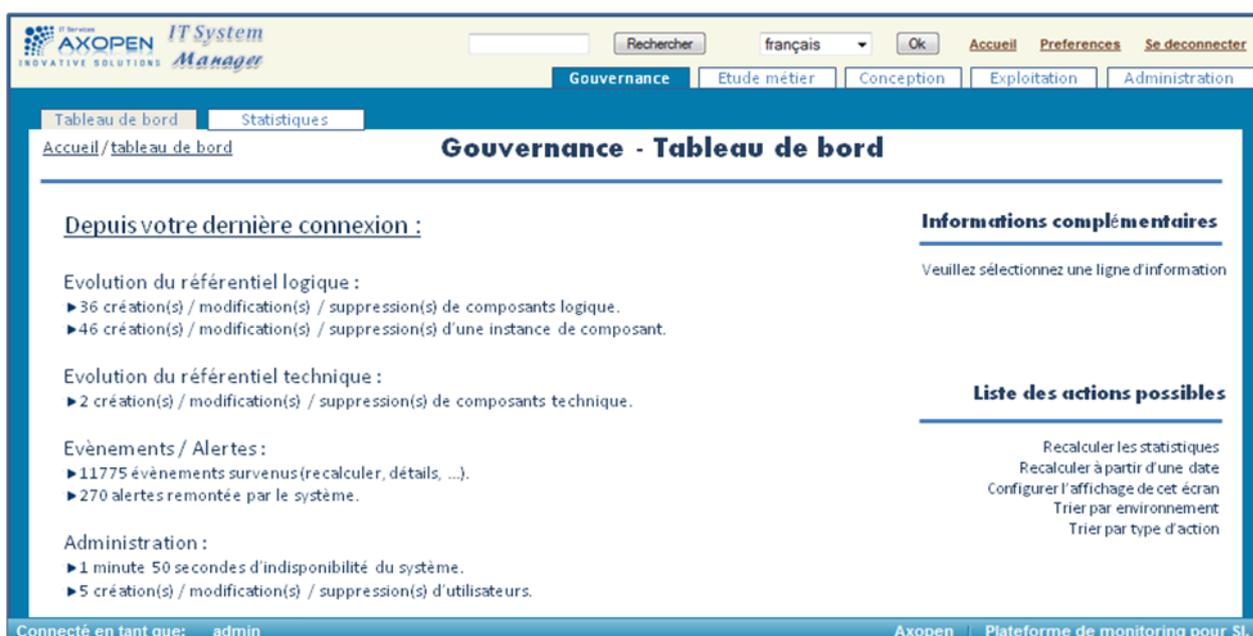


Figure 16 : Maquette de l'écran « Gouvernance - Tableau de bord »

L'écran statistique permet à l'utilisateur d'obtenir un certain nombre de statistiques en fonction d'une plage de temps, d'un type de composant mais surtout d'un certain nombre de critères disponibles en fonction du type de composant sélectionné.

L'utilisateur peut par exemple visualiser l'évolution du temps de réponse moyen d'une instance de composant en fonction de l'évolution du nombre d'utilisateurs simultanés (cf. aperçu ci-dessous).



Figure 17 : Maquette de l'écran « Statistiques de gouvernance »

L'application propose au profil « Gouvernance » uniquement ces deux écrans qui permettent d'obtenir des informations sur l'évolution et le fonctionnement du SI.

Ce profil s'adresse aux utilisateurs disposant de postes à responsabilité (DSI, directeur de programme...) dont l'apport est plus stratégique qu'opérationnel. Ce profil ne permet donc aucune action de création d'objets dans le système.

### Ecrans d'étude métier

Le profil « Etude métier » s'adresse aux utilisateurs faisant partit de la MOA (Maîtrise d'ouvrage), les fonctionnalités sous-jacentes s'orientent vers de la consultation de statistiques permettant de comprendre l'utilisation des applications mais aussi le suivi des différents chantiers menés par la MOE (Maîtrise d'œuvre).

Ainsi le tableau de bord de ce profil (cf. aperçu ci-dessous) propose à l'utilisateur les statistiques depuis sa dernière connexion à l'application :

- ▶ Le nombre des connexions et évolution de ce nombre sur le SI.
- ▶ Le temps cumulé d'utilisation des applications et évolution de ce temps sur le SI.
- ▶ Le nombre de nouvelles applications mises en production.
- ▶ Le nombre d'applications ayant changé de version.
- ▶ Le nombre d'évènements survenus sur le système.
- ▶ Le nombre d'alertes remontées par le système.

- ▶ Le nombre d’instances de service ayant dépassé le temps de réponse maximum admissible défini dans la SLA<sup>6</sup>.
- ▶ La période d’indisponibilité cumulée du système.

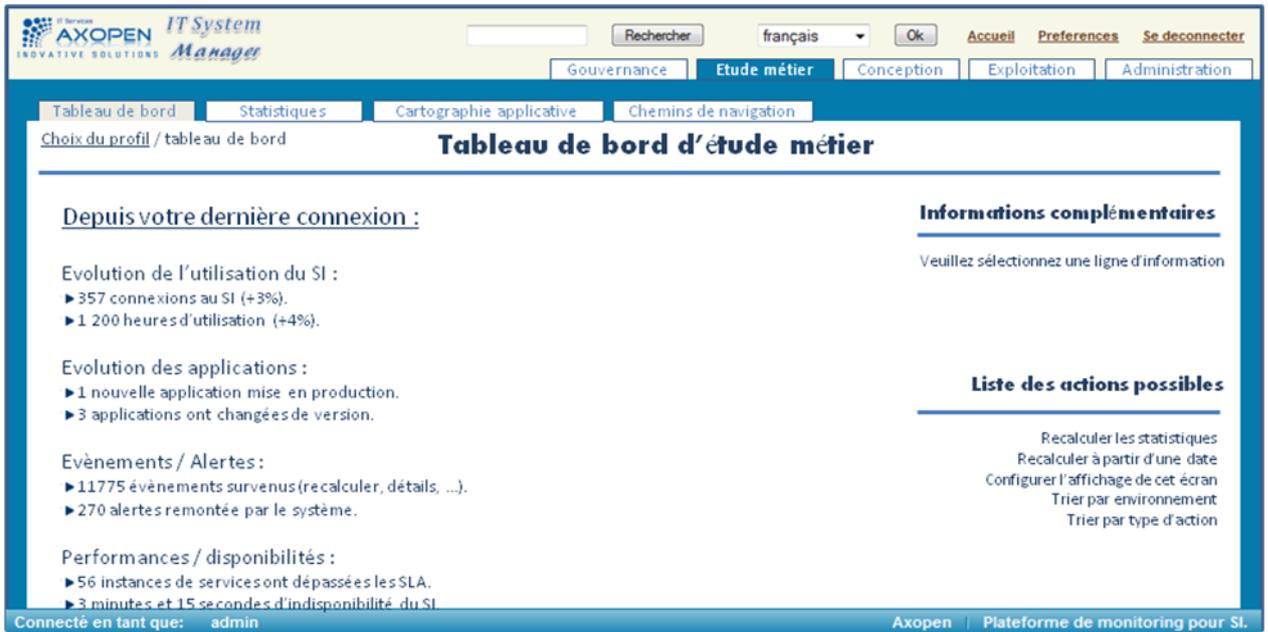


Figure 18 : Maquette de l’écran « Tableau de bord d’étude métier »

L’écran statistique permet à l’utilisateur d’obtenir un certain nombre de statistiques en fonction d’une plage de temps, d’un domaine fonctionnel, d’un processus métier mais surtout d’une application sélectionnée.

L’utilisateur peut alors observer l’évolution du nombre de connexions constatées sur l’application ainsi que la durée moyenne de connexion mais aussi la répartition du temps passé en fonction des écrans de l’application (cf. aperçu ci-dessous).

<sup>6</sup> **SLA** : Le *Service Level Agreement* est un document qui définit la qualité de service requise entre un prestataire et un client. Ce document que l’on pourrait traduire par Contrat de niveau de service est donc un contrat dans lequel on formalise la qualité du service en question (temps de réponse, nombre d’utilisateurs simultanés...).

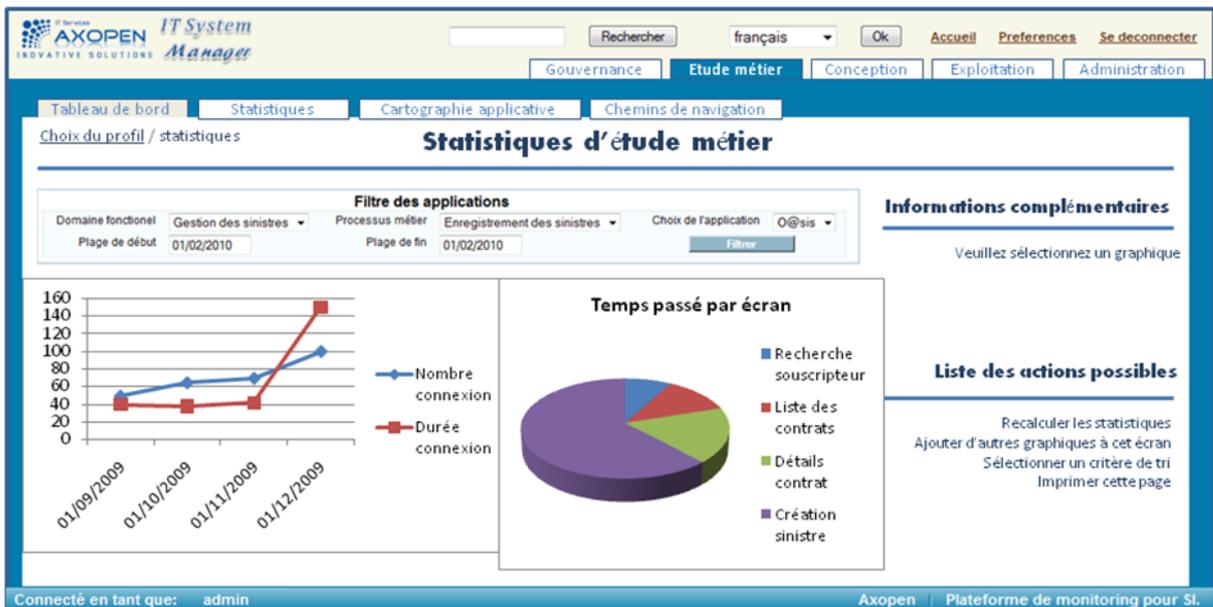


Figure 19 : Maquette de l'écran « Statistiques d'étude métier »

L'écran de cartographie des applications permet à l'utilisateur de naviguer dans la cartographie applicative en fonction d'un domaine fonctionnel ainsi que d'un processus métier. Il est alors possible de confronter la vision processus métier de l'entreprise aux applications couvrant ces processus (cf. aperçu ci-dessous).



Figure 20 : Maquette de l'écran « Cartographie des applications »

L'écran « Chemins de navigation » permet à l'utilisateur de visualiser pour une application les chemins de navigation les plus fréquents sur une période. En sélectionnant l'un des chemins de navigation l'utilisateur peut alors observer la liste des écrans de l'application traversé ainsi que la durée moyenne d'affichage de l'écran (cf. aperçu ci-dessous).

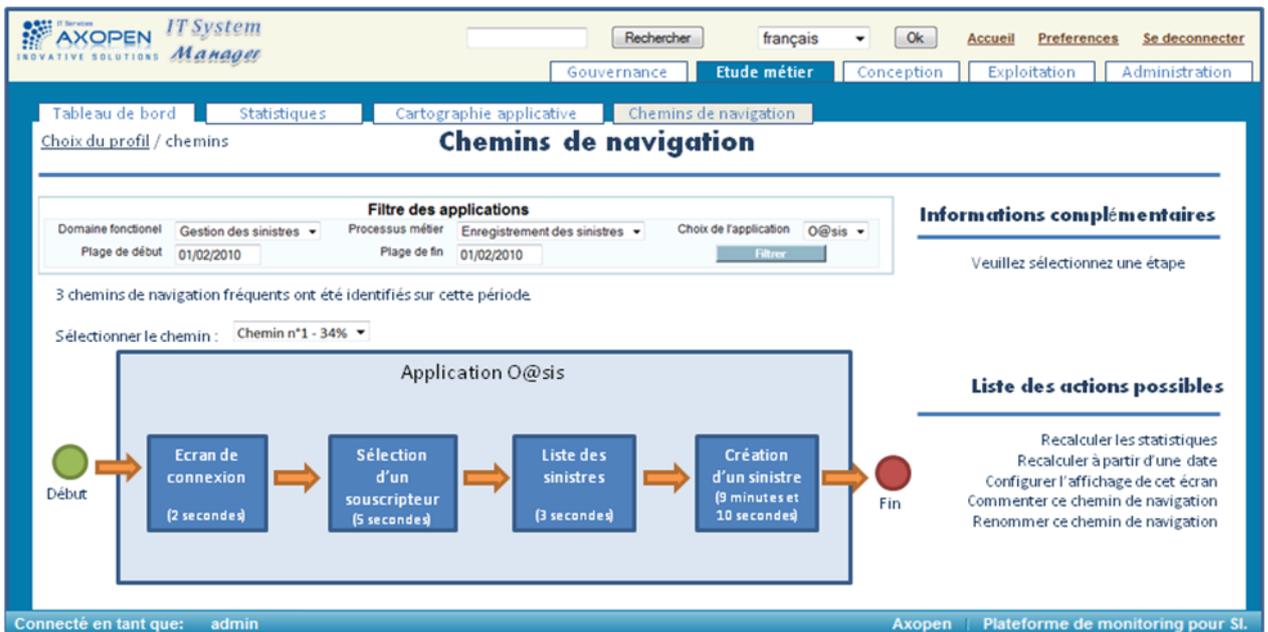


Figure 21 : Maquette de l'écran « Chemins de navigation »

## Ecrans de conception

Le profil « Conception » s'adresse aux utilisateurs faisant partie de la MOE (Maîtrise d'œuvre). L'application leur offre alors la possibilité de consulter et faire évoluer la cartographie des composants du système ainsi qu'observer le cycle de vie des objets.

Ainsi le tableau de bord de ce profil propose à l'utilisateur les statistiques depuis sa dernière connexion à l'application :

- ▶ Le nombre de composants logiques (processus, écran...) créés, modifiés ou supprimés.
- ▶ Le pourcentage de réutilisation des composants.
- ▶ Le nombre de composants ayant changé de version
- ▶ Le nombre d'évènements survenus sur le système.
- ▶ Le nombre d'alertes remontées par le système.
- ▶ Le nombre d'instances de service ayant dépassé le temps de réponse maximum admissible défini dans la SLA.

L'écran statistique permet à l'utilisateur d'obtenir un certain nombre de statistiques en fonction d'une plage de temps, d'un domaine fonctionnel et d'un composant.

L'utilisateur peut par exemple visualiser les performances ainsi que le nombre d'appel d'un composant au sein d'une application.

L'écran de « Cartographie des objets » permet à l'utilisateur d'observer l'ensemble des composants d'une application ainsi que les interactions éventuelles de ce composant au sein du processus métier, du domaine fonctionnel ou du reste du SI (cf. aperçu ci-dessous).

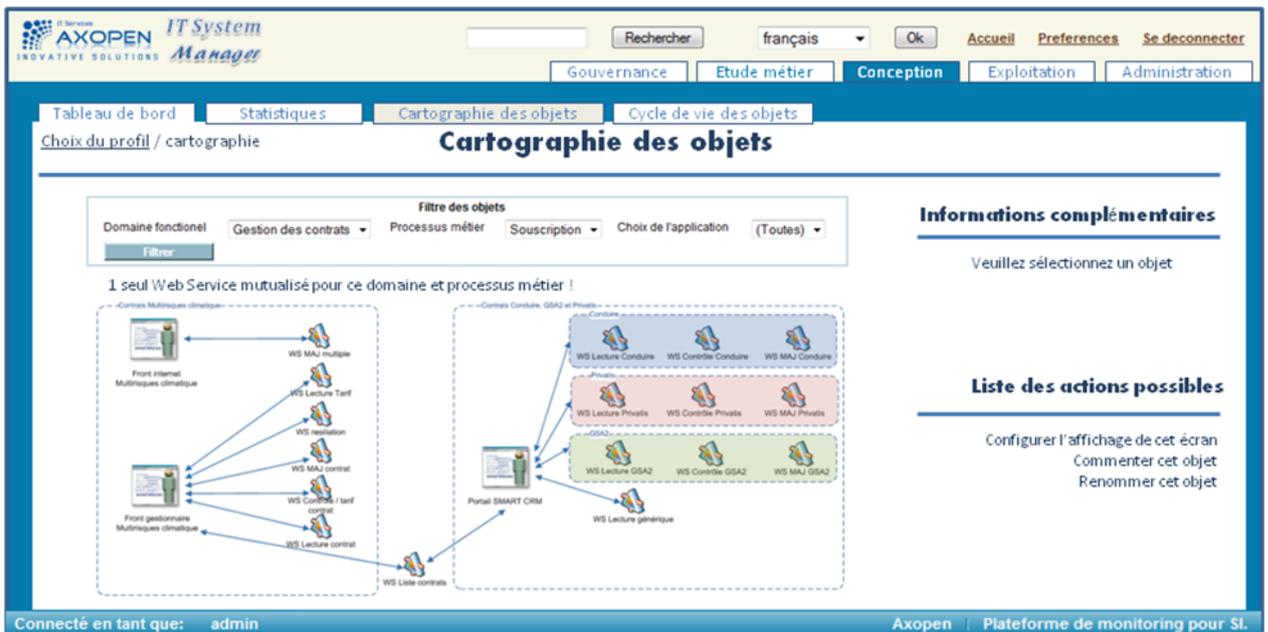


Figure 22 : Maquette de l'écran « Cartographie des objets »

L'écran de « cycle de vie des objets » permet d'observer pour un objet sélectionné au sein du SI l'ensemble des évolutions réalisées sur une période (cf. aperçu ci-dessous).

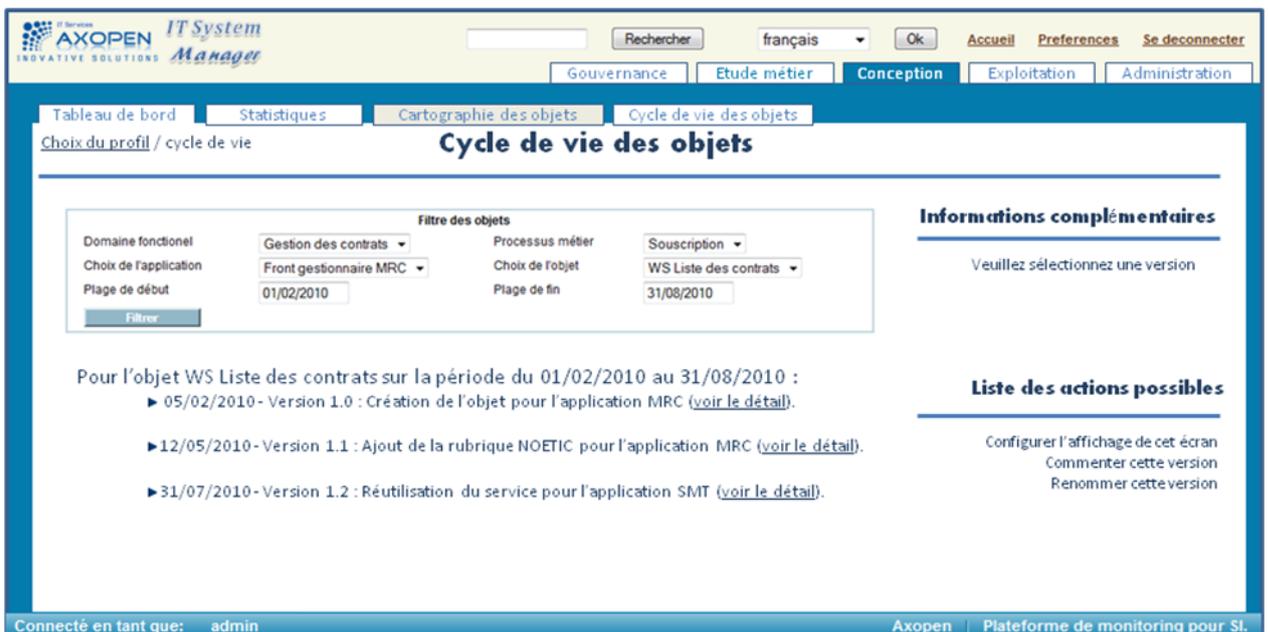


Figure 23 : Maquette de l'écran « Cycle de vie des objets »

## Ecrans d'exploitation

Le profil « Exploitation » s'adresse aux exploitants du SI en charge du bon fonctionnement de l'architecture physique en production. L'application leur offre alors la possibilité de consulter et faire évoluer la cartographie des serveurs physiques et logiques du SI ainsi que d'observer les statistiques de fonctionnement.

Ainsi, le tableau de bord de ce profil propose à l'utilisateur les statistiques depuis sa dernière connexion à l'application :

- ▶ Le nombre de serveurs physiques créés, modifiés ou supprimés.
- ▶ Le nombre de serveurs logiques créés, modifiés ou supprimés.
- ▶ Le nombre d'instances de composant créées, modifiées ou supprimées.
- ▶ Le nombre d'évènements survenus sur le système.
- ▶ Le nombre d'alertes remontées par le système.
- ▶ Le nombre d'instances de service ayant dépassé le temps de réponse maximum admissible défini dans la SLA.

L'écran statistique permet à l'utilisateur d'obtenir un certain nombre de statistiques en fonction d'une plage de temps, d'un environnement et d'un serveur logique ou physique.

L'utilisateur peut par exemple visualiser l'évolution des ressources CPU et RAM pour les serveurs physiques ou l'évolution des ressources de la JVM pour les serveurs logiques fonctionnant sur un environnement Java.

L'écran de « Cartographie des serveurs logiques » permet à l'utilisateur de répertorier l'ensemble des serveurs logiques pour un environnement donné (cf. aperçu ci-dessous). L'utilisateur peut alors sélectionner un serveur logique pour obtenir l'ensemble des statistiques d'exécution ainsi que l'ensemble des composants déployés.

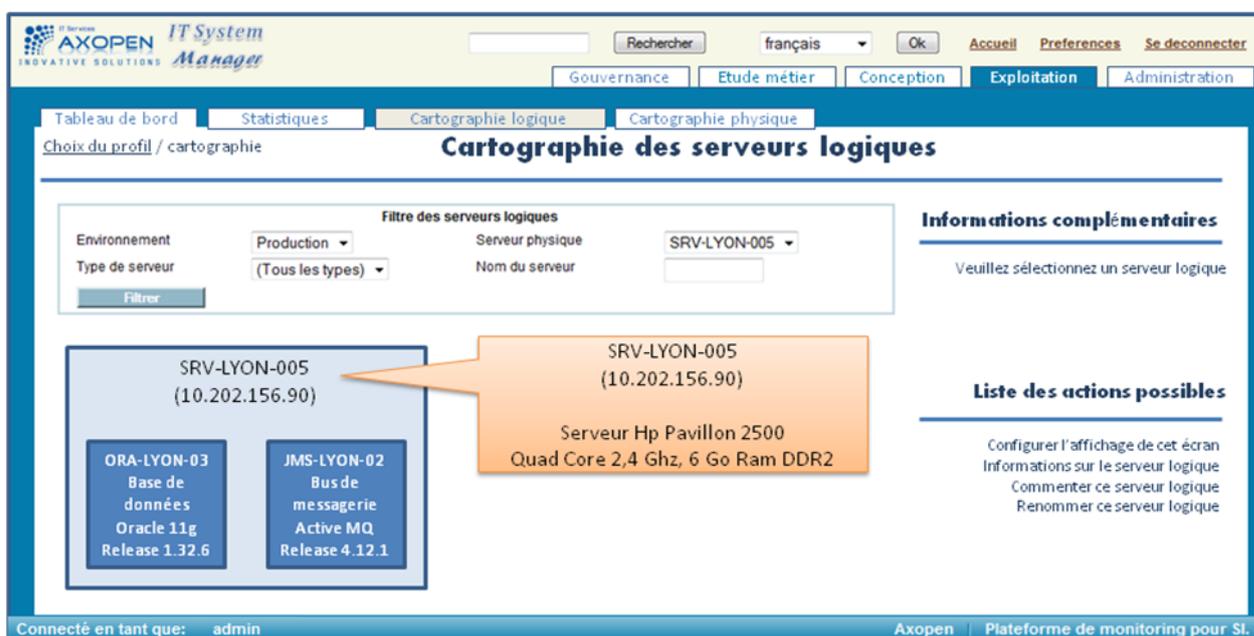


Figure 24 : Maquette de l'écran « Cartographie des serveurs logiques »

L'écran de « Cartographie des serveurs physiques » permet à l'utilisateur de répertorier l'ensemble des serveurs physiques pour un environnement donné en observant leur état de fonctionnement (cf. aperçu ci-dessous). L'utilisateur peut alors sélectionner un serveur physique pour obtenir l'ensemble des statistiques d'exécution de ce serveur, les serveurs logiques déployés ainsi que l'ensemble des composants déployés sur ces serveurs logiques (réutilisation de la vue serveurs logiques).

Figure 25 : Maquette de l'écran « Cartographie des serveurs physiques »

L'ensemble de ces maquettes ne détaille pas les phases où l'utilisateur sélectionne un objet. En effet le principe même de cette application est de permettre la navigation entre les différentes couches du SI (composant, instance, serveur logique...) afin de permettre une vision globale du système et ainsi en faciliter la maintenance et l'évolution.



## **Partie 3**

### **La construction d'une plateforme de supervision**

## Chapitre 1 – Architecture de la solution

L'architecture de la solution repose sur le principe de la répartition des responsabilités afin de permettre de distribuer la solution sur plusieurs serveurs physiques en mode mono-instance. Cette répartition permet aussi une meilleure évolutivité de la solution en rendant possible par exemple la mise en place de plusieurs technologies d'affichage en fonction du type de matériel concerné (ordinateur de bureau, téléphone portable...).

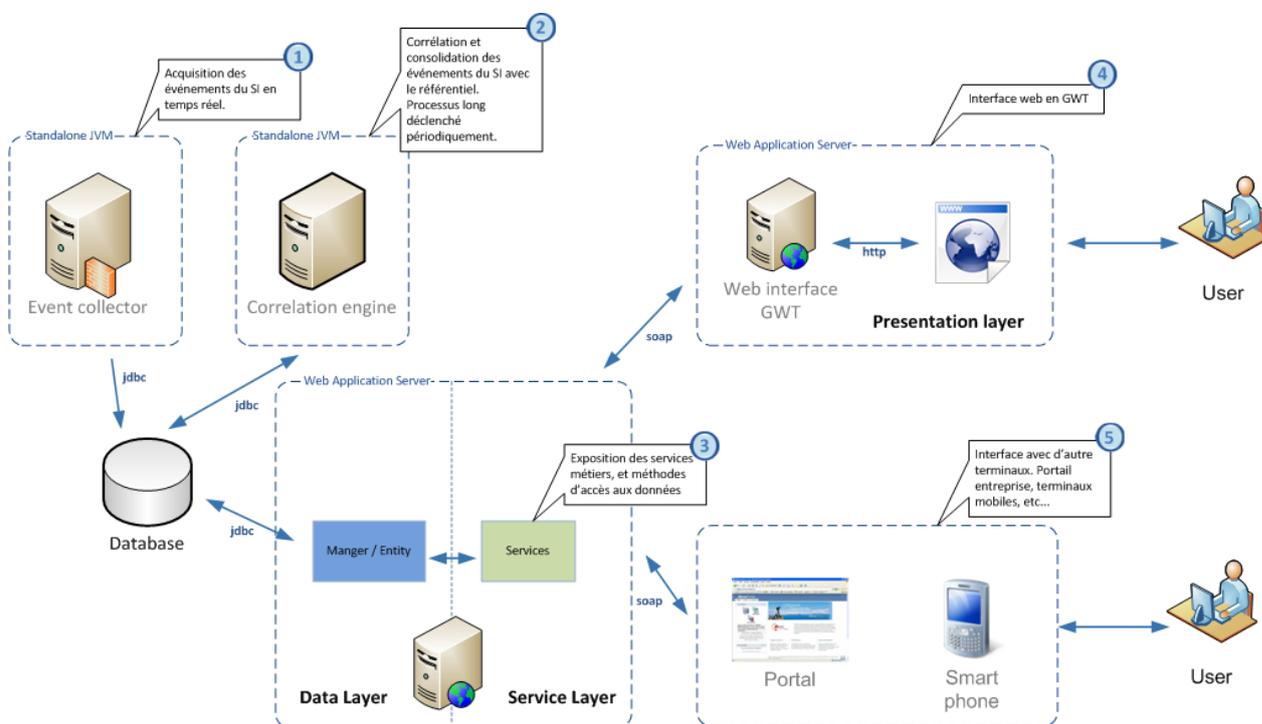


Figure 26 : Illustration de l'architecture logique de la solution

Les avantages des systèmes distribués découlent principalement d'exigences non fonctionnelles. Il s'agit en l'occurrence de contraintes d'ordre technique qui sont à l'origine de l'émergence de ce type d'architecture informatique.

Les avantages de ces architectures sont principalement :

- ▶ **Scalability** (extensibilité) – Les systèmes distribués permettent une expansion plus facile de la capacité de montée en charge du système si cela s'avère nécessaire.
- ▶ **Ouverture** – Les systèmes distribués peuvent être basés sur l'exposition de services possédant des interfaces bien définies basées sur des standards leur permettant d'être facilement extensibles et modifiables. Les *Web Services* sont un exemple de composant logiciel pouvant être distribuable et offrant une grande ouverture.
- ▶ **Hétérogénéité** – Les composants d'un système distribué peuvent être écrits en différents langages sur différentes machines. La communication entre les différents composants étant assurée par une couche de liaison appelé *middleware* (intergiciel).

- ▶ **Accès aux ressources et partage** – Les systèmes distribués fournissent un moyen de partager les ressources, c'est-à-dire à la fois le matériel, le logiciel et les données.
- ▶ **Tolérance aux pannes** – Les systèmes distribués peuvent être plus tolérants aux pannes que les systèmes centralisés, car ils permettent de répliquer facilement les composants logiciels sur plusieurs instances physiques.

Dans la pratique, l'ensemble des composants constituant un système distribué peut être déployé sur un serveur physique unique pour des raisons de coût, réparti sur autant de serveurs que de composants et même dans certains cas un composant peut être redondé sur plusieurs serveurs pour permettre la tolérance aux pannes (*fail over*) et/ou la répartition de charge (*load balancing*). Ces trois scénarios sont détaillés ci-dessous.

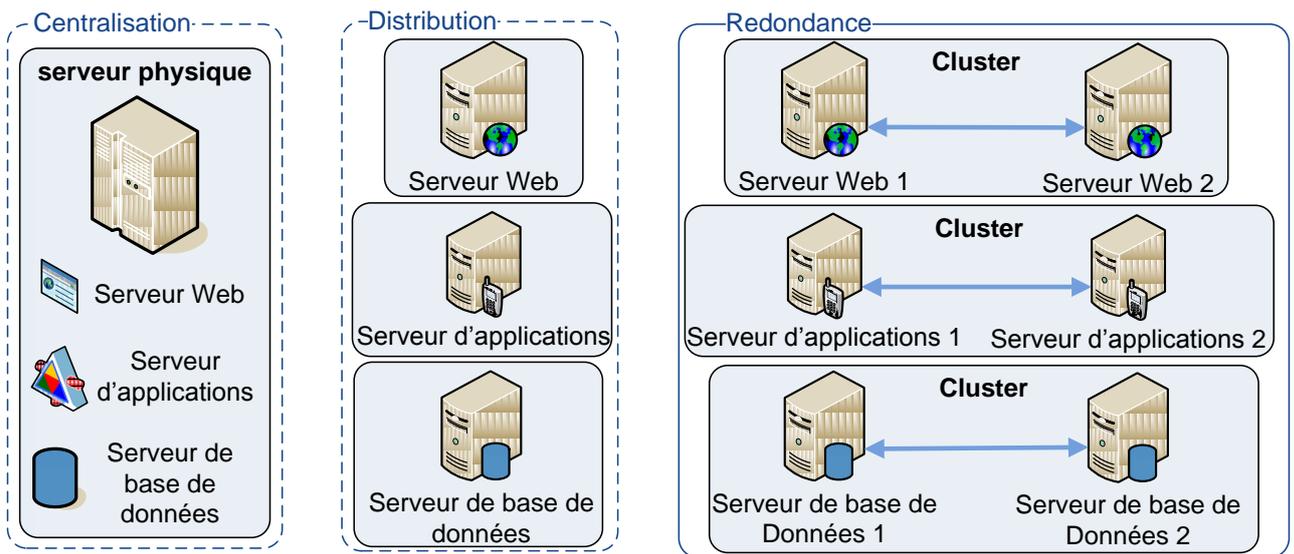


Figure 27 : Illustration des différents niveaux de distribution

Le travail des architectes informatiques consiste alors à proposer une architecture physique et logique au meilleur coût en fonction des contraintes et exigences du système concerné (temps de réponse, délai de reprise sur erreur, taux de disponibilité, volumétrie attendu, nombre d'utilisateurs simultanés...). Cette phase est communément appelée phase de *sizing*.

Les systèmes distribués disposant in fine d'une couche middleware pour assurer la communication entre les différents composants, des solutions fonctionnant sur les différentes couches (physique, logique et applicative) permettent d'augmenter plus facilement la performance globale de ces systèmes.

Le simple fait de répartir l'ensemble des composants d'un système distribué (serveur d'application, serveur de base de données, courtier de messagerie...) sur plusieurs serveurs physiques permet déjà à ce type de système d'offrir de meilleures performances que les systèmes centralisés où tous les composants sont centralisés sur un seul et même serveur physique.

## Sous-chapitre 1 – Composition du Framework

L'application est réalisée sur la base d'une architecture J2EE, un serveur d'application Open Source JBOSS et le Framework GWT de Google pour la partie affichage Web.

L'application est développée avec le langage de programmation Java et le stockage des informations est réalisé grâce à une base de données Oracle en version 11g.

L'application propose une interface Web Service et JMS (Java Message Service) pour la communication avec les applications supervisées.

L'application est développée en respectant le principe de séparation des couches logiques n-tiers (présentation, service, objets métier et données).

L'ensemble de ces choix d'architecture est détaillé dans les sous chapitres suivants.

### Serveur d'applications

Le serveur d'applications JBOSS a été créé par le français Marc Fleury (anciennement chez Sun). Il est distribué sous licence GPL depuis Mars 1999 et est soutenu par une communauté importante à travers le monde.

Ce serveur d'applications a été conçu pour tirer profit de la technologie EJB (Enterprise Java Beans). Il a été construit autour d'une infrastructure modulaire permettant ainsi le chargement dynamique des EJB sans avoir à redémarrer le serveur.

Le serveur d'applications JBoss se positionne comme l'un des leaders dans ce domaine, il est de loin le mieux reconnu dans le domaine des serveurs d'applications Open Source et concurrence les offres commerciales du marché les plus utilisées (IBM Websphere AS et Oracle WebLogic).

En effet, comme on peut le voir sur le graphique ci-dessous le Gartner<sup>7</sup> positionne l'offre JBoss comme l'un des leaders, juste en dessous des offres IBM, Oracle et Microsoft.

---

<sup>7</sup> Le **Gartner** est une entreprise américaine de conseil et de recherche dans le domaine des techniques avancées. Ayant environ 10 000 clients à travers le monde, elle mène des recherches, fournit des services de consultation, tient à jour différentes statistiques et maintient un service de nouvelles spécialisées.

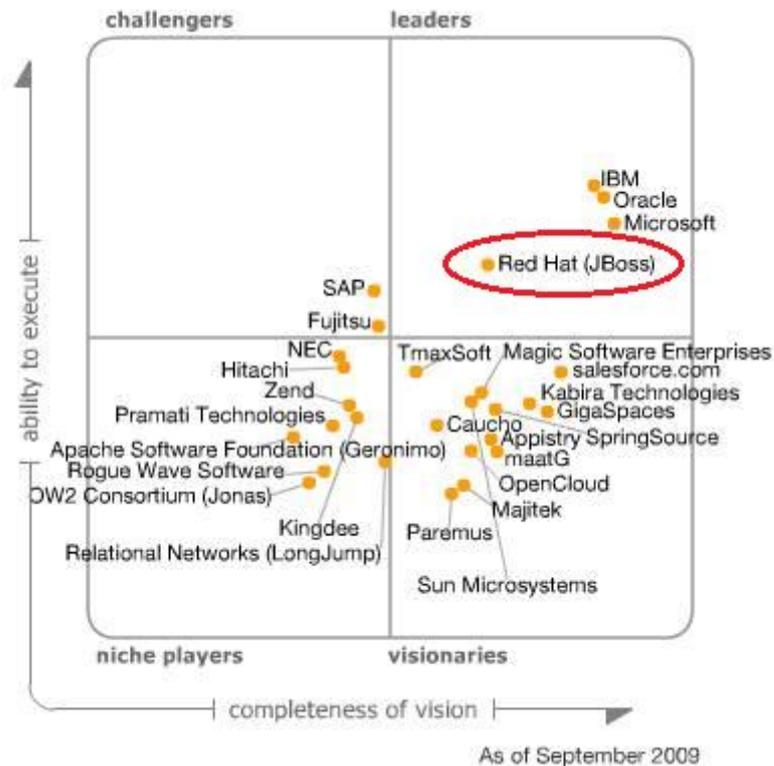


Figure 28 : Positionnement du serveur JBOSS sur le « cadrant magique » du Gartner

### Base de données

Oracle est un système de gestion de base de données relationnel (SGBDR) commercialisé par Oracle Corporation. La première version (nommée Oracle V2) a été proposée à la vente en 1979. Sur un marché pesant 19,6 milliards de dollars en 2007, Oracle totalise près de 46,5% des parts de marché et se place alors en première position devant IBM et ses 25,9%.

Les bases de données Oracle sont reconnues mondialement et sont présentes chez la majorité de nos clients. Nous avons donc décidé de construire la première version de notre application en s'appuyant sur cette base de données. Cependant, afin nous permettre de proposer par la suite des versions compatibles avec d'autres bases de données nous avons décidé de nous connecter à la base de données Oracle en utilisant l'API JDBC.

JDBC (Java DataBase Connectivity) est une API fournie avec Java (depuis sa version 1.1) permettant de se connecter à des bases de données, c'est-à-dire que JDBC constitue un ensemble de classes permettant de développer des applications capables de se connecter à des serveurs de bases de données.

L'API JDBC a été développée afin de permettre à un programme de se connecter à n'importe quelle base de données en utilisant la même syntaxe, l'API JDBC facilite donc l'indépendance vis-à-vis du SGBD.

De plus, JDBC bénéficie des avantages de Java, dont la portabilité du code, ce qui lui vaut en plus d'être indépendant de la base de données d'être indépendant de la plate-forme sur laquelle elle s'exécute.

Afin de permettre le changement de la base de données sans modification du code source nous avons externalisé la configuration de la connexion JDBC dans un fichier de propriété (cf. extrait ci-dessous).

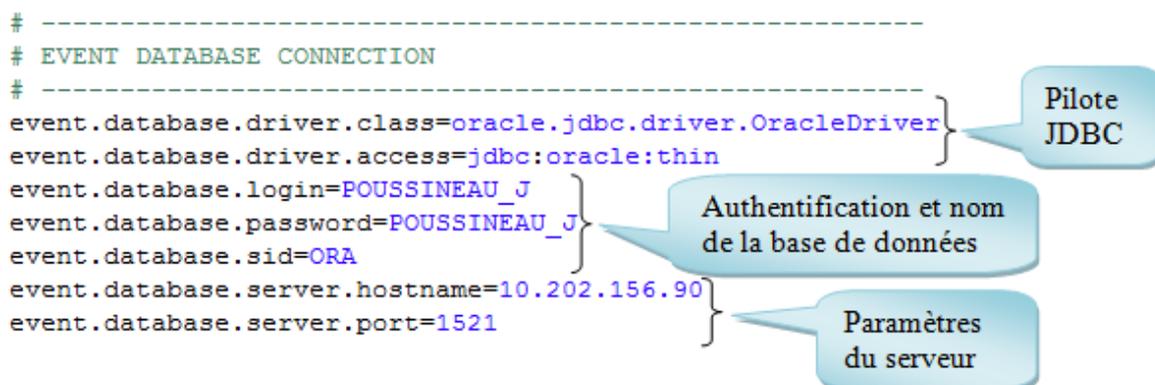


Figure 29 : Extrait du fichier propriété de configuration de l'accès JDBC

## Bus de messagerie JMS

JMS, acronyme de Java Messaging Service, est une API fournie par l'entreprise Sun pour permettre un dialogue standard entre des applications ou des composants via des *brokers* (courtiers en Français) de messages ou MOM (Middleware Oriented Messages). Elle permet donc d'utiliser des services de messaging dans des applications Java comme le fait l'API JDBC pour les bases de données.

Les *brokers* de messages permettent d'assurer l'échange de messages entre deux composants nommés clients. Ces échanges peuvent se faire dans un contexte interne (EAI) ou externe à l'entreprise (B2B).

Dans notre cas, nous utiliserons un *broker* de messagerie en entrée de l'application afin de recevoir les messages JMS émis par les différents agents déployés sur les serveurs à superviser. Afin de faciliter le déploiement du *broker* de messagerie nous avons décidé d'utiliser le *broker* Open Source Active MQ de la fondation Apache<sup>8</sup> au lieu du *broker* inclut dans le serveur d'application JBOSS et nommé JBOSS Messaging.

---

<sup>8</sup> L'*Apache Software Foundation* (Fondation Apache) est une organisation à but non lucratif qui développe des logiciels libres sous la licence Apache. Elle a été créée en Juin 1999 dans le Delaware aux Etats-Unis.

En effet Active MQ a été conçu pour fonctionner en *standalone*, c'est-à-dire en dehors du serveur d'application et avec sa propre JVM<sup>9</sup>. De plus Active MQ dispose d'une interface Web d'administration permettant de créer dynamiquement des files d'attente (*Queues*) et des files de diffusion (*Topics*).

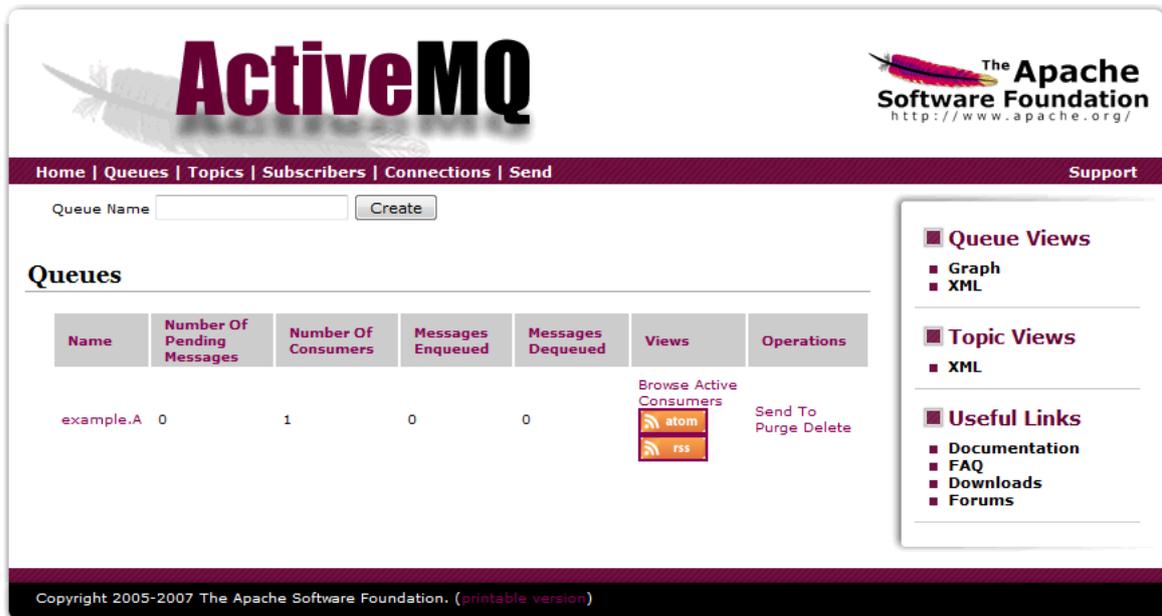


Figure 30 : Console Web d'administration d'Active MQ

## Framework technique et langage

Le Framework GWT (Google Web Toolkit) de Google est une boîte à outils permettant le développement rapide et l'optimisation d'applications de type Web. Le but de ce Framework est d'accroître la productivité des développements ainsi que la performance des applications en limitant la complexité pour les développeurs.

Le Framework GWT de Google se compose d'un SDK (Software Development Kit) offrant un nombre important d'API (Application Programming Interface) de programmation Java, d'un outil d'analyse des performances des applications (Speed Tracer) ainsi que d'un plugin Eclipse permettant le support du Framework GWT et du moteur d'application Web correspondant.

---

<sup>9</sup> La *Java Virtual Machine* (abrégé **JVM**) est une machine virtuelle permettant d'interpréter et d'exécuter le bytecode Java. Ce programme est spécifique à chaque plate-forme ou couple (machine / système d'exploitation) et permet aux applications compilées en bytecode de produire les mêmes résultats quelle que soit la plate-forme, tant que celle-ci est pourvue de la machine virtuelle Java adéquate.

Le processus de développement d'applications Web à l'aide du Framework GWT peut être matérialisé de la sorte :



Figure 31 : Processus de développement avec le Framework GWT

### Architecture n-tiers

Une architecture n-tiers comprend généralement une couche de présentation, une couche de services et d'objets métier et une couche d'accès aux données.

Certains ont l'habitude de qualifier cette architecture de 3 tiers, 4 tiers ou plus :

- ▶ La **couche de présentation** contient les différents types de clients, léger (Web, PHP, JSP...) ou lourd (Swing, WinForm...)
- ▶ La **couche de service** contient les traitements (contrôleurs de Use Case UML) représentant les règles métier (créer un compte, rechercher un client, calculer un amortissement, ...)
- ▶ La **couche d'objets métier** est représentée par les objets du domaine, c'est à dire l'ensemble des entités persistantes de l'application (Facture, Bon de Commande, Client, ...)
- ▶ La couche d'accès aux données contient les usines d'objets métier, c'est à dire les classes chargées de créer des objets métier de manière totalement transparente, indépendamment de leur mode de stockage (SGBDR, Objet, Fichiers, Legacy, ...)

La figure suivante illustre le principe :

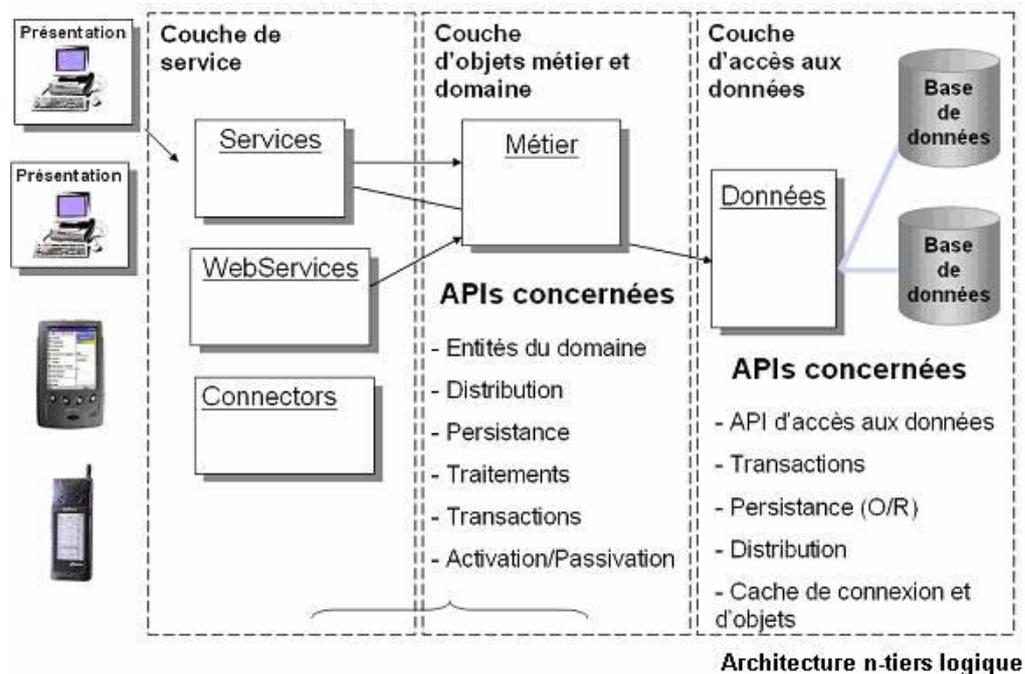


Figure 32 : Architecture n-tiers logique

Cette séparation par couches de responsabilités sert à découpler au maximum une couche de l'autre afin d'éviter l'impact d'évolutions futures de l'application. Imaginons le cas où il est nécessaire de remplacer une base de données relationnelle pour une base sous la forme de fichiers XML à plat, seule la couche d'accès aux données sera impactée, la couche de service et la couche de présentation ne seront pas concernées car elles ont été découplées des autres.

### Plateforme d'intégration continue

Dans le cadre du développement de l'ISM nous disposons de quatre environnements distincts :

- ▶ Postes de développement
- ▶ Serveur d'intégration
- ▶ Serveur de recette
- ▶ Serveur de test de performance

Le rôle de ces différents environnements est illustré par le diagramme ci-dessous :

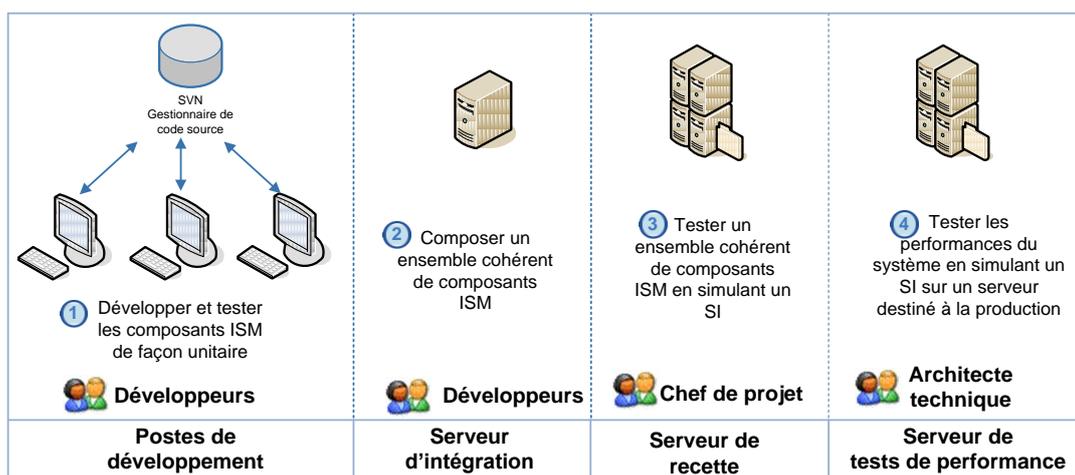


Figure 33 : Rôle des différents environnements

Afin de permettre le passage rapide des postes de développement au serveur d'intégration tout en certifiant la qualité du code nous avons mis en place une plateforme d'intégration continue. Cette plateforme basée sur l'outil Open Source Hudson<sup>10</sup> va récupérer toutes les nuits le code source à partir du gestionnaire de source SVN<sup>11</sup> et effectuer une compilation de celui-ci puis lancer un ensemble d'outils permettant de tester la qualité du code :

- ▶ Réutilisabilité des méthodes
- ▶ Fréquence des commentaires
- ▶ Optimisation des fonctions
- ▶ Etc...

Le fonctionnement de cette plateforme d'intégration continue est illustré par le schéma ci-dessous :

<sup>10</sup> **Hudson** est un outil Open Source d'intégration continue écrit en Java. Il s'interface avec des systèmes de gestion de versions tels que CVS et Subversion, et exécute des projets basés sur Apache Ant et Apache Maven aussi bien que des scripts arbitraires en Shell Unix ou Batch Windows.

<sup>11</sup> **SVN** (Subversion) est un système de gestion de versions. Il facilite le travail en équipe dans le cadre de développements informatiques en permettant le partage de codes sources et la gestion de versions.

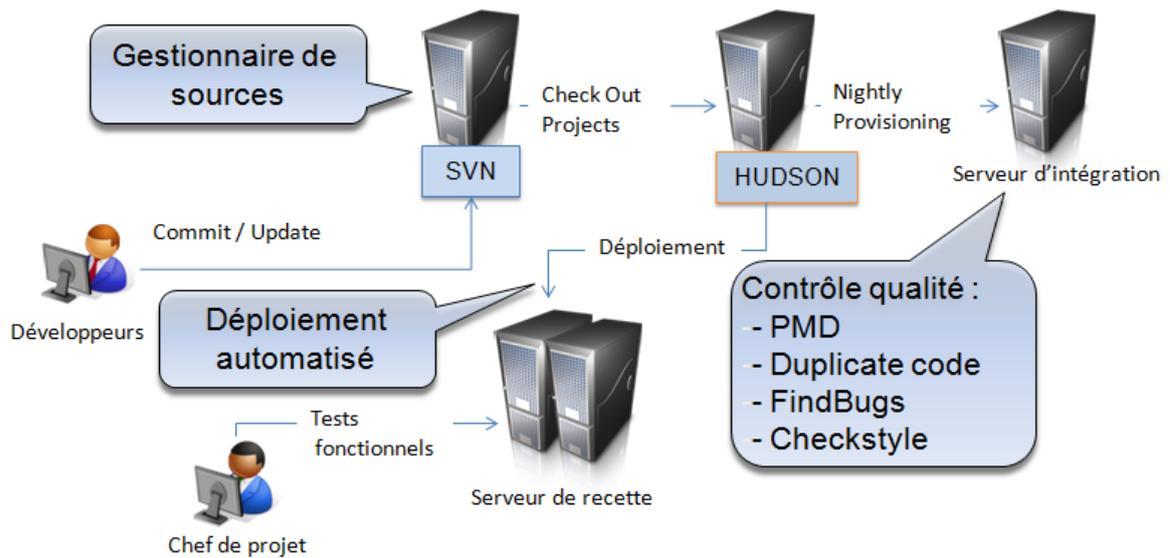


Figure 34 : Illustration du fonctionnement de la plateforme d'intégration continue

Si toutefois un développeur souhaite une intégration en cours de journée, Hudson propose une interface Web permettant de lancer manuellement une intégration et en observer le déroulement (cf. aperçu ci-dessous).

**Configuration Hudson**

**Lancer un job manuellement**

S	W	Job	Dernier succès	Dernier échec	Dernière durée
●	☁	1 - Test des services web	6 j 21 h (±21)	N/A	14 s
●	☁	2 - Test des services web MRC	6 j 18 h (±10)	6 j 18 h (±8)	46 s
●	☀	FWK.Java.Framework	16 h (±1)	N/A	18 s
●	☀	ISM.Collector.Event	9 h 51 mn (±80)	N/A	44 s
●	☀	ISM.Correlation.Enqaine	10 h (±113)	N/A	3 mn 52 s
●	☀	ISM.Generator.Entity	8 h 51 mn (±35)	N/A	18 s
●	☀	ISM.Java.Entity	10 h (±55)	N/A	1 mn 47 s
●	☀	ISM.Web.Client	1 mo. 3 j (±2)	N/A	42 s
●	☀	TST_FWK	14 h (±17)	N/A	9,9 s

**Liste des jobs**

**S'abonner à un flux RSS**

Légende: tous les builds tous les échecs pour les derniers builds seulement

Page generated: 12 janv. 2010 14:52:53

Figure 35 : Aperçu du lancement manuel d'une tâche dans l'outil Hudson

## Chapitre 2 – Développement de la solution

Afin d'éviter l'effet tunnel visible dans beaucoup de projets informatiques nous avons préféré opter pour un lotissement des développements. Le premier lot sera limité à la conception, au maquettage des écrans, à l'architecture de la solution ainsi qu'au développement des moteurs de collecte et de traitement des événements. Le deuxième lot quant à lui concernera le développement des services, des écrans de présentation de l'application ainsi que la recette et les tests de l'ensemble.

Devant la complexité technique de la solution, il nous a semblé préférable de réaliser ce lotissement à la place d'un découpage fonctionnel plus commun dans les projets informatiques afin de positionner le ou les acteurs les plus pertinents sur chacune des parties.

Etant amené à intervenir principalement sur le lot 1 planifié entre Septembre 2009 et Septembre 2010, cette étude se limitera à la présentation des travaux réalisés dans le cadre de celui-ci.

### Sous-chapitre 1 – Processus de monitoring

Le processus de monitoring est décomposé d'un point de vue logique en trois étapes distinctes :

- ▶ Phase d'arrivée des événements et polling de récupération : le moteur de collecte des événements va recueillir les événements transmis par les composants du SI mais aussi récupérer lui-même à fréquence régulière des informations contenues dans des fichiers de journalisation, des bases de données, et ou en écoutant les trames réseaux.
- ▶ Consolidation des événements et remontée des alertes : à ce niveau un moteur de corrélation permet le calcul des statistiques ainsi que le contrôle des règles prédéfinies (temps de traitement, taux d'erreurs, volumétrie...). Si une règle n'est pas respectée la solution permet l'émission d'une alerte via différents canaux (mail, flux RSS, SMS...).
- ▶ Visualisation des événements, des statistiques, du référentiel et des alertes : une interface Web avancée permet la visualisation des événements, statistiques et composants du système d'information en fonction du rôle de l'utilisateur (administration, étude, exploitation, gouvernance, métier).

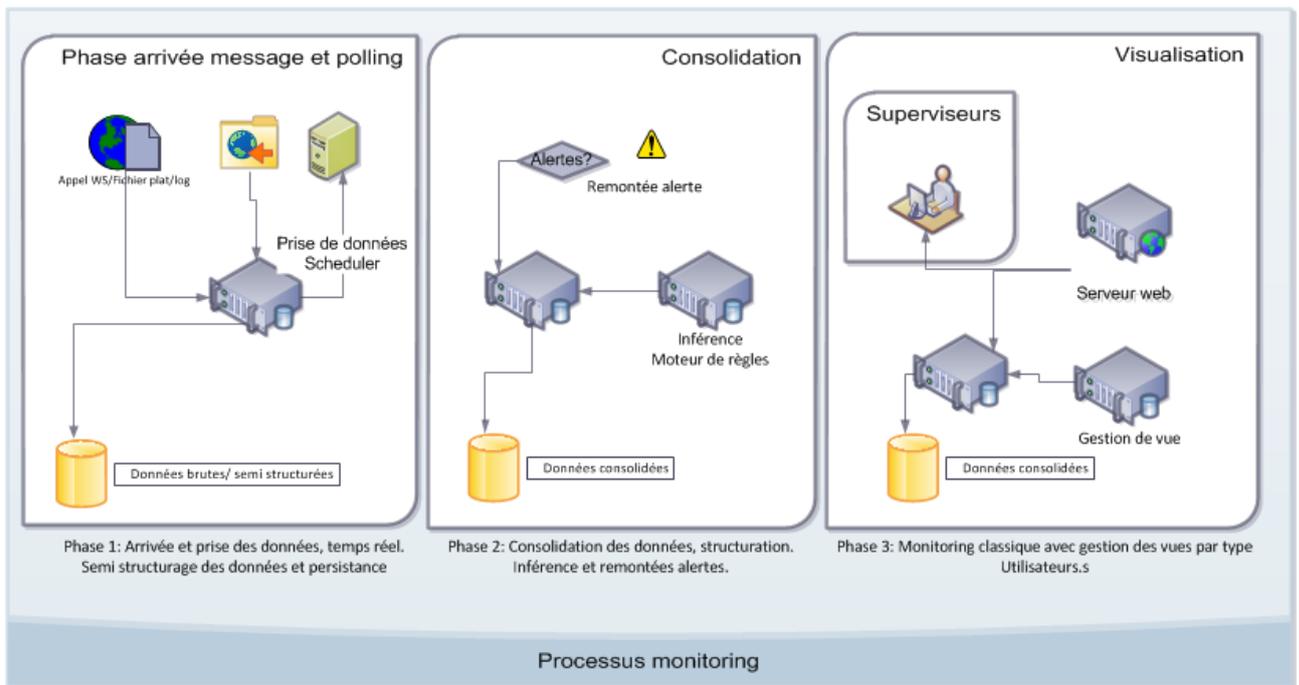


Figure 36 : Illustration du processus de monitoring

## Sous-chapitre 2 – Structure de la base de données

La base de données est composée de quatre domaines :

- ▶ **Événement** : Collecte l'ensemble des événements survenus sur le système (début d'un traitement, fin ou erreur)
- ▶ **Référentiel logique** : Recense l'ensemble des processus, services, données, interfaces et applications composant le SI de l'entreprise ainsi que le lien logique de collaboration.
- ▶ **Référentiel technique** : Recense l'ensemble des serveurs physiques et logiques de l'entreprise ainsi que les instances de processus... déployées sur ces serveurs.
- ▶ **Statistique** : Grâce au moteur de corrélation, l'ensemble des événements traités de manière unitaire permet de déduire des statistiques attribuées aux composants contenus dans le référentiel logique et le référentiel technique.

Le modèle physique des données ci-dessous est présenté à titre d'illustration, il reprend le découpage en 4 domaines distincts présenté précédemment ainsi qu'une cinquième zone pour la gestion des paramètres de l'application (utilisateurs, profils, alertes, journaux d'erreurs...).

### Stockage des événements

L'ensemble des données communes à tous les types d'évènements (date et heure de déclenchement, instance logique, serveur physique...) a été regroupé dans une table de spécialisation nommée EVENT afin de permettre le polymorphisme au niveau des fonctions Java et ainsi gérer de la même manière l'ensemble des évènements et ce, en

intégrant leur spécificité au niveau le plus bas dans les fonctions par l'intermédiaire d'un test de typage en Java réalisé grâce au mot clé *instanceof*.

La table EVENT\_TYPE permet de faire le lien entre la structure Oracle où la table EVENT contiendra une clé étrangère vers la table EVENT\_TYPE et la classe Java où le type de l'évènement sera contenu dans un Enum correspondant aux clés primaires de la table EVENT\_TYPE (cf. extrait de code ci-dessous).

```

/**
 * Event Type Status Enum.
 */
public enum EventTypeStatus {
    START(1), PENDING(2), ERROR(3), STOP(4);

    protected int mConstant;

    /** Default constructor */
    EventTypeStatus(int pConstant) {
        mConstant = pConstant;
    }

    public int getConstant() {
        return mConstant;
    }
}

```

La structure Enum permet d'associer une constante à chaque type d'évènement

Figure 37 : Code source de la structure Enum permettant de définir le type d'évènement

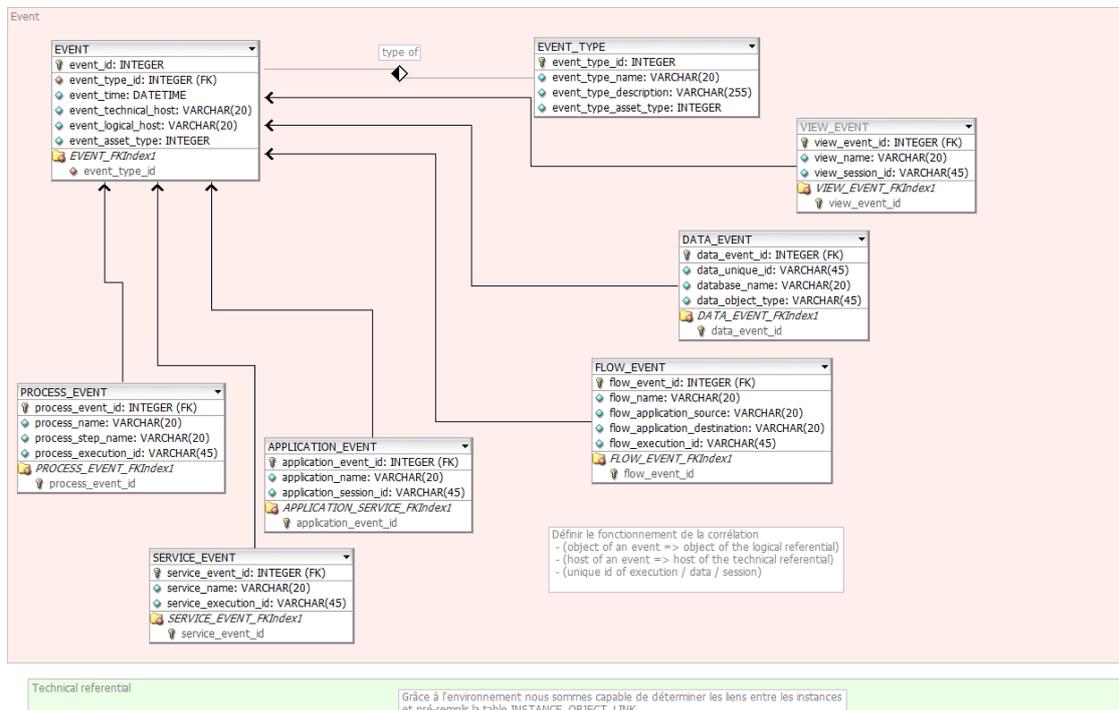


Figure 38 : Modèle de données de stockage des évènements

### Stockage du référentiel logique

A l'instar des tables de stockage des évènements, l'ensemble des données communes à tous les objets du référentiel logique (nom, description, type...) a été regroupé dans une table de spécialisation nommée ASSET. Les objets du référentiel logique sont alors regroupés par domaine fonctionnel (par exemple Gestion de contrat, Gestion de sinistre...) grâce à l'association entre les tables ASSET et FUNCTIONAL\_DOMAIN et ceci afin de faciliter la navigation et la recherche des objets dans l'application.

Une table ASSET\_LINK permet de définir des liens entre les objets du référentiel logique. Un processus de validation de commande fait par exemple appel à un service métier de paiement, il est nécessaire d'enregistrer le lien entre ces deux objets du référentiel afin de savoir que toute modification sur le service de paiement pourrait avoir un impact sur le processus de validation de commande mais surtout que tout arrêt en production de ce service entrainera un arrêt du processus appelant.

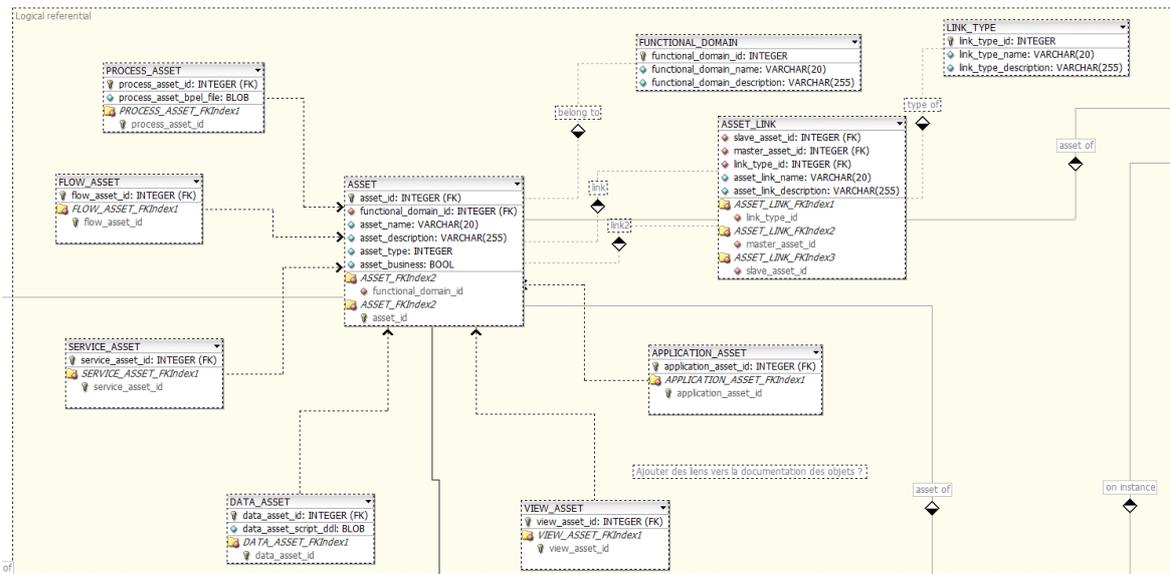


Figure 39 : Modèle de données de stockage du référentiel logique

### Stockage du référentiel technique

L'ensemble des données communes aux instances d'objets du référentiel logique a été regroupé dans une table de spécialisation nommée ASSET\_INSTANCE. Les instances d'objets du référentiel logique sont alors regroupées par environnement (par exemple Production, Pré-production, Recette...) grâce à l'association entre les tables ASSET\_INSTANCE et ENVIRONMENT et ceci afin de faciliter la navigation et la recherche d'instance d'objets dans l'application.

Une table ASSET\_INSTANCE\_LINK permet de définir des liens entre les instances d'objets du référentiel logique lors de la mise en place de mécanisme de tolérance aux pannes. Une instance est alors destinée à remplacer l'instance principale lors de toute défaillance de celle-ci. De plus, la table d'association ASSET\_INSTANCE\_LOCATION permet de stocker l'URL ou chemin d'accès à l'objet en question en fonction du protocole de communication disponible (http, JMS, JDBC...).

La table DEVICE permet la spécialisation entre serveur logique, serveur physique et matériel réseau. Un matériel (DEVICE) peut alors être associé à une instance du référentiel physique. Comme on le verra plus tard, c'est cette association qui permet au moteur de corrélation des événements de faire le lien entre un événement s'étant produit sur un serveur physique et une instance logique.

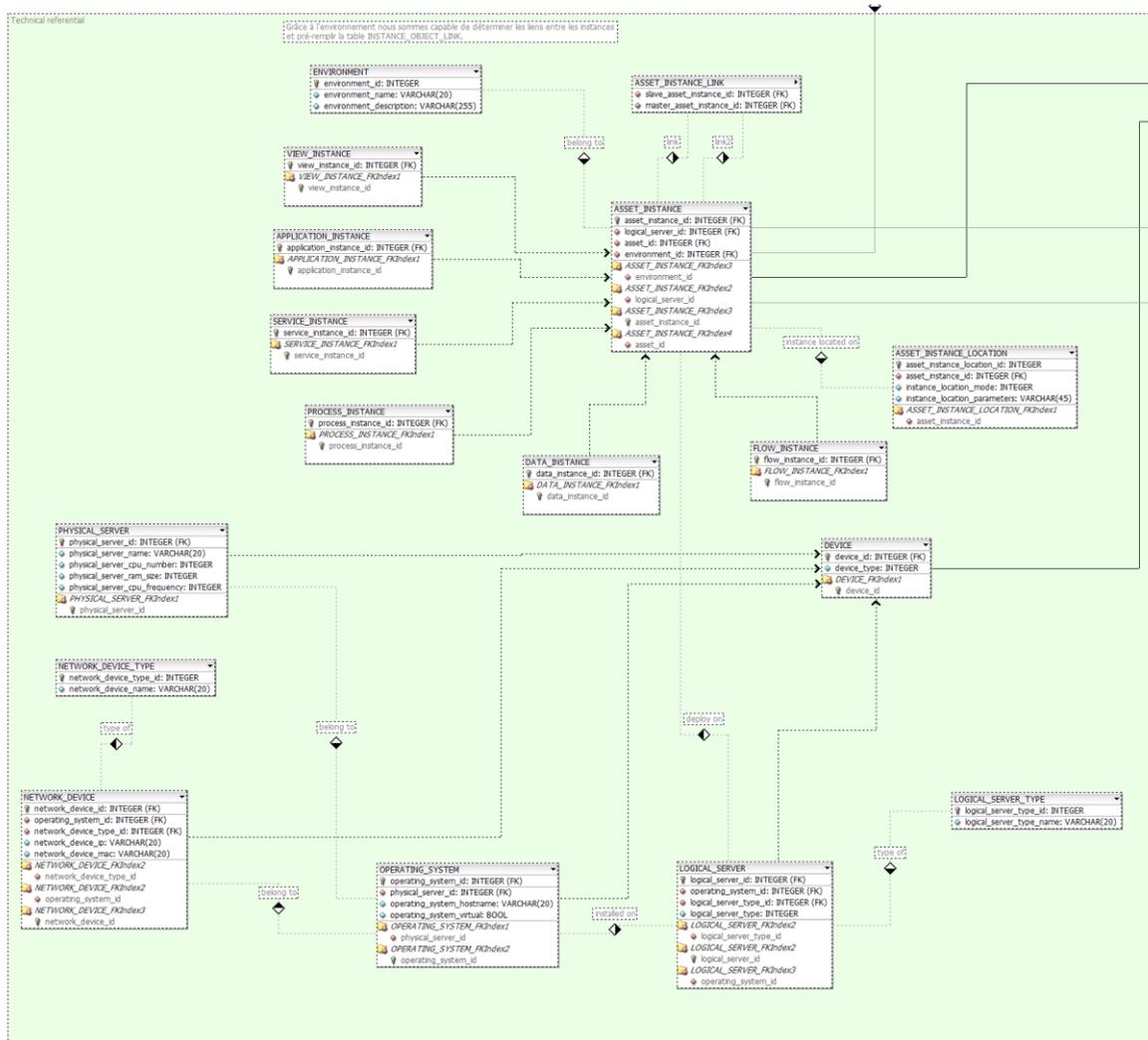


Figure 40 : Modèle de données de stockage du référentiel technique

### Stockage des statistiques

L'ensemble des données communes aux statistiques sur une instance d'un objet du référentiel logique a été regroupé dans une table de spécialisation nommée ASSET\_STATISTIC. Cette table ainsi que ses tables filles (DATA\_STATISTIC, SERVICE\_STATISTIC...) permettent de stocker des statistiques agrégées sur une instance d'un objet du référentiel :

- ▶ Nombre d'alertes retournées pour cette instance
- ▶ Nombre d'appels
- ▶ Durée de l'appel le plus court
- ▶ Durée de l'appel le plus long
- ▶ Durée de l'appel moyen
- ▶ Etc...

Quant à la table ASSET\_EXECUTION\_STATISTIC, qui est la spécialisation des statistiques sur une exécution d'une instance d'un service, elle permet avec ses tables filles

(DATA\_EXECUTION\_STAT, SERVICE\_EXECUTION\_STAT...) d'agrèger les statistiques issues des différents évènements survenant lors d'une exécution d'une instance d'un objet du référentiel logique.

Il s'agit par exemple :

- ▶ Heure de début de l'exécution
- ▶ Heure de fin de l'exécution
- ▶ Statut de l'exécution
- ▶ Etc...

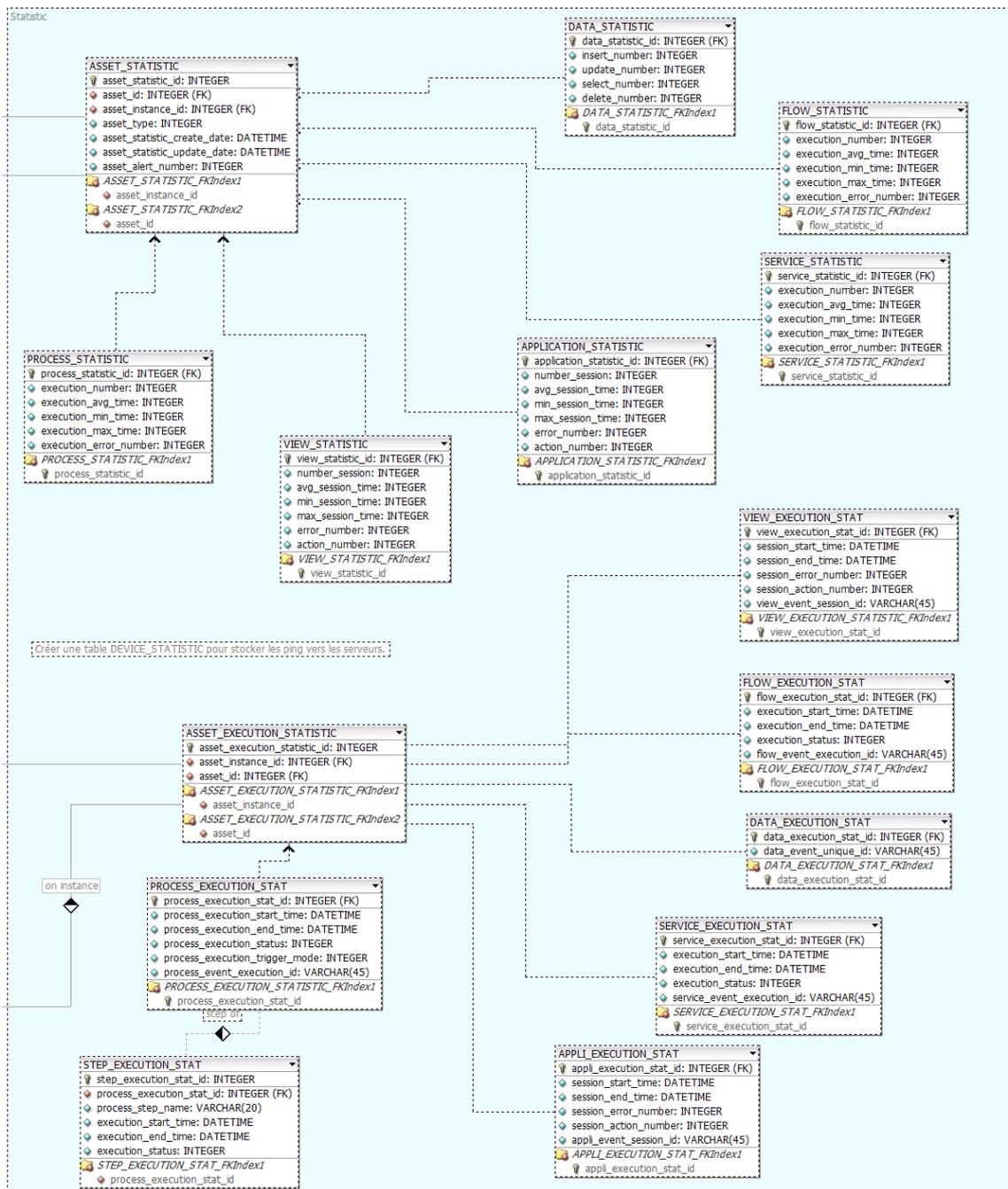


Figure 41 : Modèle de données de stockage des statistiques

## Stockage des paramètres

Les tables de stockage du paramétrage permettent de faire le lien entre un utilisateur stocké dans la table APPLICATION\_USER et ses profils stockés dans la table USER\_PROFIL par l'intermédiaire de la table d'association HAVE\_PROFIL. Un profil donne alors le droit d'accéder à certains écrans de l'application, les écrans de l'application étant stockés dans la table APPLICATION\_SCREEN.

Pour gérer de manière unifiée les droits d'accès à l'ensemble des ressources de l'application nous avons créé une table de spécialisation APPLICATION\_RESOURCE qui regroupe les ressources du type :

- ▶ Ecrans de l'application (APPLICATION\_SCREEN)
- ▶ Fonctionnalités de l'application (APPLICATION\_FEATURE)
- ▶ Paramètres de l'application (APPLICATION\_PARAMETER)
- ▶ Tables de paramétrage de l'application (APPLI\_PARAMETER\_TABLE)
- ▶ Tâches paramétrées (TASK)
- ▶ Listes de destinataires de messages (MESSAGE\_RECEIVER)
- ▶ Alertes (ALERT)

Des alertes ou tâches peuvent être programmées par l'utilisateur au travers de l'application. Leur exécution est alors stockée en base de données, ainsi que la liste des destinataires qui doivent être notifiés en cas d'exécution d'une tâche ou d'une alerte remontée par le système. Ce fonctionnement est détaillé dans le chapitre concernant le moteur de gestion des tâches et alertes.

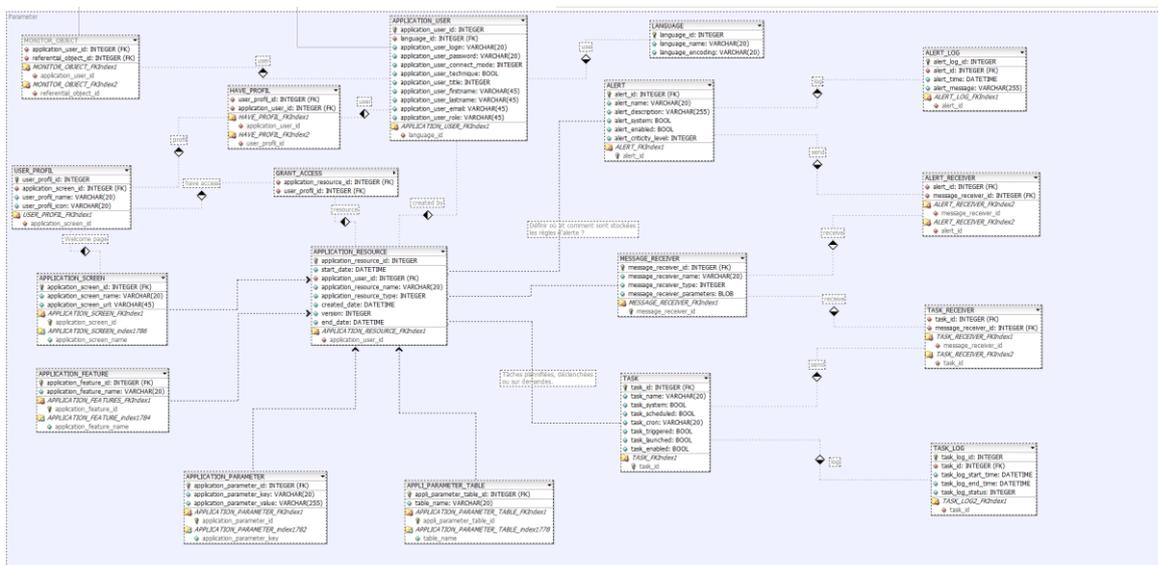


Figure 42 : Modèle de données de stockage des paramètres

## Généralisation des objets du référentiel

Nous avons procédé à une généralisation des entités du référentiel logique et technique en un objet parent nommé REFERENTIAL\_OBJECT ayant pour but de porter les notions de droits, date de création, date de validité et version communes à ces différents objets.

Cette généralisation faisant appel à trois domaines (référentiel logique, référentiel technique et paramètres) est définie en dehors de ceux-ci comme on peut le voir sur le schéma du modèle ci-dessous :

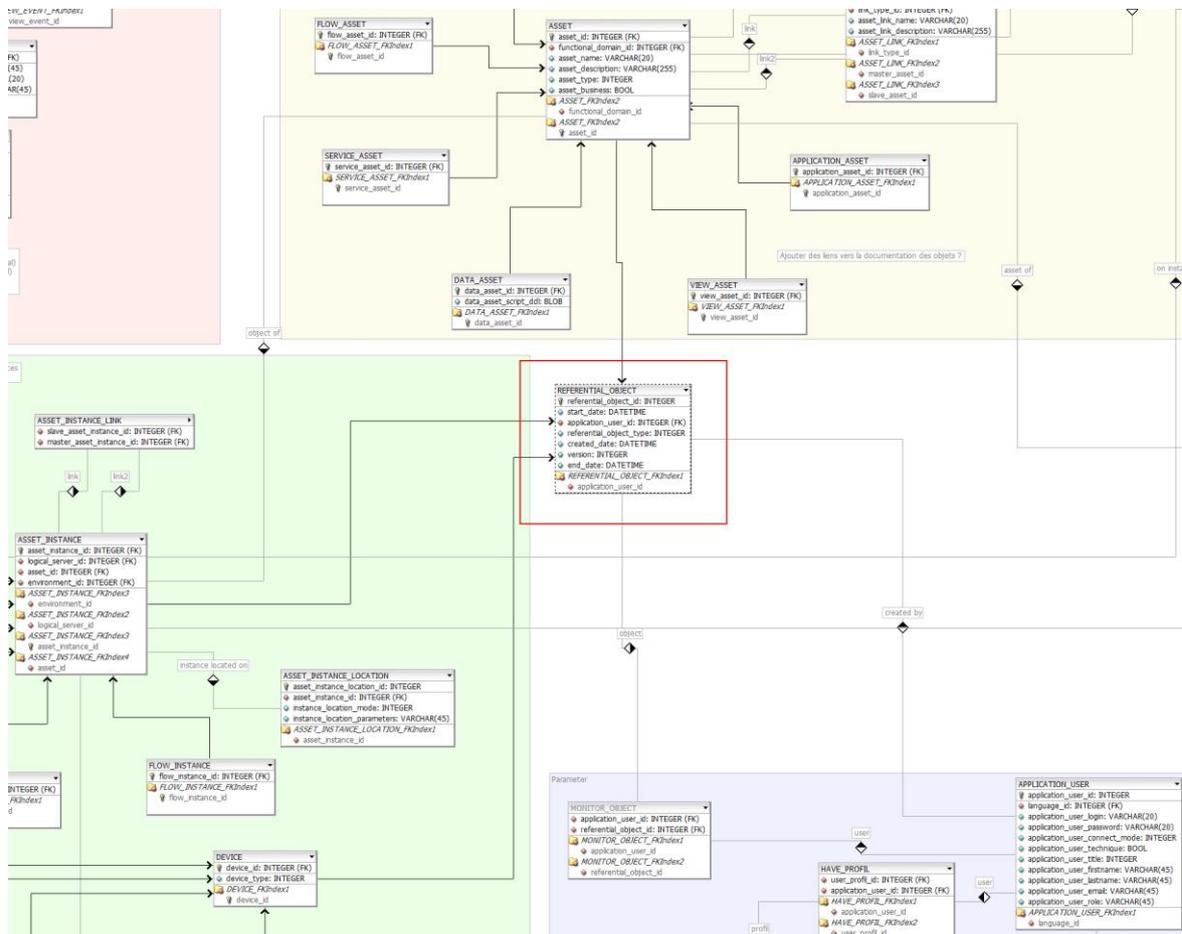


Figure 43 : Illustration de la généralisation des objets du référentiel

## Génération des ordres DDL et des entity Java

L'ensemble du modèle de données a été modélisé grâce à l'utilisation de l'application Open Source DBDesigner 4. Cet outil permet un export du modèle de données sous la forme d'un script SQL de création de table comme illustré ci-dessous.

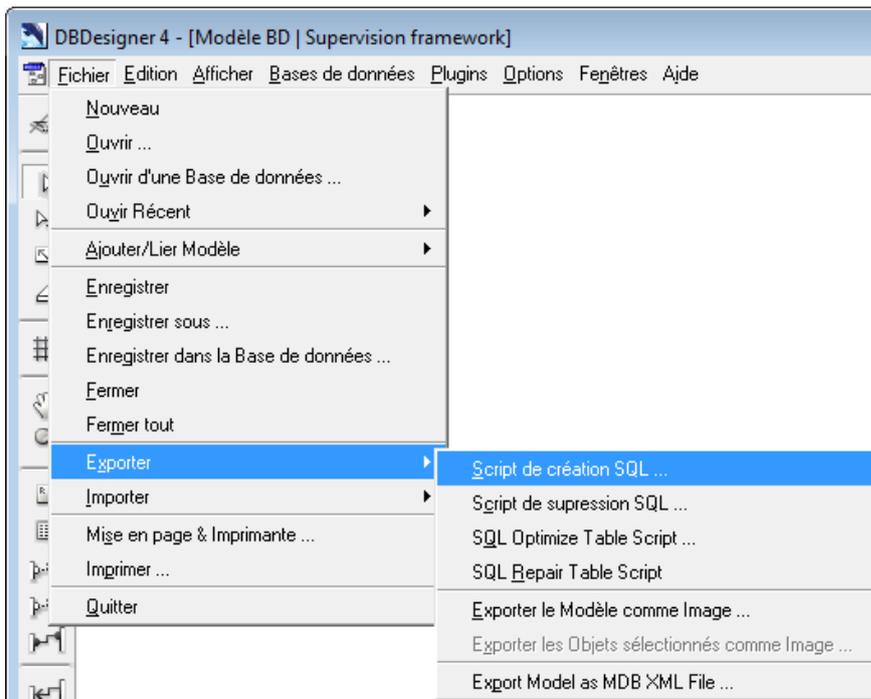


Figure 44 : Fonctionnalité d'export de l'outil DBDesigner 4

Le Script SQL alors généré est compatible avec la norme SQL-92, mais ne peut pas être exécuté directement sur une base Oracle en raison des types de données utilisés par les bases Oracle (VARCHAR2, NUMBER...). De plus, les clés étrangères ne sont pas correctement générées par l'outil.

Afin de permettre un passage rapide entre le stade de modélisation et la création des tables Oracle correspondantes, nous avons créé un programme Java nommé *ISM.Generator.SQL* permettant le passage de manière quasi-automatique d'un Script SQL-92 à un Script de création de tables Oracle comme détaillé ci-dessous :

```

// Ouverture des fichiers de sortie
lOutputFile = new File("Output/Scripts création tables Oracle.sql");
lOutputFileDelete = new File("Output/Scripts suppression tables Oracle.sql");

// Ecriture des fichiers de sortie
lFileWriter = new FileWriter(lOutputFile);
lFileWriterDelete = new FileWriter(lOutputFileDelete);

while ((lInputFileLine = lBufferedReader.readLine()) != null) {
    lTypeField = false;

    // Selon le type de ligne
    if (lInputFileLine.indexOf("CREATE TABLE") != -1) {
        lOutputFileLine = lInputFileLine;

        // Récupération du nom de la table
        lTableName = lInputFileLine.substring(13, lInputFileLine.length() - 2);

        if (lTableName.length() > 30) {
            throw new Exception("ERROR - An identifier with more than 30 characters was specified: [" + lTableName + "];");
        }

        lPrimaryKey = null;
        lForeignKey = null;
        lUniqueKey = null;
        lPrimaryKeyNumber = 1;
        lForeignKeyNumber = 1;
        lIndexNumber = 1;

        // Ecriture d'une ligne dans le fichier de suppression
        lFileWriterDelete.append("DROP TABLE " + lTableName + ";\n");

        System.out.println("[Info] - Debut du traitement de la table [" + lTableName + "].");
    } else if (lInputFileLine.indexOf("INTEGER") != -1) {
        lOutputFileLine = lInputFileLine.replaceAll("INTEGER", "NUMBER(10,0)").replaceAll("UNSIGNED ", "").replaceAll(" AUTO_INCREMENT", "");
        lTypeField = true;
    } else if (lInputFileLine.indexOf("DATETIME") != -1) {
        lOutputFileLine = lInputFileLine.replaceAll("DATETIME", "NUMBER(10,0)");
    }
}

```

Les clés étrangères sont définies dans un fichier de configuration

Test du nom de table à moins de 30 caractères

Transformation de chaque type SQL-92 en type Oracle

Figure 45 : Extrait du programme de génération du Script de création des tables Oracle

Nous définissons dans le programme les clés étrangères de chaque table afin de permettre l'ajout des ordres de création des contraintes correspondantes. Ce programme permet aussi de tester que le nom de chaque objet Oracle créé (table, contrainte, index...) ne dépasse pas 30 caractères. En effet, le nom d'un objet déclaré dans une base Oracle ne doit pas dépasser 30 caractères.

Nous sommes alors en mesure de créer les tables Oracle permettant de stocker nos données.

Le MCD ayant été réalisé avec une approche Objet, les entités manipulées dans le code Java ont une correspondance proche avec les tables. Nous avons donc décidé d'écrire un générateur de code nommé *ISM.Generator.Entity* permettant le passage entre les tables Oracle et les entités Java.

Ce générateur de code se base sur la table Oracle *USER\_TAB\_COLUMNS* qui contient la définition précise de l'ensemble des colonnes constituant les différentes tables (nom, type, longueur, pouvant être nulle...).

Pour chaque table nous créons alors une classe Java comportant l'ensemble des attributs ainsi que les accesseurs et modificateurs (méthode Get et Set) en distinguant les clés primaires des autres types d'attributs comme illustré ci-dessous :

```

if ((!lForeignKeyList.containsKey(lTableName + "." + lColumn.getColumn().toLowerCase())) && (!lColumn.getColumn().toLowerCase().equals(lTableName.
lForeignKey = (String[]) lForeignKeyList.get(lTableName + "." + lColumn.getColumn().toLowerCase());
lColumnName = lAttributeName.substring(0, lAttributeName.length() - 2);
lForeignKeyObject = GenerateEntity.getObject(lForeignKey[0]);

lContent.append(" /**\n * @param p" + lColumnName + " the m" + lColumnName + " to set.\n */\n");
lContent.append(" public void set" + lColumnName + "(" + lForeignKeyObject + " p" + lColumnName + ") {\n");
lContent.append("     m" + lColumnName + " = p" + lColumnName + ";\n");

// Récupération des paramètres pour la classe en foreign key
if (!lObjectList.containsKey(lForeignKeyObject)) {
    throw new Exception("ERREUR - Paramètres pour la classe [" + lForeignKeyObject + "] absents.");
}
lForeignKeyObjectParameter = (String[]) lObjectList.get(lForeignKeyObject);
lObjectImport.put(lForeignKeyObjectParameter[0] + "." + lForeignKeyObject, lForeignKeyObjectParameter[0] + "." + lForeignKeyObject);
} else {
lContent.append(" /**\n * @param p" + lAttributeName + " the m" + lAttributeName + " to set.\n */\n");
if ("VARCHAR2".equals(lColumn.getDataType())) {
lContent.append(" public void set" + lAttributeName + "(" + lAttributeName + ") {\n");
} else if ("NUMBER".equals(lColumn.getDataType())) {
    if (lColumn.getDataLength() == 1 && lColumn.getDataScale() == 0) {
        lContent.append(" public void set" + lAttributeName + "(Boolean p" + lAttributeName + ") {\n");
    } else if (lColumn.getDataScale() == 0) {
        lContent.append(" public void set" + lAttributeName + "(Long p" + lAttributeName + ") {\n");
    } else {
        lContent.append(" public void set" + lAttributeName + "(BigDecimal p" + lAttributeName + ") {\n");
    }
} else if ("DATE".equals(lColumn.getDataType())) {
lContent.append(" public void set" + lAttributeName + "(Date p" + lAttributeName + ") {\n");
} else if ("BLOB".equals(lColumn.getDataType())) {
lContent.append(" public void set" + lAttributeName + "(Byte[] p" + lAttributeName + ") {\n");
} else {
    throw new Exception("Unsupported data type.");
}
lContent.append("     m" + lAttributeName + " = p" + lAttributeName + ";\n");
}
lContent.append(" }\n\n");

```

On distingue les clés primaires

Selon le type Oracle on applique un type Java

Envoi d'une exception dans le cas d'un type non supporté

Figure 46 : Extrait du programme de génération des entités Java

L'ensemble des classes Java générées est alors stocké dans un projet Java à part nommé *ISM.Java.Entity* destiné à être inclus sous la forme d'un JAR<sup>12</sup> dans l'ensemble des projets Java de l'application manipulant ces entités.

En dernier lieu, afin de faciliter le travail des développeurs qui devront utiliser ces objets nous avons créé un programme Java nommé *ISM.Generator.Manager* permettant de générer les classes Java contenant les ordres SQL élémentaires applicables à chaque objet métier :

- ▶ Création de l'objet (ordre SQL INSERT)
- ▶ Mise à jour de l'objet (ordre SQL UPDATE)
- ▶ Récupération de l'objet (ordre SQL SELECT)
- ▶ Suppression de l'objet (ordre SQL DELETE)
- ▶ Recherche d'une occurrence (ordre SQL SELECT)
- ▶ Liste des objets (ordre SQL SELECT)

Ce programme fonctionne sur le même principe que le générateur des entités métiers en ouvrant un fichier sur le disque pour chaque classe Java et en y inscrivant l'ensemble des fonctions présentées ci-dessus selon les attributs propres à chaque table.

<sup>12</sup> En informatique, un fichier JAR (Java ARchive) est un fichier ZIP utilisé pour distribuer un ensemble de classes Java. Ce format est utilisé pour stocker les définitions des classes, ainsi que des métadonnées, constituant l'ensemble d'un programme.

## Sous-chapitre 3 – Collecte et traitements des évènements

### Sondes complexes

Une sonde complexe est un programme s'exécutant sur l'ordinateur où est déployé l'objet que l'on souhaite superviser. Cette sonde a pour objectif de capturer tous les événements qui peuvent se produire au niveau de l'objet supervisé (instanciation de l'objet, anomalie...).

Afin de permettre la remontée de ces informations au serveur central nous avons opté pour la mise en œuvre d'un réseau d'agents.

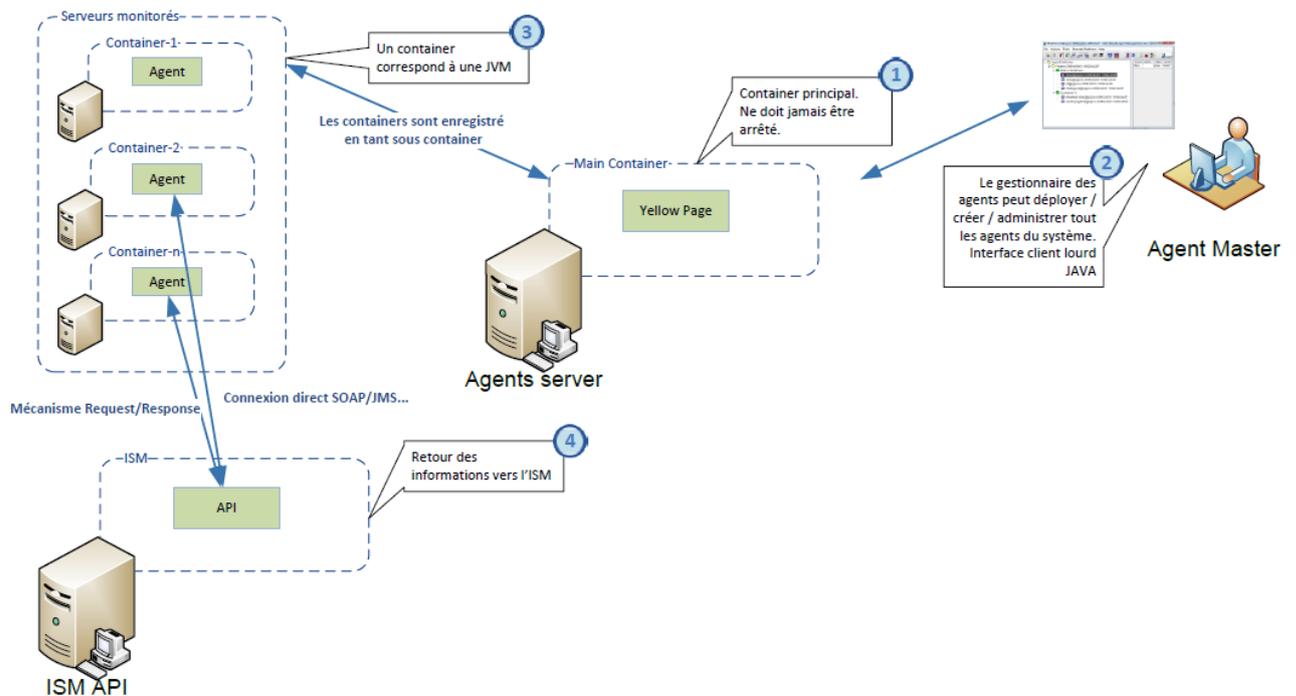


Figure 47 : Principe de fonctionnement d'un réseau d'agents

Les agents déployés sur les serveurs à superviser sont démarrés dans des containers (un container correspond à une JVM Java). Chaque container est relié au container principal (*Main container*) qui lui-même dispose de la liste des containers sous sa responsabilité.

Ainsi il est possible depuis l'agent d'envoyer des informations à l'ISM tout en étant administré par une console centrale (technologie client lourd Java) connectée au conteneur principal.

Cette solution permet ainsi de piloter à distance le réseau d'agents déployé sur un ensemble de serveurs.

### Interfaces de communication

Lorsqu'il n'est pas possible de superviser un objet en utilisant une sonde complexe et le réseau d'agent ou bien par la récupération d'informations contenues dans des fichiers de journalisation, nous mettons en place des interfaces de communication avec l'ISM.

Ces interfaces permettent notamment :

- ▶ L'enregistrement d'un événement pour un objet
- ▶ L'analyse d'un fichier de journalisation
- ▶ La récupération d'une statistique
- ▶ Etc...

Pour se faire nous avons décidé d'implémenter ces fonctionnalités sous la forme de *Web Services* (Services Web).

Toute la logique des *Web Services* repose sur l'utilisation de standard, en l'occurrence l'XML (*eXtensible Markup Language*) qui est un langage de description et d'échange des documents structurés. Cette utilisation de standard permet une indépendance vis-à-vis de la plate-forme et une compatibilité inter-entreprises.

Les *Web Services* permettent alors un accès distant aux traitements et aux données d'une application sans se préoccuper de la couche « Présentation » qui sera alors déléguée aux programmes qui consomment ces services.

### **Moteur de collecte des évènements**

Les événements de type exécutions sont envoyés directement par les instances des composants logiciels concernés, en fonction du mode d'interaction souhaité (utilisation de l'API d'interface avec l'application de monitoring, utilisation d'un analyseur de log, sniffer de trafic http...).

Le nombre d'événements d'exécution pouvant être important, il est préférable d'isoler cette partie autant sur le plan logique que physique (application de collecte des événements dédiée et serveur physique dédié).

Afin de permettre à la solution de tenir la charge lorsque le nombre d'événements reçus par le système augmente rapidement, nous avons décidé de mettre en place un moteur de collecte des événements destiné à servir de tampon entre la réception et le traitement des événements.

Comme détaillé dans le schéma ci-dessous, le moteur de collecte des événements est basé sur l'utilisation d'une file JMS de type *Queue* pour chaque type d'événement. Le bus de messagerie (ici Active MQ) va alors conserver en mémoire les événements reçus sur chaque *Queue* JMS.

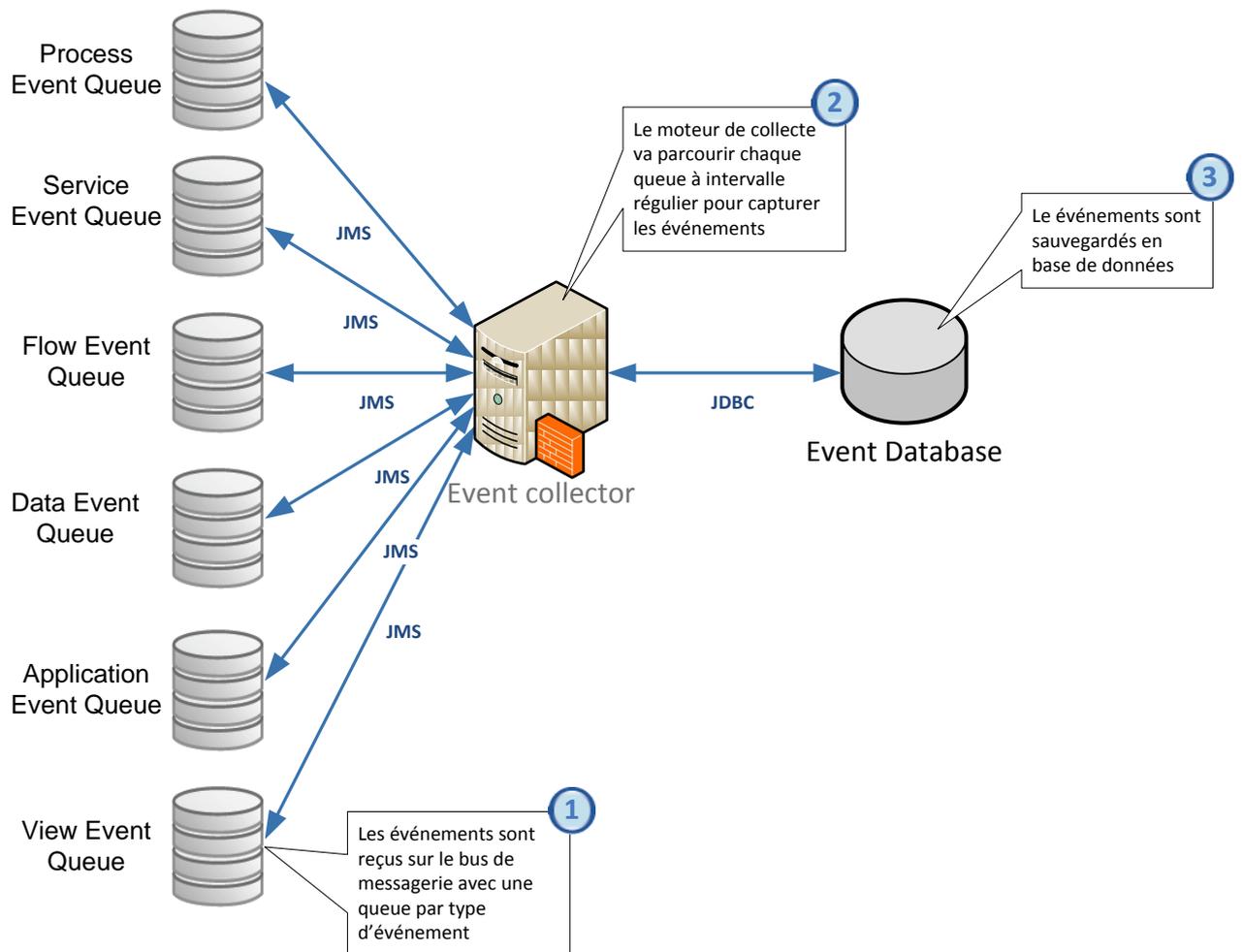


Figure 48 : Architecture du moteur de collecte des événements

Une application Java fonctionnant avec une simple JVM en mode lancement de type Main va ouvrir un *listener* pour chacun des types de composant supervisé par l'ISM comme détaillé ci-dessous.

```

try {
    // Load parameters
    Parameters.init("conf/EventCollector.properties");

    // Start process event listener
    lProcessEventListener = new EventListener(Parameters.JMS_MONITORING_PROCESS_QUEUE, Parameters.JMS_MONITORING_PROCESS_THREAD_SLEEP, Parameters..
    lProcessEventListener.start();

    // Start service event listener
    lServiceEventListener = new EventListener(Parameters.JMS_MONITORING_SERVICE_QUEUE, Parameters.JMS_MONITORING_SERVICE_THREAD_SLEEP, Parameters..
    lServiceEventListener.start();

    // Start flow event listener
    lFlowEventListener = new EventListener(Parameters.JMS_MONITORING_FLOW_QUEUE, Parameters.JMS_MONITORING_FLOW_THREAD_SLEEP, Parameters.JMS_MONIT
    lFlowEventListener.start();

    // Start application event listener
    lApplicationEventListener = new EventListener(Parameters.JMS_MONITORING_APPLICATION_QUEUE, Parameters.JMS_MONITORING_APPLICATION_THREAD_SLEEP,
    lApplicationEventListener.start();

    // Start data event listener
    lDataEventListener = new EventListener(Parameters.JMS_MONITORING_DATA_QUEUE, Parameters.JMS_MONITORING_DATA_THREAD_SLEEP, Parameters.JMS_MONIT
    lDataEventListener.start();

    // Start view event listener
    lViewEventListener = new EventListener(Parameters.JMS_MONITORING_VIEW_QUEUE, Parameters.JMS_MONITORING_VIEW_THREAD_SLEEP, Parameters.JMS_MONIT
    lViewEventListener.start();

    do {
        // Temporisation
        Thread.sleep(40L);
    } while(true);
} catch (InterruptedException lInterruptedException) {
    mLogger.error(lInterruptedException.getMessage());
} catch (Exception lException) {
    mLogger.error(lException.getMessage());
} finally {
    // Stopping all listener
    if (lProcessEventListener != null) {
        lProcessEventListener.stop();
    }
}

```

Le paramétrage de la base de données ainsi que du bus JMS est stocké dans un fichier properties

Pour chaque type de composant un listener est démarré

En cas d'arrêt du programme principal les listeners sont arrêtés

Figure 49 : Extrait de la boucle principale du programme de collecte des événements

Un *listener* est composé d'un *thread* Java qui va écouter chacune des queues JMS pour enregistrer l'ensemble des messages en base de données (cf. extrait de code ci-dessous).

Les connexions aux *queues* JMS ainsi que les connexions à la base de données restant actives, il est possible, grâce à ce procédé, d'obtenir des performances optimales dans le traitement des événements tout en offrant une zone tampon aux programmes extérieurs par l'intermédiaire de *queues* JMS, capables de recevoir un nombre de messages simultanés très important (plusieurs milliers pas seconde).

```

/**
 * Start the listener or confirm if it is already started.
 */
public void start() {
    if (mThread == null || !mThread.isAlive()) {
        mLogger.debug("BEGIN: Start listener service for " + mJmsListenedQueue + ".");

        try {
            // Initialize database for current thread
            mDBConnector = new DBConnector();
            mDBConnector.open(Parameters.EVENT_DATABASE_DRIVER_CLASS, Parameters.EVENT_DATABASE_DRIVER_ACCESS, Parameters.EVENT_DATA

            // Initialize the event manager
            mEventManager = new EventManager(mDBConnector);
            mEventManager.startEventManager();

            // Connection to the MQ broker
            mJmsConnector = new JmsConnector();
            mJmsConnector.createConnection(Parameters.JMS_SERVER_URL, Parameters.JMS_SERVER_LOGIN, Parameters.JMS_SERVER_PASSWORD);

            // Open one JMS message consumer for each type of services
            mJmsConnector.createMessageConsumer(mJmsListenedQueue, Constants.QUEUE);

            // Update listener status
            mListenerActivated = true;

            // Start running the event listener
            mThread = new Thread(this);
            mThread.start();

            mLogger.debug("END: Start listener service for " + mJmsListenedQueue + ".");
        } catch (SQLException lSQLException) {
            mLogger.error(lSQLException.getMessage());
        }
    }
}

```

Connexion à la base de données et au bus JMS

Figure 50 : Extrait de la méthode *start* du listener d'évènements

Une fois démarrée, la fonction *run* va lire un nombre limité de messages dans la file JMS pour ne pas utiliser toutes les ressources du bus JMS puis se remettre en attente jusqu'à la prochaine itération (cf. extrait de code ci-dessous). Le nombre de messages récupérés par itération ainsi que le délai d'attente entre chaque itération sont stockés dans un fichier *properties* afin de permettre le réglage de ces paramètres en fonction de la capacité de traitement de l'environnement de production de l'ISM.

```

/**
 * Run the event listener.
 */
public void run() {
    mLogger.debug("BEGIN: Running service for '" + mJmsListenedQueue + "'.");

    // Variables declaration and initialization
    Event lEvent = null;
    TextMessage lTextMessage = null;
    List<Event> lEventList = null;

    do {
        try {
            // Initialize event list
            lEventList = new ArrayList<Event>(0);

            // Get messages from the monitoring process queue
            while((lTextMessage = (TextMessage) mJmsConnector.getMessageConsumer(mJmsListenedQueue).receiveNoWait()) != null) {
                // Use XStream to deserialize the xml stream
                lEvent = (Event) XmlParser.convertXmlToObject(lTextMessage.getText());

                // Add this event to the list
                lEventList.add(lEvent);

                // Test if we have completed our list of event
                if (lEventList.size() == mNumberRows) {
                    break;
                }
            }

            // Insert messages into database
            mEventManager.saveEventList(lEventList);

            if (lEventList.size() > 0) {
                mLogger.debug(lEventList.size() + " messages saved from '" + mJmsListenedQueue);
            }

            // Temporisation
            Thread.sleep(mThreadSleep);
        } catch (InterruptedException interruptedException) {

```

Les messages JMS sont transformés en objets Java et passés dans une liste

La liste d'objets est sauvegardée en base de données

Figure 51 : Extrait de la méthode *run* du listener d'évènements

Le *manager* chargé de sérialiser les objets de type Evènement en base de données dispose d'une fonction *Start* appelée lors du démarrage du listener. Cette fonction permet de calculer à l'avance les requêtes d'insertion grâce à l'utilisation d'un *PreparedStatement* (cf. extrait de code ci-dessous).

```

/**
 * Start event manager by initializing prepared statement.
 */
public void startEventManager() throws SQLException {
    // Variables declaration and initialization
    String lEventQuery = null;
    String lProcessEventQuery = null;
    String lServiceEventQuery = null;
    String lDataEventQuery = null;
    String lViewEventQuery = null;
    String lFlowEventQuery = null;
    String lApplicationEventQuery = null;
    String lEventSeqQuery = null;

    // Save event query
    lEventSeqQuery = "select EVENT_SEQ.nextval from dual";
    lEventQuery = "insert into EVENT (EVENT_ID, EVENT_TYPE_ID, EVENT_TIME, EVENT_TECHNICAL_HOST, EVENT_LOGICAL_HOST, EVENT_ASSET_TYPE) VALUES (
    lProcessEventQuery = "insert into PROCESS_EVENT (PROCESS_EVENT_ID, PROCESS_NAME, PROCESS_STEP_NAME, PROCESS_EXECUTION_ID, APPLICATION_SESSI
    lServiceEventQuery = "insert into SERVICE_EVENT (SERVICE_EVENT_ID, SERVICE_NAME, SERVICE_EXECUTION_ID, APPLICATION_SESSION_ID, VIEW_SESSION
    lDataEventQuery = "insert into DATA_EVENT (DATA_EVENT_ID, DATA_UNIQUE_ID, DATABASE_NAME, DATA_OBJECT_TYPE) VALUES (?, ?, ?, ?)";
    lApplicationEventQuery = "insert into APPLICATION_EVENT (APPLICATION_EVENT_ID, APPLICATION_NAME, APPLICATION_SESSION_ID, VIEW_SESSION_ID) V
    lViewEventQuery = "insert into VIEW_EVENT (VIEW_EVENT_ID, VIEW_NAME, VIEW_SESSION_ID) VALUES (?, ?, ?)";
    lFlowEventQuery = "insert into FLOW_EVENT (FLOW_EVENT_ID, FLOW_NAME, FLOW_CORRELATION_ID, PROCESS_EXECUTION_ID, SERVICE_EXECUTION_ID) VALUE

    try {
        // Prepared statement initialization
        mEventSeqPreparedStatement = mDBConnector.getPreparedStatement(lEventSeqQuery);
        mEventPreparedStatement = mDBConnector.getPreparedStatement(lEventQuery);
        mProcessEventPreparedStatement = mDBConnector.getPreparedStatement(lProcessEventQuery);
        mServiceEventPreparedStatement = mDBConnector.getPreparedStatement(lServiceEventQuery);
        mDataEventPreparedStatement = mDBConnector.getPreparedStatement(lDataEventQuery);
        mViewEventPreparedStatement = mDBConnector.getPreparedStatement(lViewEventQuery);
        mApplicationEventPreparedStatement = mDBConnector.getPreparedStatement(lApplicationEventQuery);
        mFlowEventPreparedStatement = mDBConnector.getPreparedStatement(lFlowEventQuery);
    } catch (SQLException lSQLException) {
        // Trying to close all opened prepared statement
        stopEventManager();
        throw lSQLException;
    }
}

```

Initialisation d'un prepared statement par type d'objet

Figure 52 : Extrait de la méthode *start* du manager d'insertion des objets en base

Puis en fonction du type d'objet l'opération *saveEventListTransactionalMode* va permettre la sauvegarde de la liste complète d'évènements en base de données en utilisant la requête appropriée à chaque type d'évènements (cf. extrait de code ci-dessous).

```

private SqlResultSet saveEventListTransactionalMode(List<Event> pEventList) throws SQLException, Exception {
    // Variables declaration and initialization
    ResultSet lEventSeqResultSet = null;
    SqlResultSet lSqlResultSet = new SqlResultSet();
    int lCurrentSeqVal = 0;

    try {
        // Loop into the event list
        for(Event lEvent : pEventList){
            // Récupération d'un curseur de séquence
            lEventSeqResultSet = mEventSeqPreparedStatement.executeQuery();
            if (lEventSeqResultSet.next()) {
                lCurrentSeqVal = lEventSeqResultSet.getInt(1);
                lSqlResultSet.getCreatedSequenceList().add(new Integer(lCurrentSeqVal));
            }

            // Alimentation de la requête
            mEventPreparedStatement.setInt(1, lCurrentSeqVal);
            mEventPreparedStatement.setInt(2, lEvent.getEventType().getEventTypeId().intValue());
            mEventPreparedStatement.setInt(3, lEvent.getEventTime().intValue());
            mEventPreparedStatement.setString(4, lEvent.getEventTechnicalHost());
            mEventPreparedStatement.setString(5, lEvent.getEventLogicalHost());
            mEventPreparedStatement.setInt(6, lEvent.getEventAssetType().intValue());

            // Execution des prepared statements
            lSqlResultSet.incrementInsertedRowNumber(mEventPreparedStatement.executeUpdate());
            lSqlResultSet.incrementExecutedQueryNumber(1);

            // Selon le type d'event
            if (lEvent instanceof ProcessEvent) {
                // Alimentation de la requête
                mProcessEventPreparedStatement.setInt(1, lCurrentSeqVal);
                mProcessEventPreparedStatement.setString(2, ((ProcessEvent) lEvent).getProcessName());
                mProcessEventPreparedStatement.setString(3, ((ProcessEvent) lEvent).getProcessStepName());
                mProcessEventPreparedStatement.setString(4, ((ProcessEvent) lEvent).getProcessExecutionId());
                mProcessEventPreparedStatement.setString(5, ((ProcessEvent) lEvent).getApplicationSessionId());
                mProcessEventPreparedStatement.setString(6, ((ProcessEvent) lEvent).getViewSessionId());

                // Execution des prepared statements
                lSqlResultSet.incrementInsertedRowNumber(mProcessEventPreparedStatement.executeUpdate());
            }
        }
    }
}

```

Récupération du curseur d'évènement

Enregistrement de la spécialisation

Enregistrement de la généralisation selon le type d'évènement

Figure 53 : Extrait de la méthode *saveEventListTransactionalMode* de la classe *EventManager*

Pour faciliter la gestion des requêtes SQL nous avons créé une classe utilitaire faisant partie du Framework Java (projet Java *FWK.Java.Framework*). Elle permet de retourner aux classes appelant un *manager* SQL, les informations suivantes :

- ▶ Nombre de lignes mises à jour en base de données
- ▶ Nombre de lignes insérées en base de données
- ▶ Nombre de lignes supprimées en base de données
- ▶ Nombre de requêtes exécutées
- ▶ Liste des séquences Oracle créés

Ceci afin de permettre à la classe appelante de connaître précisément le résultat de requêtes exécutées et agir en conséquence (exemple : retourner une erreur si aucune ligne n'a été insérée ou utiliser la séquence Oracle créée pour mettre à jour une autre table).

### Moteur de corrélation des évènements

Le moteur de corrélation des évènements permet de faire le lien entre un évènement unitaire survenu sur le SI, un objet du référentiel et son instance afin de construire un ensemble de statistiques. Il est lancé à intervalles réguliers par le système afin de traiter les évènements recueillis par le moteur de collecte. L'intervalle de lancement est réglé en fonction de la capacité de traitement du système en production et surtout en fonction de la volumétrie des données échangées (taille et fréquence moyennes et maximales). Une volumétrie d'échanges pour un service par appel unitaire ou consolidé à l'ensemble d'un SI, peut être notée de la sorte :

Flux entrant dans l'ISM	Volumétrie		Fréquence	
	Moyenne	Maximale	Moyenne	Maximale
<i>Service passerCommande</i>	48 ko / instance	512 ko / instance	1 200 par jour	4 500 par jour
<i>Service creerCompteClient</i>	110 ko / instance	120 ko / instance	800 par jour	2 500 par jour
<i>Process expedition</i>	220 ko / instance	280 ko / instance	1 200 par jour	4 500 par jour

**Tableau 4 : Exemples d'analyse de volumétrie par service**

Grâce à ce type d'étude et la connaissance de l'architecture physique de production, nous sommes en mesure de calibrer la fréquence d'appel du moteur de corrélation des évènements afin de proposer aux utilisateurs des statistiques les plus à jour possible, tout en supportant la charge inhérente aux pics d'activité du SI. En effet, il est préférable que le moteur de corrélation ait fini de traiter totalement un lot d'évènements pendant son intervalle avant le lancement d'un autre appel ; afin de ne pas retarder celui-ci et prendre du retard sur les traitements.

Afin de mieux comprendre le rôle du moteur de corrélation des évènements, prenons comme hypothèse un référentiel logique disposant d'un objet de type service permettant à un client de passer une commande, nous nommerons ce service *ServiceCommandeVI*. Ce service a été déployé sur le serveur physique *LYONSERV01* en environnement de production, nous nommerons cette instance *ServiceCommandeVI\_001*.

Le 01/06/2010 à 14h16 un client A sollicite l'instance *ServiceCommandeVI\_001* pour passer commande à travers le site Web de notre société fictive. Le moteur de collecte des évènements va alors recevoir un premier évènement de type *Service Start* et l'enregistrer en base de données au niveau des tables *EVENT* et *SERVICE\_EVENT*. Le moteur de corrélation des évènements qui est appelé dans cet exemple toutes les 10 minutes va détecter à 14h20 l'évènement en question. Il va alors grâce au nom du service contenu dans la table *SERVICE\_EVENT* et les informations sur la machine physique contenues dans la table *EVENT*, déterminer qu'il s'agit d'un évènement *Service Start* sur l'instance *ServiceCommandeVI\_001*, celle-ci hébergée sur le serveur physique *LYONSERV01*. Ces informations vont alors générer un nouvel enregistrement dans les tables *ASSET\_EXECUTION\_STATISTIC* et *SERVICE\_EXECUTION\_STAT* ainsi que mettre à jour le nombre d'appel du service dans la table *SERVICE\_STATISTIC*.

Le client A reçoit à 14h17 à travers le site Web une confirmation de la bonne exécution de sa commande via la réponse apportée par l'instance *ServiceCommandeVI\_001*. Le moteur de collecte des évènements reçoit alors un évènement de type *Service Stop* qu'il va stocker en table *EVENT* et *SERVICE\_EVENT*. Le moteur de corrélation va prendre en compte cet évènement dans son passage à 14h20 à la suite de l'évènement *Service Start* et alors mettre à jour la table *SERVICE\_EXECUTION\_STAT* avec le statut « *Completed* » et l'heure de fin de l'appel, puisque l'instance du service s'est bien terminée.

Grâce à ce mécanisme nous sommes en mesure de connaître l'ensemble des appels effectué pour un service, leur état (% réussi, % en erreur et % en cours) ainsi que les temps de réponse minimal, moyen et maximal constatés sur une période.

Grâce aux évènements retournés pour chaque appel à un composant il est possible de déterminer le niveau de performance du composant en comparant un appel à la moyenne des appels avec pondération sur la volumétrie A/R mais aussi au temps de traitement théorique défini de manière globale pour la technologie et la volumétrie.

Il est alors possible d'afficher l'évolution des performances du composant en fonction du temps.

### **Moteur d'orchestration des tâches et de remontée des alertes**

Le moteur d'orchestration des tâches a pour rôle de lancer l'ensemble des tâches systèmes :

- ▶ Purge de la base des évènements selon le paramétrage saisi
- ▶ Purge de la base des statistiques selon le paramétrage saisi
- ▶ Appel du moteur de corrélation des évènements selon le paramétrage saisi
- ▶ Etc...

Ce moteur d'orchestration permet aussi de gérer les tâches programmées par l'utilisateur comme les alertes de coupures de service. En effet, au niveau de chaque instance d'un service, nous pouvons déterminer l'état du composant en réalisant un appel de contrôle (connexion à une *queue* d'entrée, récupération d'un WSDL d'un Web Service, affichage de l'écran principal d'une application, ping du serveur physique, connexion au serveur logique...). Cet état est calculé à l'affichage du composant en question sur l'application, il est alors possible pour un utilisateur de programmer une surveillance régulière de l'état d'un composant avec déclenchement d'une alerte en cas de problème ainsi qu'une conservation des états retournés dans le temps.

## Sous-chapitre 4 – Application de supervision

Au lieu d’opter pour la gestion des Exceptions au niveau d’un fichier de journalisation en utilisant par exemple la librairie log4j, nous avons préféré mettre en place d’une API de supervision interne.

Cette API est composée d’un ensemble de Web Services permettant notamment :

- La déclaration d’une erreur technique ou applicative
- Le suivi des traitements par lot (envoi des e-mails d’alerte, envoi des rapports d’activité...)

Ces différents Web Service permettent alors la sauvegarde en base de données de ces indicateurs de suivi selon le modèle de données ci-dessous :

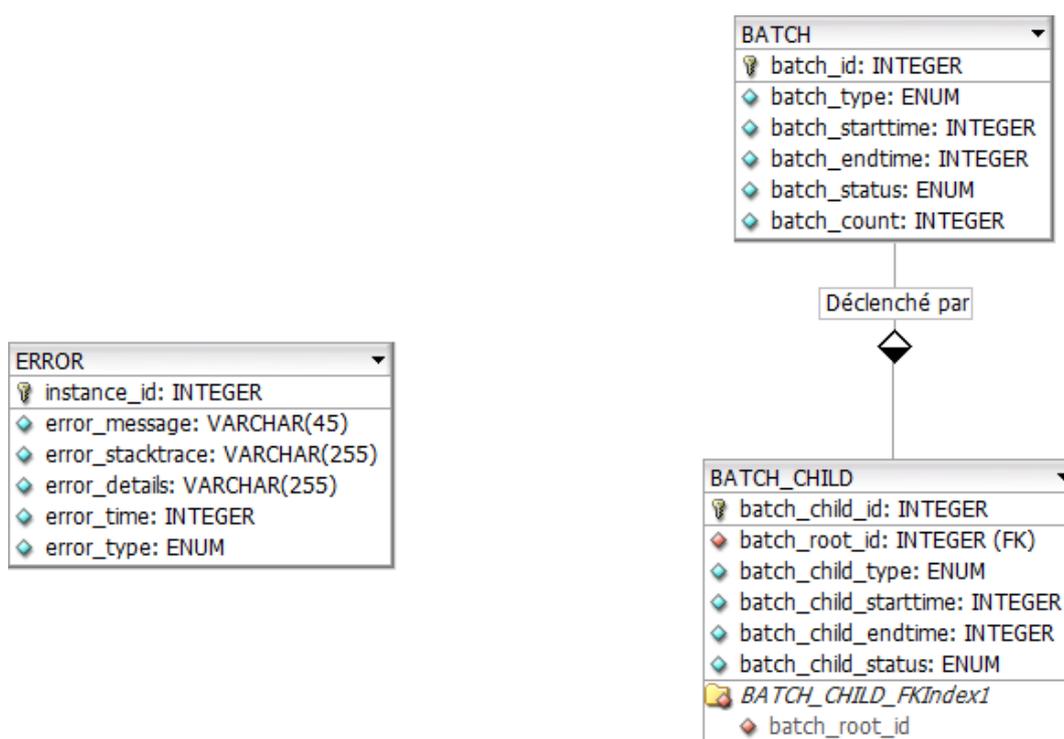


Figure 54 : Modèle de données de la partie supervision

Le Web Service d’enregistrement des erreurs expose une méthode *declare* permettant la déclaration d’une nouvelle erreur en respectant le format XSD défini ci-dessous :

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://xml.netbeans.org/schema/Error"
  xmlns:tns="http://xml.netbeans.org/schema/Error"
  elementFormDefault="qualified">
  <xsd:element name="Error">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="instance_id" type="xsd:long"></xsd:element>
        <xsd:element name="error_message" nillable="true" type="xsd:string" />
        <xsd:element name="error_stacktrace" nillable="true" type="xsd:string" />
        <xsd:element name="error_details" type="xsd:string" />
        <xsd:element name="error_time" nillable="true" type="xsd:integer" />
        <xsd:element name="error_type" nillable="true" type="xsd:integer" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

Figure 55 : XSD de déclaration d'une erreur

La partie Administration de l'application ISM permet la visualisation des erreurs applicative et la supervision des traitements par lot.

The screenshot shows the 'Suivi des erreurs' (Error Tracking) page. At the top, there is a navigation bar with 'Administration' selected. Below it, there are tabs for 'Erreurs', 'Traitements', and 'Utilisateurs'. The main content area is titled 'Suivi des erreurs' and contains a search filter section. The filter includes a dropdown for 'Type d'erreur' set to 'Fonctionnelle', a 'Message' input field, and date range inputs for 'Plage de début' (01/02/2010) and 'Plage de fin' (31/06/2010). A 'Filtrer' button is present. Below the filter, it states '3 résultats retournés pour cette recherche'. A table displays the following data:

Error ID	Type	Date	Message	Détails	Pile d'appel
0000029	Fonctionnelle	02/06/2010 11h12s23	Adresse e-mail erronée	S.O.	S.O.
0000031	Fonctionnelle	04/06/2010 19h25s12	Informations profil utilisateur manquantes	S.O.	S.O.
0000076	Fonctionnelle	09/06/2010 09h07s08	Adresse e-mail erronée	S.O.	S.O.

At the bottom of the page, it shows 'Connecté en tant que: admin' and 'Axopen | Plateforme de monitoring pour SI'.

Figure 56 : Maquette de l'écran de suivi des erreurs

## Conclusion

La construction d'une application de supervision du SI est un projet important nécessitant des connaissances pointues en réseau information, architecture applicative et infrastructure. De plus la charge de travail importante nécessite une équipe de 3 à 4 personnes à plein temps pendant près de deux ans.

Etant une jeune entreprise, la solution que nous avons choisie pour pouvoir amortir les frais relatifs à un tel projet a été de constituer une équipe composée de collaborateurs étant à mi-temps sur d'autres projets.

Cette solution nous a permis de maintenir un équilibre financier tout en permettant le développement de cette application. En contre partie, ce choix a limité in fine la capacité de production de l'équipe de développement et nous a contraint à être très vigilant sur le planning.

Au regard de la taille du projet et des spécialités requises pour ces différentes tâches, nous avons découpé sa réalisation en deux parties. En effet le lot 1 présenté ci-dessous requiert des connaissances plus pointues en architecture logicielle et en conception métier.

1<sup>er</sup> Lot (Architecture et spécifications) : De Septembre 2009 à Septembre 2010 :

- ▶ Conception des écrans et maquettage.
- ▶ Conception du schéma de base de données.
- ▶ Architecture de la solution.
- ▶ Ecriture des scripts de génération des tables Oracle.
- ▶ Développement des entités métiers et managers Java.
- ▶ Développement de la solution de collecte des évènements.
- ▶ Développement du Framework technique Java comprenant les classes utilitaires permettant la gestion de la base de données, des fichiers, du bus JMS, de la sérialisation XML.
- ▶ Développement du moteur de lancement des tâches comme par exemple l'envoi des alertes par e-mail ou SMS ou la purge des bases de données.

2<sup>ème</sup> Lot (Ecrans et recette) : De Septembre 2010 à Septembre 2011 :

- ▶ Développements des services métiers de l'application (Web Services).
- ▶ Mise en œuvre du réseau d'agents.
- ▶ Développement des écrans en méthode agile.
- ▶ Prise en compte des impacts au niveau du modèle de données.
- ▶ Tests et recette de l'application.
- ▶ Tests de performances.

Pour ma part, ma contribution se limite à la réalisation du lot 1 où les notions d'architecture requises ainsi que la connaissance métier y sont plus importantes. C'est ce lot qui a été présenté dans ce document. Le lot 2 est quant à lui confié à une deuxième équipe moins expérimentée mais bénéficiant des travaux réalisés dans le lot 1.

Comme on peut le voir sur l'illustration ci-dessous ainsi que le planning de développement présenté en annexe 1, l'ensemble des tâches et développements prévus dans le lot 1 ont été réalisés. Bien entendu, le développement du lot 2 aura des impacts mineurs sur les composants du lot 1. En effet, il est quasiment impossible lors de tout projet informatique d'éviter à avoir à modifier le modèle de données lors du développement des écrans de l'application concernée. Ce qui est primordial est de prévoir le délai et l'équipe pour prendre en compte ces impacts et leur offrir une structure agile (générateur de code, Framework de mapping Objet-Base de données...) pour en limiter la charge.

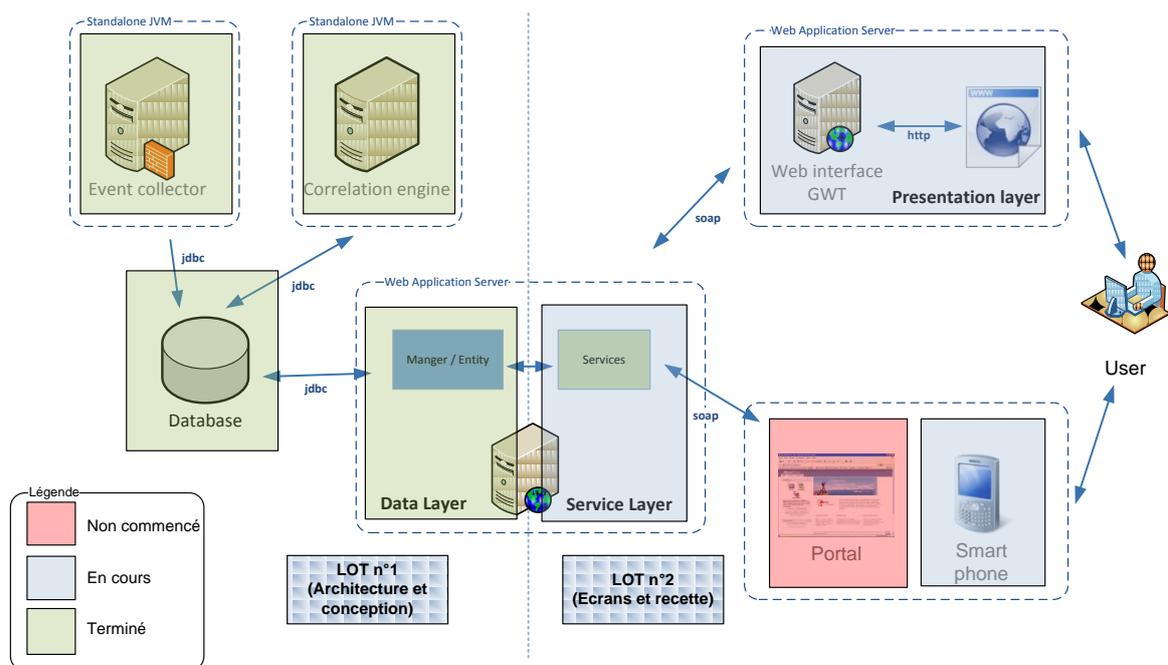


Figure 57 : Aperçu de l'avancement des développements

En informatique de gestion, les petites structures de par leur taille adaptée et leur capacité à lever des fonds pour un projet, contribuent activement au développement de nouvelles solutions.

Lorsque l'opération est réussie ces entreprises sont souvent rachetées par de plus grosses structures pour faciliter la commercialisation de leurs produits mais surtout étendre le catalogue et le nombre de clients de l'entreprise réalisant l'acquisition. L'entreprise Américaine IBM, qui est le N°1 mondial du secteur, a par exemple réalisé plus de 50 acquisitions d'entreprise depuis 2001.

Cette vision reste cependant souvent limitée aux Etats-Unis ; dans l'hexagone les petites structures manquent cruellement d'investisseurs. En effet, la prise de risque est beaucoup plus limitée dans nos frontières.

Il est alors difficile pour les petites entreprises françaises d'innover en informatique.

## Bibliographie

### Ouvrages imprimés :

BONNET Pierre, DETAVERNIER Jean-Michel et VAUQUIER Dominique. *Le Système d'information durable*. Lavoisier, 2008, 307 p. (Etudes et Logiciels informatiques)

RIVARD François, ABOU HARB Georges et MERET Philippe. *Le Système d'information transverse, nouvelles solutions du SI et performance métier*. Lavoisier, 2008, 296 p. (Management et informatique)

REGNIER-PECASTAING Frank, GABASSI Michel et FINET Jacques. *MDM, Enjeux et méthode de la gestion de données*. Dunod, 2008, 280 p. (Management des systèmes d'information)

GROJEAN Pascal, MOREL Médéric et PLOUIN Guillaume. *Performance des architectures IT. Disponibilité, temps de réponse, robustesse et montée en charge*. Dunod, 2007, 261 p. (Management des systèmes d'information)

### Sites web :

DSI – CNRS – Cartographie des SI d'appui à la recherche. In : CNRS. Direction des systèmes d'information, [en ligne]. Disponible sur : <http://www.dsi.cnrs.fr/si/cartographies/> (consulté le 12/05/2010)

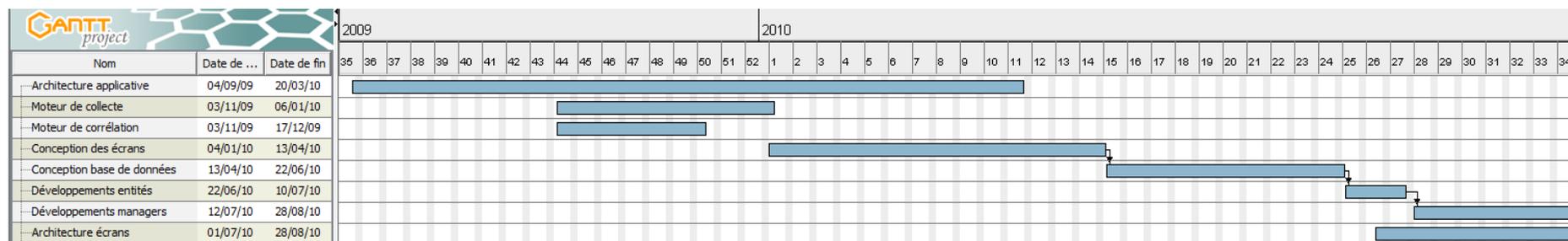
## **Table des annexes**

Annexe 1 Planning de développement des lots 1 et 2 de l'application .....	85
Annexe 2 Application de supervision développée pour l'entreprise ST Ericsson.....	86
Annexe 3 Application de supervision développée pour l'entreprise Boiron.....	89

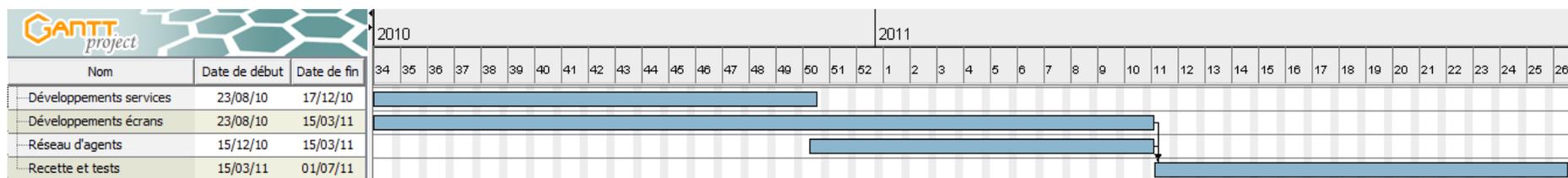
## Annexe 1

### Planning de développement des lots 1 et 2 de l'application

Voici le planning de développement du lot 1 de l'application :



Voici le planning prévisionnel de développement du lot 2 de l'application :



## Annexe 2

### Application de supervision développée pour l'entreprise ST Ericsson

Ecran de supervision de l'exécution des processus :

Process Monitor
Statistics
Preferences
Services Framework Error
Batch Monitor

**BPEL Instance (6)**

	BPEL Name	Status	Start Time	End Time	Updated Time
	ONRAMP_STE_IRQA	FAULTED	Sep 3, 2009 2:53:14 PM	Sep 3, 2009 2:53:16 PM	Sep 3, 2009 2:53:16 PM
	OFFRAMP_NOKIA_FTP	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:52 PM	Sep 3, 2009 1:16:52 PM
	BUSINESSFLOW_EXPORT_REQUIREMENT	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM
	ONRAMP_STE_IRQA	COMPLETED	Sep 3, 2009 1:16:40 PM	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM
	ONRAMP_STE_IRQA	FAULTED	Sep 3, 2009 1:07:38 PM	Sep 3, 2009 1:07:38 PM	Sep 3, 2009 1:07:38 PM
	ONRAMP_STE_IRQA	FAULTED	Sep 3, 2009 1:01:01 PM	Sep 3, 2009 1:01:03 PM	Sep 3, 2009 1:01:03 PM

**BPEL Activity (6)**

Activity	Iteration	Status	Start Time	End Time
ReceiveFromIrq	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:4
AssignToVerue	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:4
CallVerue	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:4
GiveOffRamp	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:4
SendToOffRamp	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:4

```

graph LR
    Start(( )) --> R1[ReceiveFromOnramp]
    R1 --> R2[GiveCorrelation]
    R2 --> R3[InsertCorrelation]
    R3 --> R4[GiveOffRamp]
    R4 --> R5[SendToOffRamp]
    R5 --> End(( ))
    
```

Ecran d'affichage des erreurs survenues durant l'exécution d'un processus :

<b>Process Monitor</b>	<b>Statistics</b>	<b>Preferences</b>	<b>Services Framework Error</b>	<b>Batch Monitor</b>	
<b>Services Framework Error (1)</b>					
<b>Instance Id</b>	<b>Bpel Id</b>	<b>Message</b>	<b>Stack Trace</b>	<b>Error Time</b>	<b>Details</b>
test	test	test	test	Jan 1, 2010 12:00:00 AM	test

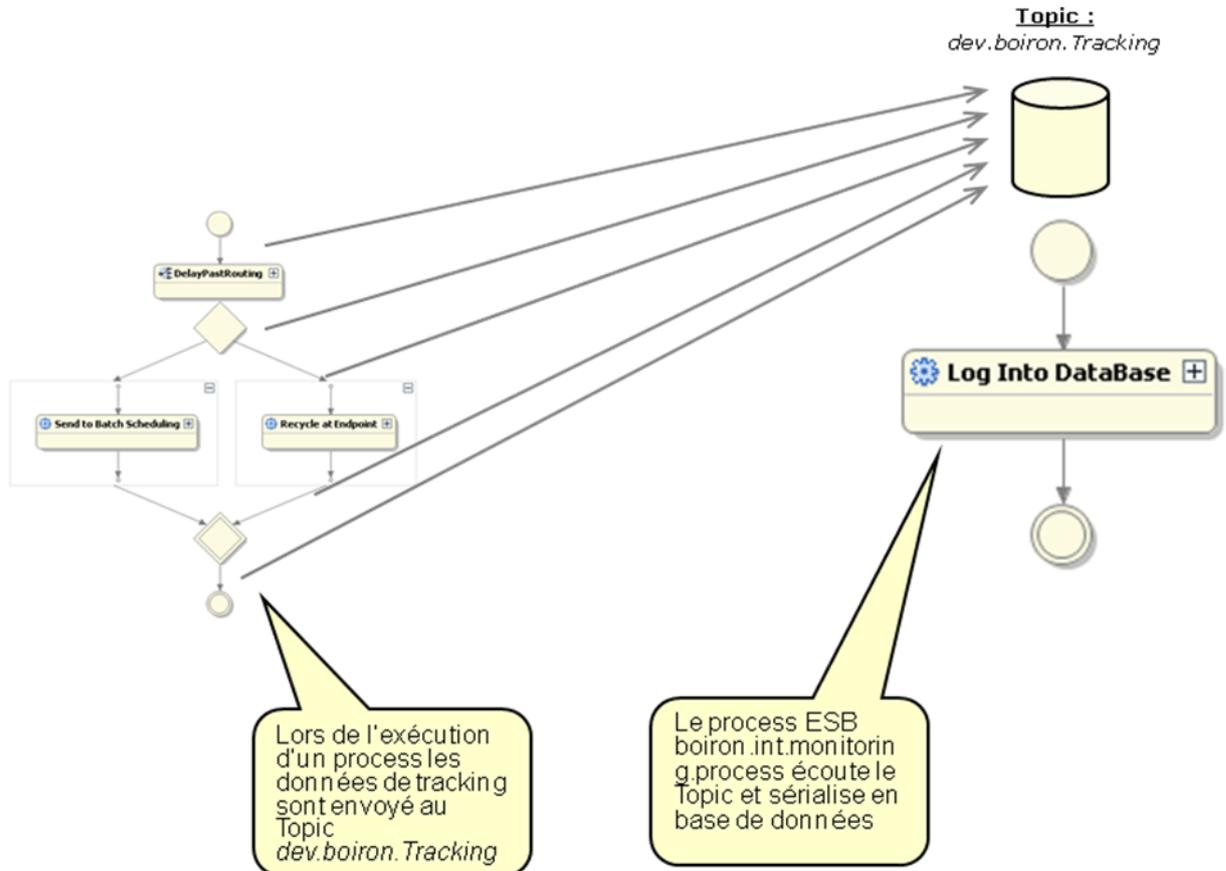
Écran de supervision de l'exécution des traitements par lot (Batch) :

Process Monitor	Statistics	Preferences	Services Framework Error	Batch Monitor		
<b>Batch Monitor (2)</b>						
						
Batch Id	Batch Type	Start Time	End Time	Directory	Status	Count
 export_requirement_test_2009-01-02	export_requirement_test	Sep 3, 2009 12:57:41 PM	Sep 3, 2009 12:57:44 PM	export_requirement_test/2009-01-02	COMPLETED	
 export_requirements_1251985303975	export_requirements	Sep 3, 2009 3:41:43 PM		export_requirements/1251985303975	IN_PROGRESS	
<b>Batch Child (1)</b>						
Batch Id	Object Name	Status				
 export_requirement_test_2009-01-02	STE_NOK_REQ_MB-417-4156_35	COMPLETED				
<b>Bpel Instance (1 - 3 of 3)</b>						
						
Bpel Name	Status	Start Time	End Time	Updated Time	Instance Id	
 ONRAMP_STE_IRQA	COMPLETED	Sep 3, 2009 1:16:40 PM	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM	10.129.202.246:1d6ac7be:1237f991bff:-7fad	
 BUSINESSFLOW_EXPORT_REQUIREMENT	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM	10.129.202.246:1d6ac7be:1237f991bff:-7fac	
 OFFRAMP_NOKIA_FTP	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:52 PM	Sep 3, 2009 1:16:52 PM	10.129.202.246:1d6ac7be:1237f991bff:-7fab	
<b>BPPEL Activity (6)</b>						
Activity	Iteration	Status	Start Time	End Time		
Not Found	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM		
Not Found	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM		
Not Found	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM		
Not Found	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM		
Not Found	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM		
Not Found	0	COMPLETED	Sep 3, 2009 1:16:45 PM	Sep 3, 2009 1:16:45 PM		

# Annexe 3

## Application de supervision développée pour l'entreprise Boiron

Architecture de la solution de supervision des processus ESB :



Base de données de stockages des étapes de processus :

MONITORSIMPLEPROPERTY ▾	
🔑	INSTANCEID: VARCHAR
🔑	PROPNAME: VARCHAR
🔑	SCOPEID: INTEGER
🔑	VARID: INTEGER
🔗	PROPVOLUME: VARCHAR

MONITORSIMPLEVARIABLE ▾	
🔑	INSTANCEID: VARCHAR
🔑	SCOPEID: INTEGER
🔑	VARNAME: VARCHAR
🔗	DATEVALUE: TIMESTAMP
🔗	NUMVALUE: INTEGER
🔗	STRVALUE: VARCHAR
🔗	VARID: INTEGER
🔗	VARTYPE: CHAR

SERVICEUNIT ▾	
🔑	SUNAME: VARCHAR
🔗	LASTUPDATETIME: TIMESTAMP
🔗	SUZIPARCHIVE: BLOB

MONITORNMPROPERTY ▾	
🔑	INSTANCEID: VARCHAR
🔑	SCOPEID: INTEGER
🔑	VARID: INTEGER
🔑	PROPNAME: VARCHAR
🔗	PROPVOLUME: BLOB

MONITORBPPELVARIABLE ▾	
🔑	INSTANCEID: VARCHAR
🔑	SCOPEID: INTEGER
🔑	VARNAME: VARCHAR
🔗	ISFAULT: CHAR
🔗	VARVALUE: BLOB

MONITORBPPELINSTANCE ▾	
🔑	INSTANCEID: VARCHAR
🔗	BPPELID: VARCHAR
🔗	STATUS_2: VARCHAR
🔗	ENDTIME: TIMESTAMP
🔗	ENGINEID: VARCHAR
🔗	STARTTIME: TIMESTAMP
🔗	UPDATEDTIME: TIMESTAMP

MONITORVARIABLEATTACHMENT ▾	
🔑	INSTANCEID: VARCHAR
🔑	NAME: VARCHAR
🔑	SCOPEID: INTEGER
🔑	VARID: INTEGER
🔗	ATTACHMENT: BLOB

MONITORBPPELACTIVITYVARIABLE ▾	
🔑	ACTIVITYID: INTEGER
🔑	INSTANCEID: VARCHAR
🔑	VARNAME: VARCHAR
🔑	VARTYPE: VARCHAR
🔗	ISFAULT: CHAR
🔗	VARDATATYPE: VARCHAR
🔗	VARID: INTEGER
🔗	VARVALUE: BLOB

MONITORBPPELACTIVITY ▾	
🔑	ACTIVITYID: INTEGER
🔑	INSTANCEID: VARCHAR
🔑	ITERATION: INTEGER
🔗	ACTIVITYXPATH: VARCHAR
🔗	CRMPINVOKEID: VARCHAR
🔗	CRMPRECEIVEID: VARCHAR
🔗	ENDTIME: TIMESTAMP
🔗	ENGINEID: VARCHAR
🔗	HASFAULTED: CHAR
🔗	STARTTIME: TIMESTAMP
🔗	STATUS_2: VARCHAR

## Liste des figures

Figure 1 : Logo de l'entreprise Axopen.....	10
Figure 2 : Organisation Axopen .....	11
Figure 3 : Offres de l'entreprise Axopen.....	12
Figure 4 : Illustration de l'hétérogénéité des systèmes d'information .....	17
Figure 5 : Liste des traitements mutualisables dans un SI .....	22
Figure 6 : Illustration du caractère cyclique de la gouvernance du SI .....	23
Figure 7 : Aperçu d'une méthodologie permettant de définir un schéma directeur .....	24
Figure 8 : Illustration de la notion de disponibilité.....	25
Figure 9 : Méthodes de calcul de la disponibilité .....	28
Figure 10 : Caractère exponentiel des coûts de haute disponibilité .....	29
Figure 11 : Périmètre des différentes solutions de supervision.....	32
Figure 12 : Aperçu de l'outil Actional commercialisé par Progress Software .....	35
Figure 13 : Couverture de l'offre de supervision de Systar .....	36
Figure 14 : Périmètre de la solution .....	38
Figure 15 : Maquette de l'écran de choix d'un profil .....	39
Figure 16 : Maquette de l'écran « Gouvernance - Tableau de bord ».....	40
Figure 17 : Maquette de l'écran « Statistiques de gouvernance ».....	41
Figure 18 : Maquette de l'écran « Tableau de bord d'étude métier » .....	42
Figure 19 : Maquette de l'écran « Statistiques d'étude métier ».....	43
Figure 20 : Maquette de l'écran « Cartographie des applications » .....	43
Figure 21 : Maquette de l'écran « Chemins de navigation » .....	44
Figure 22 : Maquette de l'écran « Cartographie des objets » .....	45
Figure 23 : Maquette de l'écran « Cycle de vie des objets » .....	45
Figure 24 : Maquette de l'écran « Cartographie des serveurs logiques » .....	46
Figure 25 : Maquette de l'écran « Cartographie des serveurs physiques » .....	47
Figure 26 : Illustration de l'architecture logique de la solution .....	50
Figure 27 : Illustration des différents niveaux de distribution .....	51
Figure 28 : Positionnement du serveur JBOSS sur le « cadrant magique » du Gartner .....	53
Figure 29 : Extrait du fichier propriété de configuration de l'accès JDBC.....	54
Figure 30 : Console Web d'administration d'Active MQ.....	55
Figure 31 : Processus de développement avec le Framework GWT.....	56
Figure 32 : Architecture n-tiers logique .....	57
Figure 33 : Rôle des différents environnements .....	58
Figure 34 : Illustration du fonctionnement de la plateforme d'intégration continue .....	59
Figure 35 : Aperçu du lancement manuel d'une tâche dans l'outil Hudson.....	59
Figure 36 : Illustration du processus de monitoring .....	61
Figure 37 : Code source de la structure Enum permettant de définir le type d'évènement .....	62
Figure 38 : Modèle de données de stockage des évènements .....	62
Figure 39 : Modèle de données de stockage du référentiel logique .....	63
Figure 40 : Modèle de données de stockage du référentiel technique.....	64
Figure 41 : Modèle de données de stockage des statistiques .....	65
Figure 42 : Modèle de données de stockage des paramètres.....	66
Figure 43 : Illustration de la généralisation des objets du référentiel.....	67
Figure 44 : Fonctionnalité d'export de l'outil DBDesigner 4 .....	68
Figure 45 : Extrait du programme de génération du Script de création des tables Oracle.....	69
Figure 46 : Extrait du programme de génération des entités Java.....	70
Figure 47 : Principe de fonctionnement d'un réseau d'agents .....	71
Figure 48 : Architecture du moteur de collecte des évènements.....	73
Figure 49 : Extrait de la boucle principale du programme de collecte des évènements .....	74
Figure 50 : Extrait de la méthode <i>start</i> du listener d'évènements.....	74
Figure 51 : Extrait de la méthode <i>run</i> du listener d'évènements .....	75
Figure 52 : Extrait de la méthode <i>start</i> du manager d'insertion des objets en base .....	76
Figure 53 : Extrait de la méthode <i>saveEventListTransactionalMode</i> de la classe <i>EventManager</i> ...	76

Figure 54 : Modèle de données de la partie supervision .....	79
Figure 55 : XSD de déclaration d'une erreur.....	80
Figure 56 : Maquette de l'écran de suivi des erreurs .....	80
Figure 57 : Aperçu de l'avancement des développements.....	82

## Liste des tableaux

Tableau 1 : Liste des abréviations .....	4
Tableau 2 : Glossaire.....	5
Tableau 3 : Exemples de coûts d'indisponibilité pour des applications critiques .....	29
Tableau 4 : Exemples d'analyse de volumétrie par service .....	77

## **Etude, conception et développement d'une solution de supervision / gouvernance du système d'information.**

**Mémoire d'Ingénieur C.N.A.M., Lyon 2010**

---

### **RESUME**

Dans un monde où l'économie tend à se mondialiser et où le changement permanent est devenu la norme, le système d'information des entreprises doit s'adapter à ces nouvelles contraintes. Il doit être plus réactif pour aider les métiers à mettre sur le marché le plus rapidement possible de nouveaux produits tout en limitant la consommation de ressources (*green computing* ou *green IT*).

Il est donc primordial pour toute entreprise s'appuyant fortement sur son système d'information pour la création de valeur d'avoir une vision exhaustive sur sa composition (cartographie), son évolution (schéma directeur) mais aussi son état de fonctionnement (supervision de l'exécution).

Ce sont autant de domaines sous-tendus par la supervision/gouvernance du système d'information.

Cette étude a pour but de répondre à cette problématique grâce au développement d'une solution sur mesure.

**Mots clés : Système d'information, gouvernance, supervision, cartographie, schéma directeur.**

---

### **SUMMARY**

In a world where the economy becomes more and more globalized and where constant change is the norm, the information system of many companies must adapt to these constraints. It should be more responsive to business to help companies to bring to the market new products as quickly as possible while limiting the consumption of resources (*green computing* or *green IT*).

It is therefore crucial for any company relying heavily on its information system for creating value to have a comprehensive view on its composition (mapping), its development (master plan) but also its operational condition (execution monitoring).

These are all areas that are underpinned by monitoring / governance of information system.

This study aimed to answer this problem by developing a custom made solution.

**Key words: information system, governance, monitoring, mapping, master plan.**