N° d'ordre : 807-2016/SEIMSDE/EDSTII Année universitaire : 2016

UNIVERSITÉ D'ANTANANARIVO

ÉCOLE SUPÉRIEURE POLYTECHNIQUE D'ANTANANARIVO

ECOLE DOCTORALE EN SCIENCES ET TECHNIQUES DE L'INGENIERIE ET DE L'INNOVATION

Laboratoire : Systèmes Embarqués – Instrumentation – Modélisation des Systèmes et Dispositifs Electroniques (SE-I-MSDE)

THÈSE de DOCTORAT

pour l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ D'ANTANANARIVO

Domaine : **Sciences de l'Ingénieur**

Spécialité **: Modélisation des Systèmes et Dispositifs Electroniques**

Intitulée : **Du Design Pattern aux applications Entreprises**

Par

RAZAFINDRAIBE Tovohery Andriampamonjy Alain

Soutenue le 16 **Juin 2017**, devant la Commission d'Examen composée de :

Examinateurs :

M. ANDRIANAHARISON Yvon, Professeur Titulaire
M. RANDRIAMITANTSOA Paul Auguste, Professeur Titulaire
M. RAKOTOMIRAHO Soloniaina, Professeur

Rapporteurs :

M. RATIARISON Adolphe, Professeur Titulaire, Rapporteur Externe
M. RANDIMBINDRAINIBE Falimananana, Professeur, Rapporteur Interne

Directeur de thèse : **M. RASTEFANO Elisée, Professeur**

Co-Directeur de thèse : **Mme RABEHERIMANANA Lyliane Irène, Maîtres de Conférences**

Remerciements

Je rends grâce à notre Seigneur Dieu parce qu'il nous a comblé de ses bienfaits : santé, courage, et miséricorde.

Je tiens à remercier principalement avec sincérité :

- *Madame **RAMIARISON MAVOARILALA Claudine**, Professeur, Directeur Générale de la Recherche Scientifique au sein du Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.*

- *Monsieur **RAMANOELINA Panja**, Professeur Titulaire, Président de l'Université d'Antananarivo*

*J'adresse mes vifs remerciements à mon directeur de thèse, Monsieur **RASTEFANO Elisée**, et mon co-encadreur, Madame **RABEHERIMANANA Lyliane Irène**, pour avoir dirigé et encadré mes travaux de recherches, ainsi que pour leurs conseils certainement très utiles pour mon avenir. Je suis aussi très reconnaissant envers eux pour leur soutien et leurs encouragements permanents.*

*Je tiens à remercier profondément le Professeur **ANDRIANAHARISON Yvon** qui a su, malgré ses lourdes responsabilités et son emploi du temps extrêmement chargé, m'apporter son soutien et me faire l'honneur de participer à la soutenance de cette thèse.*

*Je suis également tout à fait honoré de la présence à ce jury du Professeur **RANDRIAMITANTSOA Paul Auguste** et du Professeur **RAKOTOMIRAHO Soloniaina**, et je les remercie vivement pour l'intérêt qu'ils ont accordé à mon travail et d'avoir accepté de participer à ce jury.*

*J'adresse mes sincères remerciements au Professeur **RATIARISON Adolphe** et au Professeur **RANDIMBINDRAINIBE Falimananana**, pour m'avoir fait l'honneur d'étudier mes travaux de thèse et de participer à ce jury en qualité de rapporteurs.*

Je ne saurais terminer sans que soient remerciées toutes les personnes qui, de près ou de loin ont contribué à la réussite de ce travail, par leur soutien moral, administratif, technique.

*Finalement, je suis très reconnaissant envers ma femme **Harilanto**, mes trois enfants **Norah**, **Tolotra**, **Carolle** et toute ma famille qui ne m'a jamais oublié dans leurs prières et pour leur soutien tant matériel que moral. Aussi, je remercie tous mes amis pour leurs soutiens et prières.*

Teny fisaorana

Faly midera sy mankalaza ny Tompo Andriamanitra ny fanahiko, fa efa nanao zava-dehibe tamiko izy.

Etoana dia manantitra ny fisaorana manokana :

- *Ramatoa **RAMIARISON MAVOARILALA Claudine**, Tale Jeneralin'ny Fikarohana eo anivon'ny Ministeran'ny Fampianarana Ambony sy ny Fikarohana Siantifika.*
- *Andriamatoa **RAMANOELINA Panja**, Filohan'ny Oniversiten' Antananarivo*

*Izao asa fikarohana izao no tontosa dia teo ny tari-dalana sy ny fiaraha-miasa. Isaorako manokana etoana Andriamatoa **RASTEFANO Elisée**, sy ny mpiara-miasa aminy, Ramatoa **RABEHERIMANANA Lyliane Irène**, izay tsy nikely soroka tamin'ireo toro-hevitra maro sy ny fanampiana isankarazany taloha sy nandritra ny fotoana namitana izao asa fikarohana izao.*

*Eto koa izaho dia manantitra ny fisaorana sy fankatelemana ho an' Andriamatoa **ANDRIANAHARISON Yvon** izay nanome lanja izao asa fanohanana fikarohana izao ka nanaiky handray anjara amin'ny fanadihadiana sy ny fitsarana ny asa na dia teo aza ireo adidy sy andraikitra maro sahanina.*

*Manaraka izany dia velom-pisaorana koa izaho ho an' Andriamatoa **RANDRIAMITANTSOA Paul Auguste** sy Andriamatoa **RAKOTOMIRAHO Soloniaina**, raha nanome ny traikefa ilaina izy ireo nandritra ny fotoana nianarana sy nanaiky koa handray anjara amin'ny famakafakana ity antontan-kevitra ity.*

*Misaotra sy mankasitraka an' Andriamatoa **RATIARISON Adolphe** sy Andriamatoa **RANDIMBINDRAINIBE Falimananana**, tamin'ny fanekena handalina manokana izao asa fikarohana izao sy hanao tatitra momba izany.*

Ny Fisaorana toy ny fary lava vany ka tsy lany amamiana, koa tsy haiko ny tsy hisaotra ireo rehetra izay nandray anjara sy nanampy tamin'ny fampianarana, fikarakarana, fanampiana samihafa ka nahatotosa ity asa fikarohana ity.

*Ny asa vadi-drano tsy vita raha tsy ifanakonana, ary ny akoho tsy tapa-doha raha tsy fanampahankivitry ny mpivady; koa eto ampamaranana dia maneho ny hafaliam-poko aho ho an-dRamatoa **Harilanto** vadiko, ny zanako telo mianadahy **Norah**, **Tolotra**, **Carolle,** ny Ray aman-dReny, ny fianakaviana manontolo sy ny tapaka sy namana izay tsy nanadino ahy tamin'ny asa natao, tamin'ny fanampiana sy ny vavaka nasandratrareo.*

Table des matières

Liste des figures

Liste des tableaux

Notation & Abréviations

3NF: Third Normal Form

API: Application Programming Interface

BAM: Business Process Monitoring

BC: Binding Component

BI: Business Intelligence

BPEL: Business Process Execution Language

BPM: Business Process Management

BPMN: Business Process Model and Notation

BPMS: Business Process Management System

COTS: Commercial Off The Shelf

CRM: Customer Relationship Management

CSV: Comma-Separated Value

DMAIC: Define, Measure, Analyze, Improve and Control

DOLAP: Dynamic OLAP.

EAI: Enterprise Application Integration

EIP: Enterprise Integration Pattern

ERP: Enterprise Resource Planning

ESB: Enterprise Service Bus

ETL: Extract, Transform, Load

EVA: Economic Value Added

FTP: File Transfert Protocol

HOLAP: Hybrid OLAP.

HTTP: HyperText Transfer Protocol

IDC: Intelligence, Design, Choice

IHM: Interface Homme Machine

JBI: Java Business Service

JCA: Java Connector Architecture

JMS: Java Message Service

JMS: Java Message Service

KDD: Knowledge Discovery in Databases

KPI: Key Performance Indicator

LDAP: Lightweight Directory Access Protocol

MDX: MultiDimensional eXpressions

MEP: Message Exchange Pattern

MOLAP: Multidimensional OLAP.

MOM: Message Oriented Middleware

ODS: Operational Data Store

OLAP: On-Line Analytical Processing

PDCA: Plan-Do-Check-Act

POJO: Plain Old Java Object

Q-BPM: Quanta BPM

QoS: Quality Of Service

RàPC: Raisonnement à Partir de Cas

RCO : Représentations des Connaissances par Objets

REST: REpresentational State Transfer

ROE: Return On Equity

ROI: Return On Investment

ROLAP: Relational OLAP

SCA: Service Component Architecture

SE: Service Engine

SI : Système d'Information

SIS : Système d'Information Stratégique

SLA : Service Level Agreement

SMO : Système de Management Opérationnel

SOA: Service Oriented Architecture

SOAP: Simple Object Access Protocol

SQL: Structured Query Language

TIC : Technologies d'Information et de Communication

UDDI: Universal Description Discovery and Integration

UML: Unified Modeling Language

WSDL: Web Services Description Language

XML: Extensible Markup Language

XMPP: eXtensible Messaging and Presence Protocol

XSLT: eXtensible Stylesheet Language Transformations

Introduction et Présentation du Problème

L'internationalisation de la concurrence et la mondialisation des marchés caractérisent le nouvel environnement économique des entreprises. En plus, l'évolution des technologies de l'information et de la communication a rapproché les distances et a éliminé les frontières, réduisant ainsi le monde en un grand marché dans lequel les informations, les capitaux et les marchandises circulent facilement et rapidement. Ces mutations apportent certainement de nouvelles perspectives, mais surtout de nouveaux défis à cause des nouvelles contraintes.

Face à ce contexte, l'ingénierie de système d'information devient de plus en plus complexe : les entreprises deviennent plus exigeantes en termes d'outils de traitement de données et de support à la prise de décision, et le rapport qualité prix restent inflexibles au choix de système.

Le concept de génie logiciel traditionnel ne permet plus répondre à ces besoins du système d'information car les besoins en matière de logiciel ne cessent de s'évoluer ; la réutilisation des codes et des API avec le concept des applications distribuées ne suffisent plus à rendre la conception de logiciel plus efficace et plus rentable.

Nous sommes donc confrontés à des divers problèmes sur la conception des outils de gestion de système d'information :

- la perpétuelle évolution de l'organisation, de son environnement et de son métier
- le délai d'étude et de la mise en place de la solution
- le coût de la réalisation et de la maintenance
- la vitesse de la mutation du secteur TIC
- la valorisation des savoir-faire de l'entreprise dans le processus de prise de décision

Ces problèmes sont liés à la maitrise de principe et de fonctionnement du système d'information, la nature de l'information et la catégorisation des traitements. Le concept traitement d'information est très vague, il commence par la génération de l'information puis la collecte, la transformation, et se termine par la génération d'une autre information. La qualité du système ne se mesure pas donc sur le mode de traitement mais plutôt par l'appréciation de la valeur de l'information obtenue.
Or, l'interprétation de l'information est relative, elle dépend du contexte et la position de l'acteur qui l'interprète. Ceci nous amène à conclure que le système d'information possède deux entités bien distinctes :

- une entité statique qui dépend entièrement de la technologie de l'information du système
- une entité dynamique qui permet de contenir les besoins ponctuels des acteurs dans le système

Comme tout système, l'ingénierie de conception de logiciel possède aussi ses propres expériences et ses savoir-faire, on en déduit qu'il y a de factorisation possible dans le mode de présentation des solutions à des divers problèmes de génie logiciel.
Le design pattern est en général le résultat des diverses expériences lors des conceptions de logiciel. La récurrente de la situation du problème et la solution trouvée ont permis d'identifier un pattern.

Avec le concept orienté objet, l'ingénierie logicielle a obtenu le gain de la réutilisation des classes et de ses dérivées ; et en ajoutant le niveau de la réutilisation offert par le design pattern, on a l'opportunité de trouver une solution fiable au problème de SI.

Notre solution, pour améliorer la conception, la réalisation et la mise en place d'un outil de gestion de système d'information est donc basée sur la réutilisation. Ce concept possède quatre niveaux d'intégration :

- composant
- processus
- service
- Cas de pilotage

Dans cette recherche, on propose la modélisation d'un noyau d'application réutilisable pour des applications entreprises en utilisant des design patterns. Cette application nous permettra de mettre en évidence les caractéristiques du SI et d'élaborer d'autre méthodologie de mise en place et de maintenance du système.

Pour parvenir à cette modélisation, on va procéder à la démarche suivante :

- définir un modèle pour présenter un système d'information
- créer un catalogue de design pattern
- trouver les métiers basic liés à l'utilisation de la technologie d'information
- quantifier les besoins dynamiques du système d'information par des processus

La technique de base offerte par cette étude est indépendante de l'organisation qui va l'implémenter. Le processus d'insertion du métier de l'organisation est une autre phase dans la mise en place du SI et il requiert d'autre analyse de besoin et d'autre développement. Le génie logiciel standard peut être toujours appliqué pour concevoir les applications relatives à cette phase d'insertion et d'adaptation.

Le fait de déployer cette solution dans des divers SI, nous mènera à un autre degré de réutilisation car le système peut sauvegarder les expériences acquises par ces organisations et d'en réduire au maximum le coût, le temps et la complexité d'une nouvelle installation.

CHAPITRE 1 : Traitement de l'information

I.1 Introduction

Le concept système d'information est devenu standard aujourd'hui pour l'ensemble des organisations privées et publiques pour organiser ses ressources afin de permettre la collecte, regroupement, classification, traitement et diffusion de ses informations sur son environnement de travail.

Cinq points spécifiques caractérisent l'apport des techniques du traitement électronique de l'information :

- la compression du temps; elle permet le recours systématique à l'automatisation des calculs et l'usage de certaines méthodes de résolution de problèmes
- la compression de l'espace (transmission de gros volumes de données entre des points très éloignés)
- l'expansion de l'information stockée
- la flexibilité de l'usage
- la connectivité

Ce qui nous amène à identifier les propriétés techniques du système d'informations et l'information elle-même afin d'en déduire le modèle de l'outil technologique que l'on va mettre en place.

I.2 Système d'information

Définition 1.1 :

L'information est ce qui nous apporte une connaissance, qui modifie notre vision du monde, qui réduit notre incertitude ; c'est un renseignement.

Définition 1.2:

Un système d'information est un réseau complexe de relations structurées où interviennent hommes, machines et procédures qui a pour but d'engendrer des flux ordonnés d'informations pertinentes provenant de différentes sources et destinées à servir de base aux décisions.

Définition 1.3 :

Le système d'information représente un élément sur lequel l'organisation peut s'appuyer ou se construire autour. Il reflète les aspects de l'activité de l'organisation.

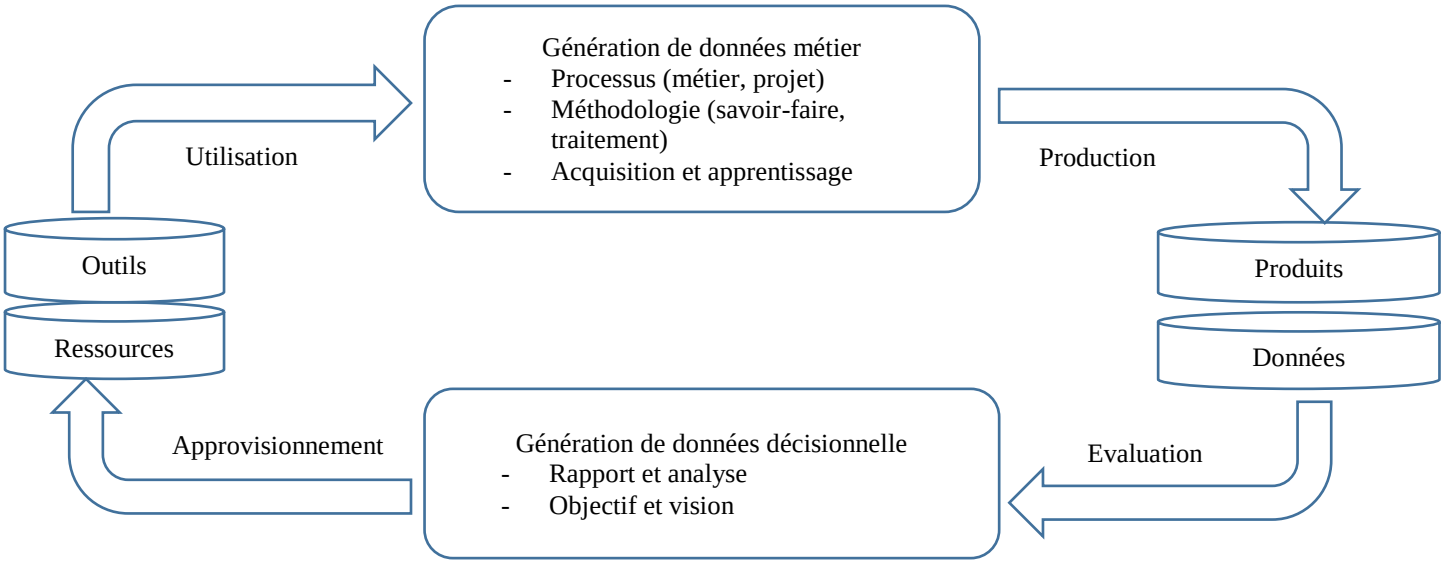


Figure 1.1 : Flux d'information dans un SI

Toute activité au sein du système génère des informations relatives à la production, à la gestion et à la prise de décision. Le système d'information est donc équivalent à un circuit à boucle fermé (Figure 1.1) :

- génération de données métiers : chaine d'action
- génération de données décisionnelles : chaine de retour

Dans ce circuit, on distingue deux aspects de l'information

- logique et/ou virtuel (donnée binaire, ressource logicielle)
- physique (ressource matérielle, produit)

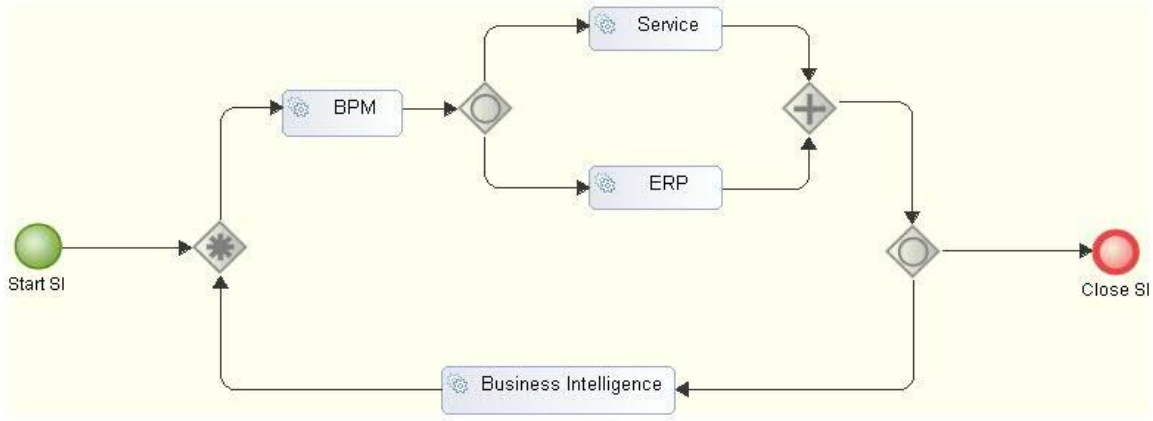


Figure 1.2 : Schéma fonctionnel d'un SI

Après une réorganisation du système, on déduire le schéma fonctionnel illustré par la figure 1.2. Le traitement de l'information passe par quatre blocs dans un système d'information :

- décomposition de l'organisation en processus métier, pilotage et projet
- collecte et transformation des données dans les services qui implémentent les métiers

- collecte et transformation des données dans les services qui gèrent les ressources de l'organisation
- calcul des indicateurs et génération de tableau de bord

L'ajustement des processus et les traitements de métier se font à partir du résultat du bloc BI. L'automatisation de cet ajustement permet de rendre le système tout entier comme étant un système intelligent.

I.2.1 Nature de l'information

I.2.1.1 Donnée et information

Définition 1.4 :

La donnée est une représentation d'un ou des symboles de ce qui est survenu ou de ce qui survient au moment même. Elle n'est pas signifiante en soi, elle ne le devient que lorsque quelqu'un en fait une interprétation, lui donne un sens. Elle devient information pour celui qui l'interprète.

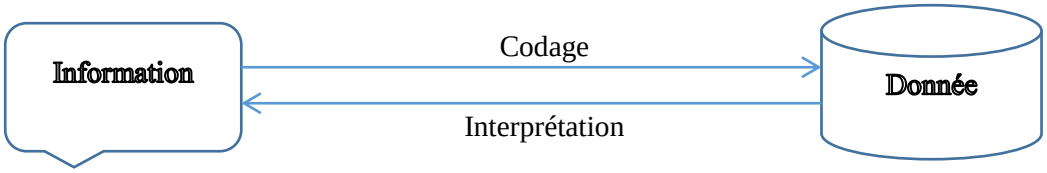


Figure 1.3 : Dualité information / données

La figure ci-dessous montre la relation entre information et donnée :

- donnée : Information sous forme conventionnelle (codes) en vue d'être traitée
- information : Une donnée avec une sémantique (un sens, une signification)

Une donnée est un élément brut qui n'a pas été interprété, c'est-à-dire mis en contexte. L'information, pour sa part, est une donnée interprétée. Elle met en relation différentes données pour obtenir un fait. La connaissance, quant à elle, est basée sur une information assimilée et utilisée pour aboutir à une action. La connaissance permet la généralisation des problèmes alors que l'information ne permet de prendre que des décisions.

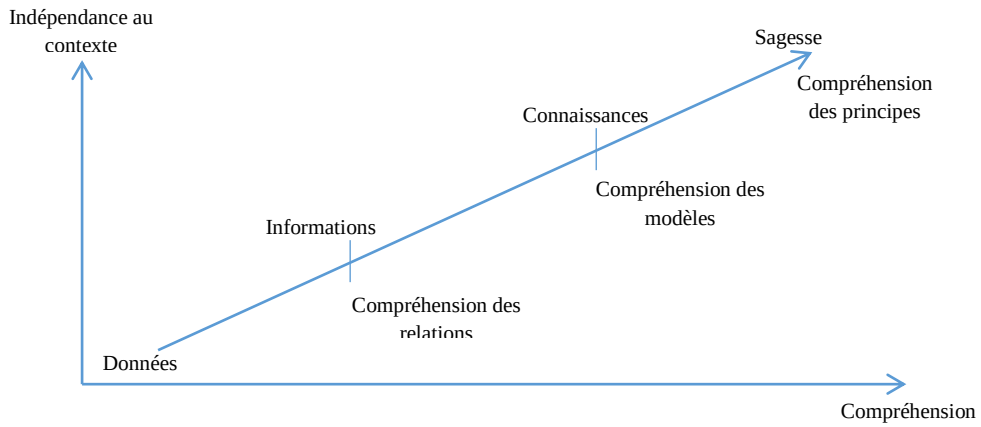


Figure 1.4 : Evolution de la compréhension

I.2.1.2 Caractéristique d'une information

L'information est une entité qui peut être définie avec trois paramètres :

- son aspect
- son état
- sa classe

Avec le temps, l'information se métamorphose au sein du SI (Figure 1.5) ; sa forme, sa catégorie et ses valeurs dépendent de la nature de l'observateur et le principe de la mesure.
La perception de l'information se transforme au sein même du système. L'information se manipule, l'expérience se simule, l'intelligence s'est artificialisée :

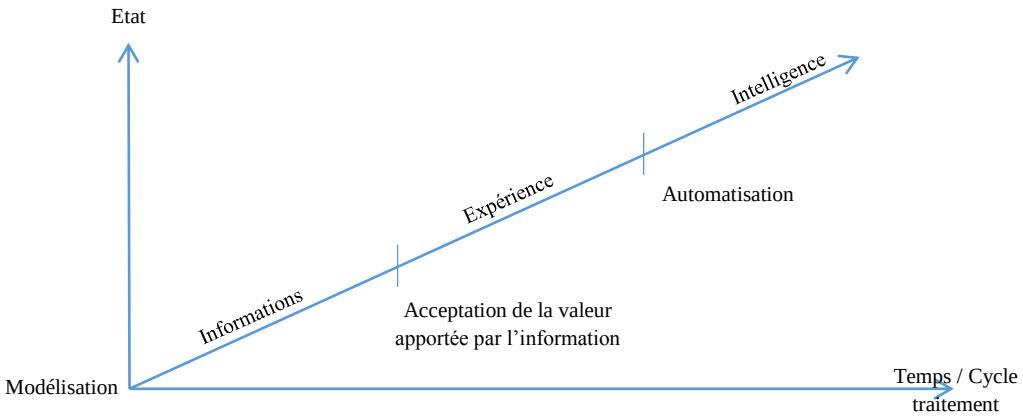


Figure 1.5 : Evolution de l'état de l'information dans un SI

Cette évolution n'est pas linéaire, il est entièrement dépendant de la dynamique du SI :

- il commence par la génération de l'information
- avec un traitement spécifique, cette information donne au SI un aspect d'expérience
- en automatisant le traitement avec cette expérience vécue, le système devient intelligent

I.2.1.3 Aspect de l'information

Cette caractéristique permet de définir la perception physique de l'information :
« Cette information a- t- elle une existence réelle ? »

En général, dans une organisation on utilise trois types de ressources :

- logicielle
- matérielle
- mixte

Ces ressources en tant que telles, sont déjà des informations et leurs utilisations au sein du système génèrent des informations à caractéristique matérielle ou virtuelle (numérique).

I.2.1.4 Etat de l'information

Elle définit la valeur de l'information au sein du système par rapport au temps et le cycle de traitement de l'information.

- si l'information est en cours d'évolution, elle devient dépendante du temps, du cycle de traitement, et sa valeur est ponctuelle.

- si l'information est acceptée selon l'expérience, sa valeur devient indépendante du temps mais la reproduction de l'information passe encore par le cycle de traitement

- si la production de l'information est automatisée, la valeur de l'information est devenue indépendante du temps et du traitement et elle devient à l'état intelligent.

I.2.1.5 Classe de l'information

La classification (axe ou dimension d'analyse) de l'information est essentiellement dépendante du métier de l'organisation. Elle a pour objectif de déterminer les actifs informationnels pour l'organisation en fonction des trois objectifs :

- la disponibilité
- l'intégrité
- la confidentialité

I.2.2 Rôle du Système d'information

Comme le système d'information est un moteur de collecte, de transformation et d'évaluation des informations, il offre un moyen pour l'organisation de :

- structurer les traitements et les ressources
- valoriser les acquis
- identifier les performances et les indicateurs
- définir les objectifs

I.2.2.1 Traitement de l'information

Il est le fondement de l'existence du système d'information. Il peut être représenté par l'association de quatre fonctions principales :

- modélisation
- collecte/stockage
- transformation
- communication

Le traitement de l'information est la partie la plus dynamique d'un SI. La capacité d'un système de supporter plusieurs versions de traitement opérationnel pour un même métier constitue l'indicateur de flexibilité.

I.2.2.2 Relation entre organigramme et le système

Le système interagit avec l'organisation d'une manière transversale pour assurer son rôle. D'après la figure 1.6, les éléments qui constituent l'organisation sont définis comme étant acteurs, processus ou ressources :

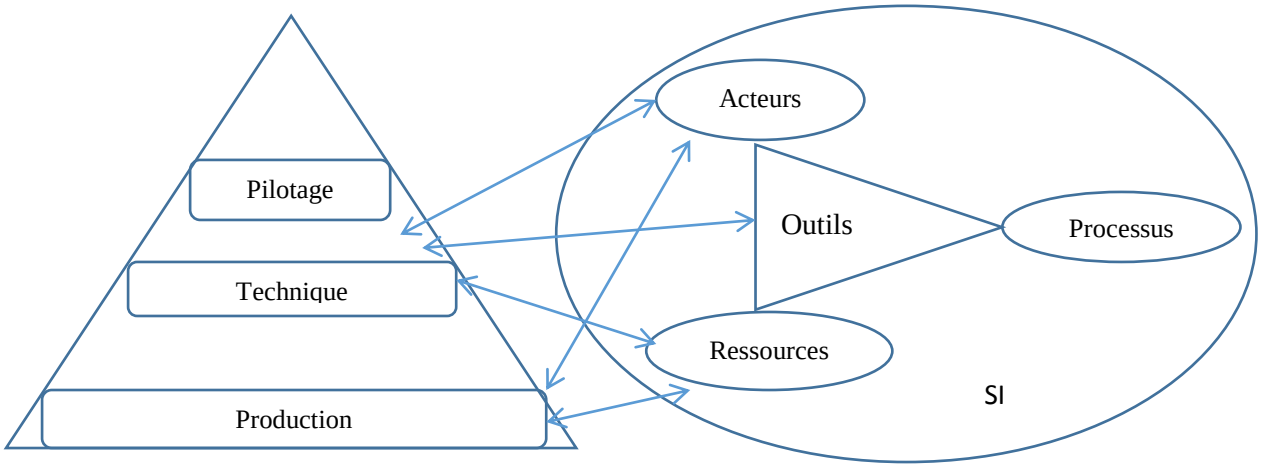


Figure 1.6 : Relation entre l'organisation et son SI

Une organisation possède en général trois organes hiérarchisés : pilotage, technique et production. La structure ou organigramme de cette organisation est définie selon la responsabilité de chaque membre. Tandis que les trois entités de SI sont liées entre elles selon le métier de l'organisation défini dans le processus et l'outil.

L'effet de l'organigramme de l'organisation sur le SI est un indicateur du degré de liberté de son système d'information, le processus qui gère le profil des entités qui réalisent les tâches permet de calculer cet indicateur.

a. *Relation entre système opérant et le SI*

Le système opérant constitue la base de l'organisation. Elle a pour responsabilité d'assurer la production et la réalisation de l'objectif de l'organisation. Le traitement des informations dans ce système est basé essentiellement sur :

- collecte/stockage : formulaire, gestion des entrées/sorties, gestion de client/fournisseur, etc.
- la communication : formation, affichage, etc.

b. *Relation entre système technique et le SI*

La partie technique de l'organisation assure la mise en place et la gestion du système d'information. Le traitement dans cet organe est donc basé sur :

- modélisation : processus, données, cube, etc.
- transformation : service, application, rapport, etc.

c. Relation entre système de pilotage et le SI

Ce système est le plus gourmand en information, il utilise le tableau de bord offert par le système d'information pour prendre de décision :

- diffuser au système technique les mesures correctives à tenir compte dans le SI
- diffuser au système opérant le résultat de l'organisation et le nouvel objectif à atteindre

I.2.3 KPI

Ce concept offre au SI le moyen de se doter d'un organe de sens pour mesurer les paramètres susceptibles d'affecter le système et toute l'organisation.

Définition 1.5 :

Un indicateur est une grandeur spécifique observable et mesurable qui peut servir à montrer les changements obtenus ou les progrès accomplis en vue de la réalisation d'un effet spécifique.

I.2.3.1 Caractéristique d'un indicateur

Lors de la modélisation d'un SI, il est important de choisir de bon indicateur à implémenter dans le système pour s'assurer sa survie. Ils doivent avoir les propriétés suivantes :

- valide : mesure exacte d'un comportement, d'une pratique ou d'une tâche qui sont l'extrant ou l'effet attendus de l'intervention
- fiable : mesurable de manière constante dans le temps et de la même façon par différents observateurs
- précis : défini en termes clairs du point de vue opérationnel
- mesurable : quantifiable au moyen des outils et méthodes disponibles
- opportun : fournir une mesure à des intervalles temporels pertinents et appropriés compte tenu des buts et activités de l'organisation
- important : lié à l'objectif de l'organisation

I.2.3.2 Classification des indicateurs

La première classification des indicateurs est définie selon l'interprétation de son impact sur le système:

- le domaine touché : production, vente, ressource humaine, matière première…
- le résultat obtenu (forme et valeur) : pourcentage, diagramme, tableau
- l'étendue dans le temps : l'analyse de la valeur peut se faire avec un paramètre temps

Le deuxième critère de classification est le type d'information transmise (résultat) :

- alerte : ce type d'indicateur de type tout ou rien, signale un état anormal du système sous contrôle nécessitant une action, immédiate ou non. Un franchissement de seuil critique par exemple entre dans cette catégorie d'indicateur
- équilibration : l'indicateur est lié aux objectifs. Il informe sur l'état du système sous contrôle en relation avec les objectifs suivis.
- Anticipation : il offre une projection de la valeur de l'indicateur dans le futur.

I.2.3.3 Méthode d'évaluation de l'indicateur

Il existe deux approches d'évaluation :

- qualitative
- quantitative

L'évaluation qualitative s'intéresse à définir des propriétés structurelles et comportementales.

- absence de blocage
- existence d'une solution
- respect de cahier de charge
- respect de norme
- etc.

L'évaluation quantitative consiste à calculer les critères clés du système.

- débit
- nombre
- volume
- durée
- coût
- taux
- etc.

I.2.3.4 Indicateur clé de performance

Définition 1.6 :

Un indicateur de performance KPI est une mesure ou un ensemble de mesures braquées sur un aspect critique de la performance globale de l'organisation.

Un KPI donne l'assurance que le système est sous contrôle (commandable), il permet à l'organisation de définir l'orientation de ces décisions.

a. Mesure de la performance

La notion de performance est difficile à appréhender car elle peut être mesurée par de très nombreux indicateurs et être interprétée sous différents angles.

Mais en général, la performance est le rapport entre la ressource et la technique mise en cause, le résultat obtenu et l'objectif défini par le système. L'interprétation de cette valeur peut prendre plusieurs formes.

Par exemple :

- efficacité : l'objectif défini est- il atteint
- efficience : l'utilisation des ressources est- elle optimale
- pertinence : le choix de l'objectif est- il convenable et rentable par rapport à la situation

En d'autre terme, la mesure de la performance est une partie de processus de pilotage du SI (Figure 1.7), il permet de transformer les indices fournis par tous les indicateurs en un ensemble des indicateurs clés (KPI) que l'on nomme tableau de bord.

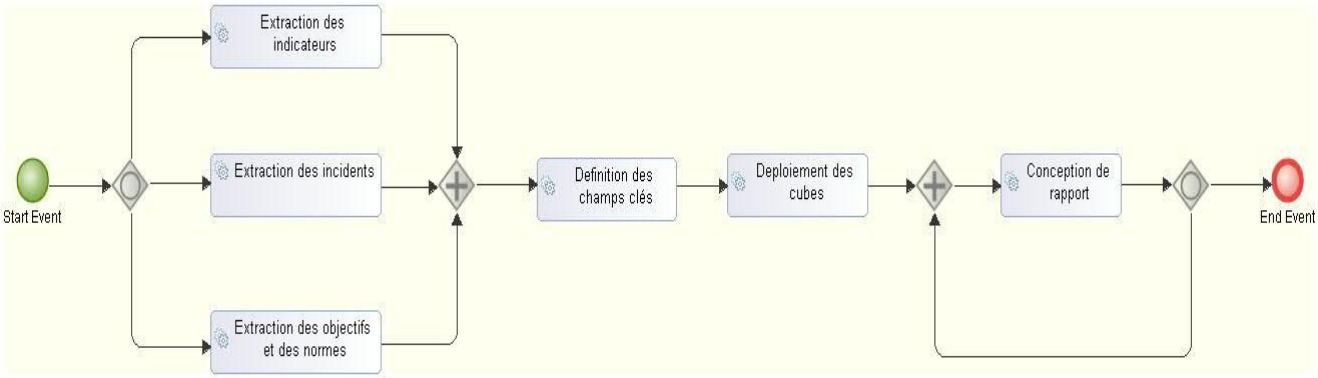


Figure 1.7 : Schéma fonctionnel de la génération de tableau de bord

Ce processus utilise cinq types de sources d'information :

- le système opérant et technique (indicateurs)
- le système de gestion des incidents
- le système de pilotage (objectifs et champ clé ou dimension de cube)
- le système de gestion de connaissance de l'organisation (Historique de TDB, processus, décisions, etc.. de l'organisation)
- organisation internationale (norme)

b. Objectif de l'organisation

Selon sa valeur morale, son métier, ses connaissances et sa capacité, une organisation définit un ensemble d'objectif dans un intervalle de temps donnés. Cet objectif peut être de nature qualitative ou quantitative.

En général la valeur de chaque objectif est déduite de la dernière information du tableau de bord. Un objectif est une information spécifique qui devrait avoir les propriétés suivantes :

- borné : limité dans le temps
- mesurable
- accessible : quels moyens, quelles contraintes, quels risques
- réaliste : quelle méthode d'accès
- cohérente : s'adapte avec les autres objectifs

c. Incident

Un incident est un événement interne ou externe qui n'est pas modélisé ou traité lors de la mise en place du SI. On distingue trois types d'incidents :

- erreur (conception, organisation, humaine, matériel, communication) : cet incident provoque un arrêt d'une partie ou la totalité du système. On ne peut pas relancer la partie défaillante tant qu'on n'arrive pas à corriger le problème
- exception (conception, organisation, humaine, matériel, communication) : événement qui perturbe le bon fonctionnement du système mais qui ne provoque pas un arrêt immédiat. Si on ne traite pas une exception, il devient une erreur
- événement (marché, catastrophe naturel, crise économique et politique, etc.) : incident qui est totalement indépendant du système et que l'on arrive jamais à maitriser. Ce type d'événement peut être négatif ou positif au bon fonctionnement du système

Le système d'information doit considérer le traitement des indicateurs liés à ces incidents pour améliorer sa robustesse.

I.2.4 Business Intelligence

Le concept Business Intelligence relève de la stratégie de l'organisation, elle a comme objectif d'assister à la réalisation du processus de décision.
En utilisant le flux défini par le figure 1.8, le processus BI permet au SI de convertir les informations issues de processus business ou bloc opérationnel à d'autre forme d'information plus avancée : connaissance et sagesse.

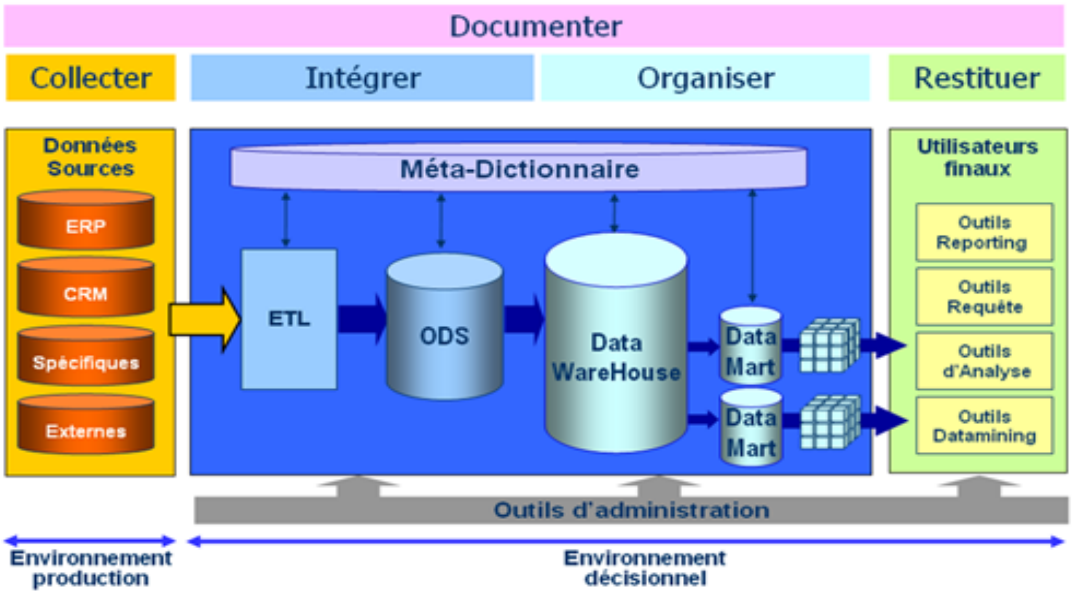


Figure 1.8 : Flux informationnel lié au processus BI

Tableau 1.1 : Résumé des traitements dans un processus BI

Module	**Description**
ETL	Processus d'agrégation et de transformation pour remplir une base de données. La transformation a pour objectif de rendre la structure et le contenu plus lisible du côté métier.
ODS	Une base de données conçue pour centraliser les données issues de sources hétérogènes afin de faciliter les opérations d'analyse et de *reporting*. L'intégration de ces données implique souvent une purge des informations redondantes
Data warehouse	Une structure de données utilisée par le processus ETL pour sauvegarder les données après le traitement.
Data Mart	Sous ensemble de *data warehouse* dédié à une entité de traitement.
OLAP	Un cube OLAP est une représentation abstraite d'informations multidimensionnelles. Opérations OLAP : - pivoter (*pivot*, *swap*) : changer de dimension ou axe. - naviguer vers le bas (*drill-down*) : descendre dans la hiérarchie de la dimension (détail). - naviguer vers le haut (*drill-up*, *roll-up*) : remonter dans la hiérarchie de la dimension (résumé). - naviguer latéralement (*drill-across*) : permet de passer d'un membre de dimension à un autre. Langage utilisé par les requêtes : MDX
Data mining ou KDD	Analyse axé sur : - reconnaissance - apprentissage - prédiction
Report	Présentation des données et des statistiques sous plusieurs formes (tableau, liste, graphe)

I.2.4.1 Axe d'analyse

Cet axe permet de définir la dimension des données à traiter dans le module OLAP.
La définition de nombre et des propriétés des axes d'analyse dépend de :

- la liste des indicateurs (observable)
- la liste des objectifs (comparable)

I.2.4.2 Donnée multidimensionnelle

Il existe principalement deux types de base de modèle dimensionnel :

- schéma en étoile
- schéma en flocons de neige

a. Schéma en étoile

Le schéma en étoile est une façon de représenter un modèle dimensionnel au sein d'une base de données relationnelle. Un schéma en étoile (Figure 1.9) se constitue d'une table de faits associée à un ensemble de tables de dimensions. Ces dernières sont toujours directement reliées à la table de faits.

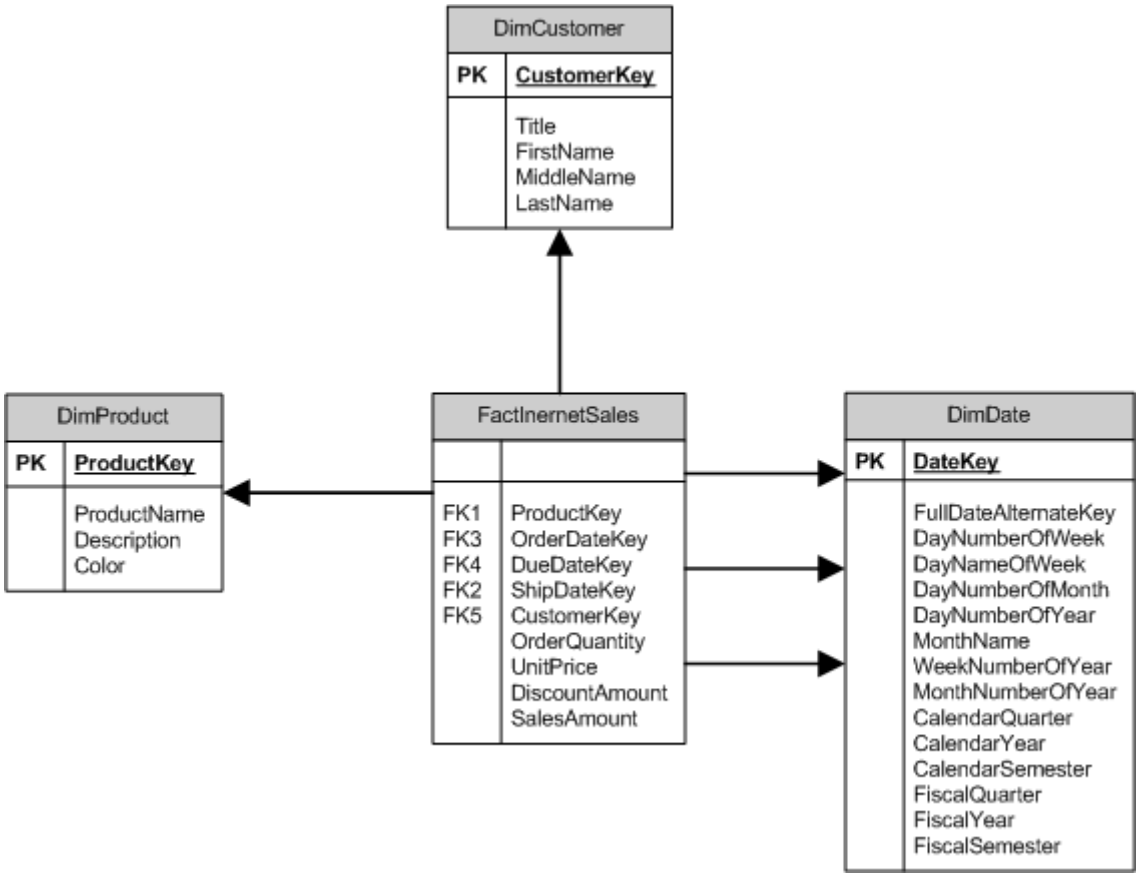


Figure 1.9 : Schéma en étoile des dimensions

b. Schéma en flocons de neige

Un schéma en flocon est une représentation normalisée (3NF) d'une unique table de dimension. Cette table, qui peut parfois contenir un grand nombre d'attributs, est scindée en plusieurs entités. Il est basé du schéma en étoile mais la dimension est repartie en plusieurs entités. Les figures suivantes montrent des exemples d'un schéma en étoile et en flocon :

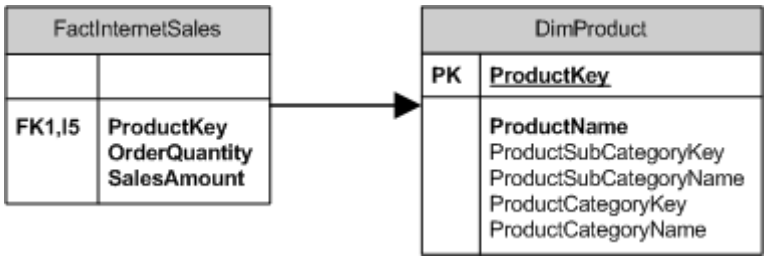


Figure 1.10 : Dimension Produit modélisée sous la forme d'un schéma en étoile

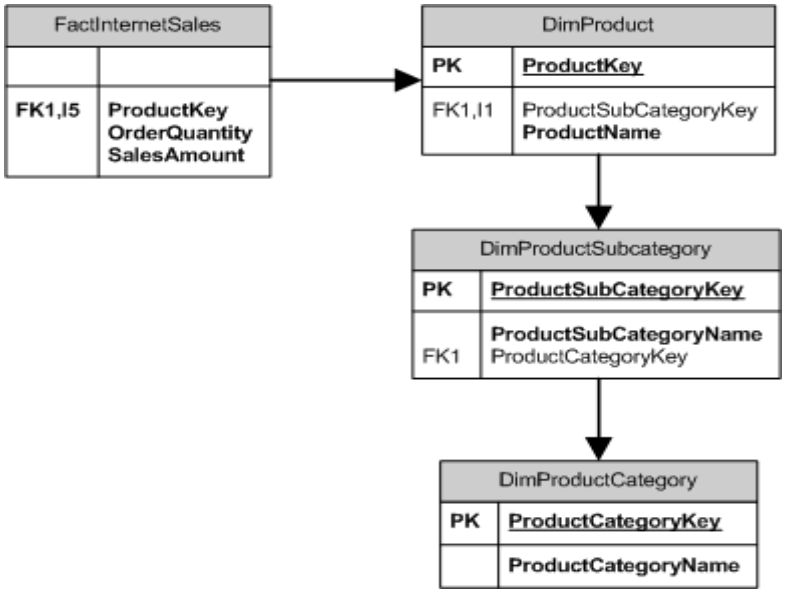


Figure 1.11 : Dimension Produit modélisée sous la forme d'un schéma en flocon

I.2.4.3 Processus décisionnel

Toute décision est une prise de risque, le rôle du BI est de limiter au maximum le risque et d'apporter de valeur ajoutée à la décision.
Les décisions dans l'organisation sont liées aux activités qui s'y déroulent et en général on ne peut pas prendre de décision si on n'a pas encore défini les objectifs.

a. *Type de décision*

Il y a trois niveaux de management dans l'organisation :

- régulation (contrôle opérationnel) : activités concernant principalement des rythmes inférieurs à un mois et conduit à des décisions de portée limitée (champ local et limité)
- pilotage (Planification et le contrôle managériaux) : activités conduisant à des décisions dont les conséquences sont à moyen terme
- planification Stratégique : activités engendrant des décisions majeures dont les conséquences sont à long terme. La décision est de portée globale (champ global)

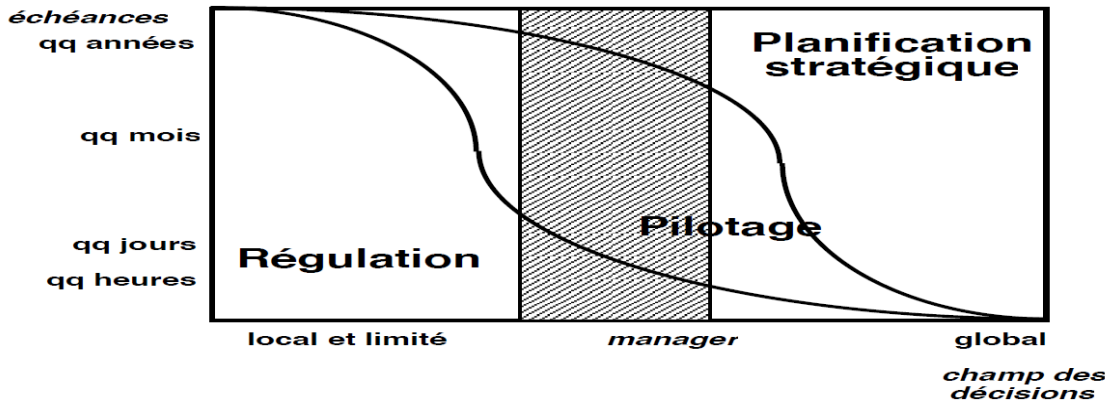


Figure 1.12 : Champs des décisions et Management de l'organisation

De façon générale, pour chaque niveau managérial, la répartition des types de décision concernés est définie par la figure suivante :

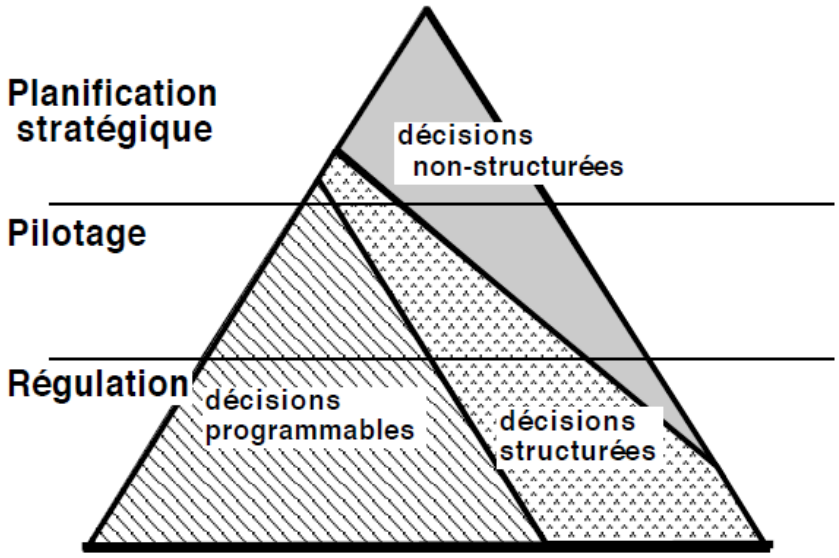


Figure 1.13 : Type de décision et Management de l'organisation

Cette configuration mène à proposer deux types de décisions :

- décisions programmables : décisions répétitives et routinières, et une procédure a été définie pour les effectuer, évitant ainsi d'avoir à les reconsidérer chaque fois qu'elles se présentent
- décisions non programmables : il n'a pas été possible de définir une procédure spécifique pour les effectuer; soit du fait qu'elles sont nouvelles, non structurées, inhabituelles, …

b. Choix de décision rationnelle

Définition 1.7 :

Choix rationnel : une décision cohérente et générant de la valeur dans la limite des contraintes données.

Pour arriver à la sélection d'une décision, on devrait passer par deux étapes (modèle IDC de la figure 1.14) :

- phase d'intelligence (investigation) :
  - processus de formulation du problème décisionnel
  - confrontation entre situation perçue et situation voulue : perception de dissonance
  - définition de valeurs, d'objectifs, de frontières, d'actions (solutions) possibles
- phase de modélisation (conception) :
  - élaboration de modèle, d'actions possibles, de plans d'action intentionnels, de stratégies
  - possibles permettant la résolution du problème
  - décrire/prévoir l'état du système si on lui applique une action possible
- phase de choix (sélection) :
  - évaluation, comparaison, classement des actions possibles
  - choix d'une action parmi ces actions possibles
  - si aucune action n'est satisfaisante, reconsidérer les phases antérieures

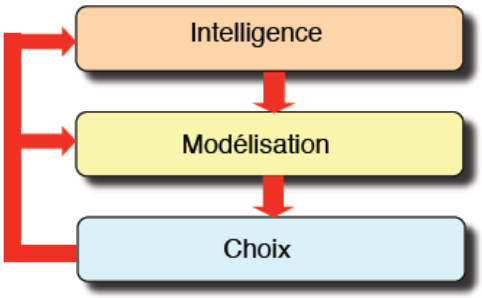


Figure 1.14 : Modèle de prise de décision IDC

c. Choix de décision créative

Définition 1.8 :

La créativité : capacité à développer des idées inédites, originales et utiles face aux problèmes à résoudre ou aux opportunités à saisir.

La créativité est basée sur trois éléments :

- expertise : capacités, connaissances, compétences ou expertises nécessaires à la créativité.
- ingéniosité : tous les aspects de la personnalité associés à la créativité, la capacité à employer des analogies ainsi que le don de voir le connu sous un nouveau jour.
- motivation : le souhait de travailler sur une tâche parce qu'elle suscite un intérêt, suppose une implication, apporte une satisfaction ou constitue un défi personnel

I.2.4.4 Gestion de risque

Toute décision est liée à des risques. Dans le processus décisionnel, le système essaie de définir la liste des risques et leurs probabilités d'apparition :

- si le risque est identifié, il devrait être accompagné d'un processus de surveillance
- si le risque est non identifié, il peut devenir un incident

Le processus décisionnel devrait inclure les étapes dans le tableau 1.2 pour annuler les effets de risques après la prise de décision.

Tableau 1.2 : Liste des étapes de traitement de risque

Etape	**Description**
L'identification des risques	Cette phase consiste à identifier et à structurer les risques opérationnels qui seront présentés dans un support de cartographie des risques
L'évaluation des risques	A partir de la cartographie des risques opérationnels établis, il convient d'effectuer une évaluation des risques identifiés - gravité : c'est l'impact maximum de l'exposition réelle ou potentielle aux situations de risque. - détection/Gestion : il s'agit de la capacité de l'entreprise à identifier et à réagir face aux évènements de risques. - occurrence : c'est la probabilité d'apparition des situations de risque.
La surveillance des risques	Cette phase correspond à la mise en place d'un dispositif de suivi et de contrôle du profil de risque de l'entreprise.

I.3 Dimension du système d'information

Définition 1.9 :

Le SI est un système d'interprétation d'un ensemble d'acteurs sociaux qui mémorisent et transforment des représentations via des technologies et des modes opératoires.

Selon cette définition, le système d'information se caractérise par trois dimensions :

- organisationnelle (acteurs)
- informationnelle
- technologique

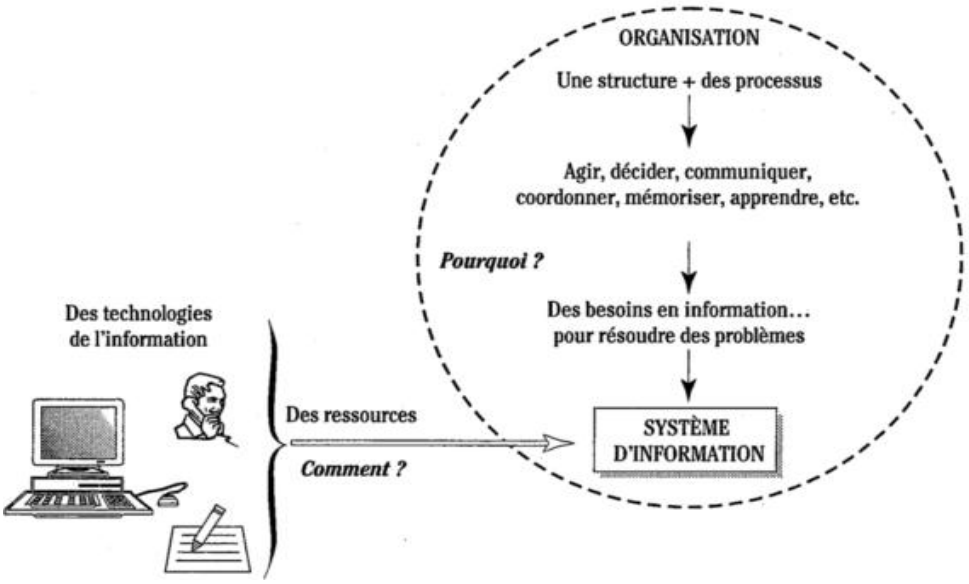


Figure 1.15 : Description de besoin du SI

Selon la figure 1.15, les systèmes d'information sont conçus pour aider l'organisation à prendre des décisions.
Pour représenter ces besoins de l'organisation, on a besoin d'identifier la dimension informationnelle de l'organisation selon les catégories suivantes :

- opérationnel
- ressources
- organisation

La dimension du SI est définie à l'aide des graphes qui relient chaque dimension listé dans chaque catégorie d'informations. Cette information permet de structurer et d'analyser l'effet de propagation d'un changement au sein d'une catégorie vers l'ensemble du système.

I.4 Outil de gestion du SI

L'objectif principal d'un outil de gestion de système d'information consiste à restituer l'information à la personne concernée sous une forme appropriée et au moment opportun. Il est estimé que près de 60% des systèmes mis en place dans le secteur privé ont échoué. Par « échoué », on entend :

- le système n'est pas utilisé par les utilisateurs supposés ne se le sont pas approprié
- le système n'est pas pertinent pour répondre aux préoccupations des utilisateurs qui ont donc recours à d'autres ressources
- le système ne contient pas l'information recherchée
- le projet de système, trop ambitieux, n'est pas arrivé à terme

De ce fait, les besoins principaux que l'on doit prendre en compte sont :

- simplicité de l'intégration
- évolutif (le traitement du métier ne dépend que des décisions des utilisateurs)
- Paramétrable en entrée et en sortie

I.4.1 Modèle d'un système d'information

En tenant compte du schéma fonctionnel d'un SI (Figure 1.1), le processus de gestion des incidents et des risques et le principe de génération de décision, on peut détailler le modèle de traitement d'un système d'information comme suit :

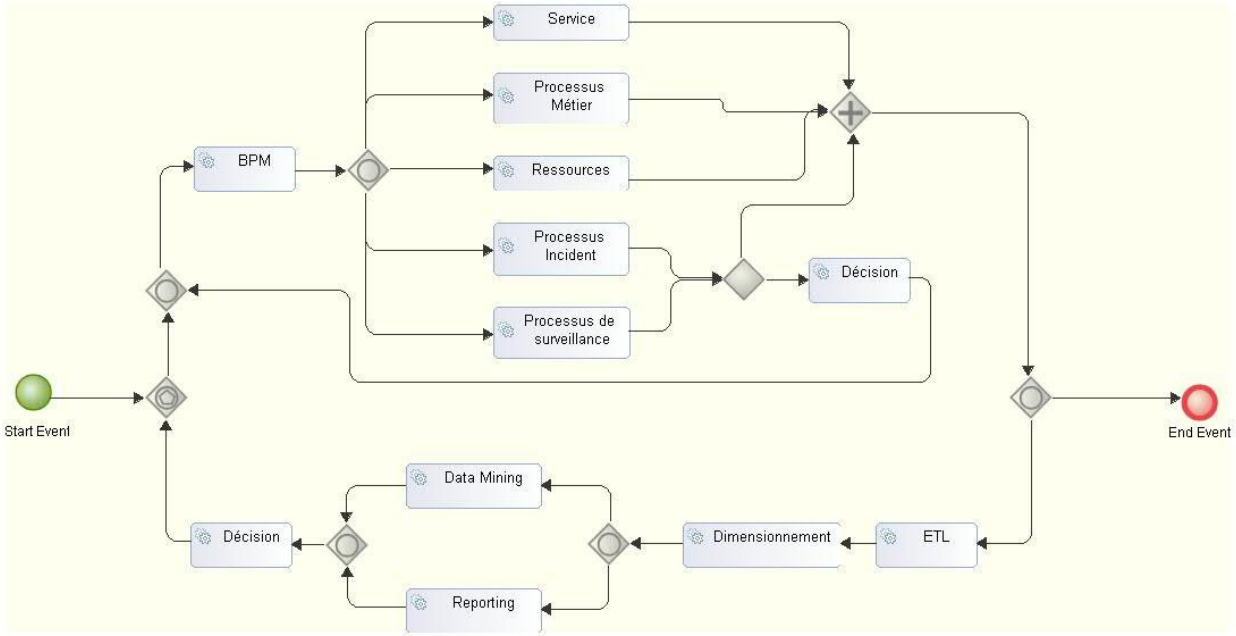


Figure 1.16 : Modèle d'un SI

Le type de décision mis en cause dans ce modèle concerne uniquement les SMO car il impacte directement la structure des traitements au niveau opérationnel.

Avec cette structure, le système est techniquement constitué :

- de gestionnaire de processus
- des applications basées sur des processus (métiers, incident, surveillance de risque)
- des applications clients / serveurs (réutilisable et modulaire)
- des applications de gestion des ressources (information dont la nature est statique)
- des matériels de communication et d'informatiques

La réalisation d'un tel modèle passe par plusieurs niveaux de conception. Certains blocs (Processus, Services, Report, ETL, Data mining) sont flexibles et d'autres sont rigides (BPM, Décision, Dimensionnement)
La conception commence par l'étude de l'existant et l'aspect technique de l'organisation. Cette étude permet de modéliser les traitements et les processus initiaux.

I.4.2 Contrainte de l'existant

La modélisation des données est un facteur clé de succès ou d'échec dans l'évolution des organisations. La phase d'étude de l'existant et des besoins est une phase essentielle à l'élaboration et la mise en place du système d'information. Elle doit aboutir à des spécifications générales qui décrivent les données manipulées, et les traitements à effectuer sur ces données.

Cette analyse permet de modéliser les besoins opérationnels du système :

- métier et activités de l'organisation
- ressources (humaine, matériels, documents, etc.)
- organigramme
- procédés
- produits
- clients
- fournisseurs
- processus

I.4.3 Solution orientée métier

Avec des différentes méthodes d'analyse standard du génie logiciel, le centre de toute la préoccupation est le métier de l'organisation et ses objectifs.
Cette démarche est appliquée à l'état de l'organisation à un moment donné par la séquence de traitements défini dans la figure 1.17 :

- définition des objectifs à atteindre
- décomposition en sous système
- identification des acteurs
- schématisation des interactions internes et externes
- modélisation des traitements
- modélisation des données
- planification et mise place

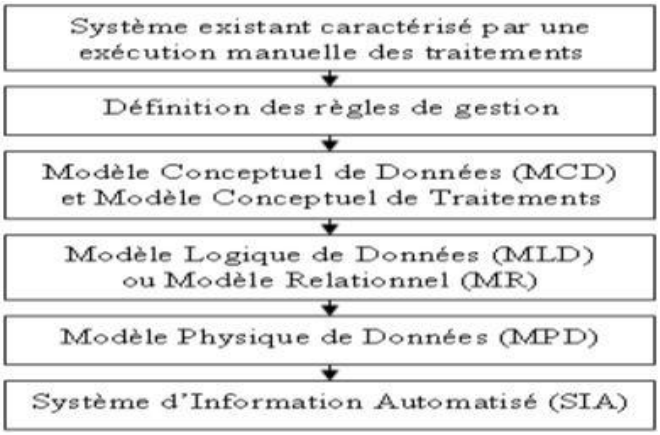


Figure 1.17 : Démarche de conception d'un SI avec la méthode d'analyse MERISE

I.5 Conclusion

L'ingénierie des SI reste un domaine complexe qui partant de l'analyse des besoins d'une application, aboutit à une solution logicielle fiable dans un système technologique cible choisi. Actuellement, la pratique industrielle de l'ingénierie des SI dans les entreprises industrielles rencontre diverses difficultés techniques, méthodologiques, organisationnelles et humaines.
Des difficultés qui constituent actuellement des goulots dans le processus de développement, engendrant des coûts considérables et menant dans beaucoup de cas à l'échec des projets SI. Il s'agit notamment :

- de la lenteur du développement des SI qui engendre des coûts considérables et mène parfois à la livraison de systèmes déjà obsolètes,
- de la nature statique de modélisation des données et des traitements,
- de l'insuffisance de l'analyse de l'impact des ressources technologiques dans la démarche,
- de l'insuffisance de la mise en valeur de la boucle de retour dans la conception. La solution est basée essentiellement au bloc opérationnel

Le SI est un moteur qui traite des informations pour produire d'autres informations. Cette boucle de traitement est dynamique selon le type de commande engendré par les décisions de l'organe de pilotage.

C'est sur ces hypothèses que se base donc la définition de notre problématique. Ainsi le travail de cette thèse a pour objectif de mettre en place un cadre méthodologique pour l'ingénierie des SI en abordant trois aspects :

- la spécification de différents blocs qui couvrent la complexité du SI (représentation des processus, des services de traitement des métiers, etc.) et favorisent l'échange d'information entre acteurs;
- la capitalisation et la réutilisation de savoir et de savoir-faire expérimentés et mis en œuvre dans des projets antérieurs. Cette démarche consiste à modéliser les patterns et les expériences du SI ;
- la modélisation d'un système ouvert permet de supporter l'évolution des processus au sein de l'organisation tout en gardant les gains des patterns au niveau composant et au niveau SI

CHAPITRE 2 : Design Pattern

II.1 Introduction

L'idée sur le design pattern a été introduite par Alexander dans le domaine de l'architecture et formalisée dans le domaine de génie logiciel par la publication du livre « *Design Patterns - Elements of Reusable Object-Oriented Software* » en 1995. Le fait que les patrons capitalisent un savoir-faire consensuel sur un domaine donné, ils se sont développés au sein de la communauté du génie logiciel et en particulier parmi les personnes s'intéressant aux approches orientées objet et à la réutilisation.

Dans le domaine de l'ingénierie logicielle, il est primordial la conception de l'architecture du logiciel, c'est-à-dire sa décomposition en sous-systèmes, modules et composantes individuelles qui la forment. Représenter précisément les éléments structuraux et comportementaux de cette architecture à l'aide de notations normalisées.

Cette conception nécessite la maitrise de chaque élément (entité logiciel) qui le constitue et ses interactions. Les patterns sont des solutions et des outils fiables pour nous aider à choisir les composants du SI ainsi que la modalité qui gère sa structure.

II.2 Design Pattern et l'expérience

Définition 2.1:

"*Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*".

Chaque patron décrit un problème qui se produit encore et encore dans notre environnement, et ensuite décrit le cœur d'une solution à ce problème de manière à ce qu'on puisse utiliser cette solution plus d'un million de fois, sans jamais le faire deux fois exactement de la même manière.

Définition 2.2 :

La réutilisation se définit comme une approche de développement de systèmes, qui propose notamment de construire un système nouveau à partir de composants logiciels sur étagère (*COTS, Commercial Off The Shelf*). Elle s'oppose ainsi aux approches traditionnelles de développement, dans lesquelles la construction d'un nouveau système part de rien (*from scratch*) et nécessite de réinventer de grandes parties d'applications à chaque fois.

Les design patterns permettent d'apporter une solution éprouvée, efficace et réutilisable, il est le résultat d'une meilleure compréhension du design complet du problème. Ils sont donc le fruit d'une expérience : un patron décrit une situation fréquente et une réponse éprouvée à cette situation.

II.2.1 Savoir faire

La notion sous-jacente aux patrons de conception est aussi connexe à la notion de savoir-faire. Elle englobe en particulier l'idée d'un plus haut niveau de compréhension élaboré à partir d'interactions entre composants primaires.

Pour illustrer cette définition, on peut comparer l'apprentissage de la conception de logiciels et l'apprentissage du jeu d'échecs, où trois niveaux de maîtrise sont bien identifiés :

Tableau 2.1 : Niveau d'apprentissage de jeux d'échecs

Niveau	**Jeux d'échecs**	**Programmation**
1 : Apprendre les règles de base :	Apprendre le nom des pièces, les mouvements légaux, l'orientation et la géométrie de l'échiquier, etc.	Apprendre des algorithmes de base, des structures de données et des langages de programmation
2 : Apprendre les principes	Apprendre la valeur relative de certaines pièces (comme la reine), la valeur stratégique des cases centrales de l'échiquier	Apprendre la connaissance des principes d'organisation du logiciel (modularité, orientation objet, généricité, etc.).
3 : Apprendre les motifs récurrents	Pour devenir un maître d'échecs, il faut en effet étudier le jeu des autres maîtres : les parties d'échecs disputées par des grands maîtres contiennent en effet des patrons qui devraient être compris, mémorisés, et appliqués de manière répétitive	Les patterns s'acquièrent en étudiant les applications déjà existantes

II.2.2 Pattern prouvé

Un pattern est un ensemble de variables, d'événements et d'invariants qui vont permettre de modéliser une notion que l'on veut voir figurer dans le système que l'on construit (système cible).

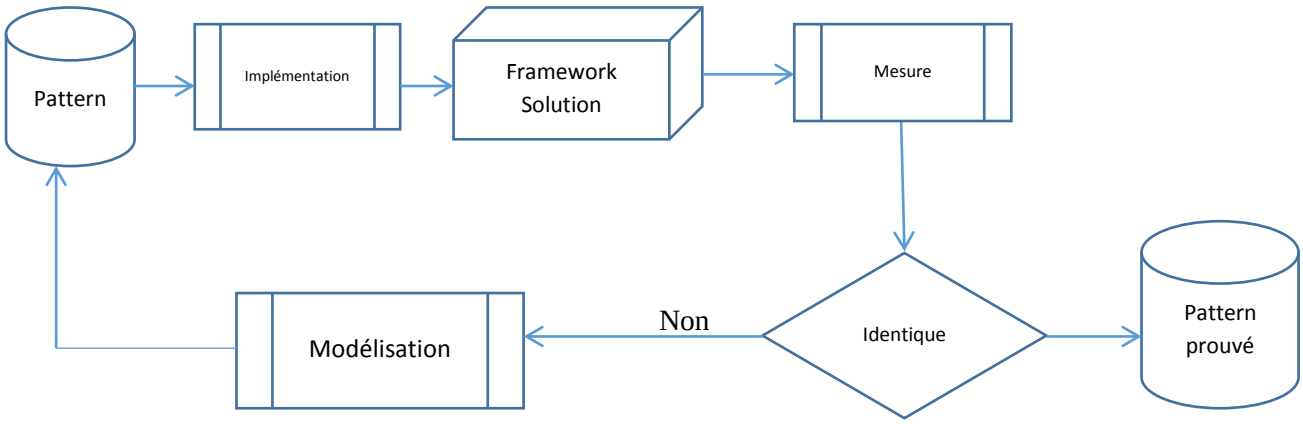


Figure 2.1 : Procédure d'approbation de pattern

En utilisant le processus défini par la figure 2.1, un pattern est prouvé lorsque les événements décrits dans ce pattern vérifient les propriétés invariantes exprimées sur le système cible quels que soient les nombres des instanciations.

II.2.3 Pattern et UML

La structure et la représentation font partie des propriétés d'un pattern. UML en tant que langage de modélisation nous offre le moyen standard de présenter les participants dans la solution ainsi que leurs interactions.

UML permet de présenter un modèle sous plusieurs diagrammes mais en général pour les patterns, les diagrammes de classe, de séquence et de cas d'utilisation sont les plus utilisés.

La description de la solution finale en termes de classe et d'interaction (diagramme de classe et séquence) est la première préoccupation des utilisateurs de pattern pendant son implémentation. Tandis que le cas d'utilisation est utile pour choisir lequel est intéressant pendant la capture de besoin.

II.3 Pattern de Gang of Four

Les 23 patterns définis par Gamma et ses équipes restent une référence pour toute forme de pattern. La plupart des Framework implémentent les solutions définies dans ces patterns.
Ils sont groupés en 3 catégories :

- création :
    - description de la manière dont un objet ou un ensemble d'objets peuvent être créés, initialisés, et configurés
    - isolation du code relatif à la création, à l'initialisation afin de rendre l'application indépendante de ces aspects
- structure
    - description de la manière dont doivent être connectés des objets de l'application afin de rendre ces connections indépendantes des évolutions futures de l'application
    - découplage de l'interface et de l'implémentation de classes et d'objets
- comportement.
    - description de comportements d'interaction entre objets
    - gestion des interactions dynamiques entre des classes et des objets

Tableau 2.2 : Liste des Pattern de Gang of Four

Catégorie	**Pattern**	**Motivation**
Création	Abstract factory	Fournir une interface pour la création de famille d'objets sans spécifier les classes concrètes
	Builder	Séparer la construction d'un objet complexe de sa représentation. Le même procédé de construction peut donc créer différentes représentations
	Factory Method	Définir une interface pour la création d'objet, en laissant les sous classes décider quelle classe instancier. (Laisser le travail d'instanciation aux sous classes)
	Prototype	Spécifier le type d'objet à créer en utilisant une instance d'un prototype, et créer de nouveaux objets en copiant ce prototype

	Singleton	S'assurer qu'une classe n'ait qu'une instance et lui fournir un point d'accès global
Structure	Adapter	Convertir l'interface d'une classe dans une autre interface compatible avec d'autres clients
	Bridge	Séparer une abstraction de son implémentation dans le but que chacune puisse être modifiée indépendamment
	Composite	Composer des objets en structure d'arbre hiérarchisé
	Decorator	Ajouter dynamiquement des responsabilités à un objet
	Facade	Proposer une interface commune à un ensemble d'interfaces d'un sous-système (améliorer l'utilisation d'un sous-système).
	Flyweight	Utiliser le partage afin de supporter un large panel d'objets bas niveau
	Proxy	Proposer un clone ou un point d'entrée à un objet pour contrôler son accès
Comportement	Chain of responsibility	Permettre d'associer l'émetteur et le récepteur d'une requête en donnant à plus d'un objet la chance de manipuler la requête (enchaîner des objets et parcourir cette chaîne avec la requête jusqu'à trouver quelqu'un capable de la manipuler)
	Command	Encapsuler une requête dans un objet
	Interpreter	Proposer un langage, définir une représentation pour sa grammaire et fournir un interprète capable de manipuler cette grammaire
	Iterator	Proposer séquentiellement une façon d'accéder aux éléments d'un objet
	Mediator	Définir un objet qui encapsule la façon dont interagit un autre ensemble d'objets
	Memento	Sans toucher à l'encapsulation, capturer et extraire l'état d'un objet de manière à ce que cet objet puisse retrouver son état initial
	Observer	Proposer une dépendance entre objets de manière à mettre à jour automatiquement toutes ces dépendances lors du changement d'état d'un objet
	State	Permettre à un objet de modifier son comportement lorsque son état interne change.
	Strategy	Définir une famille d'algorithmes, l'encapsuler, la rendre interchangeable, et permettre à chaque algorithme de varier indépendamment du client qui l'utilise
	Template method	Définir le squelette d'un algorithme en une opération et allouer certaines parties aux sous classes (les sous classes redéfinissent certaines parties de l'algorithme sans changer la structure principale)
	Visitor	Permettre de définir une nouvelle opération sans changement de classe des éléments utilisés

II.3.1 Structure des Patterns

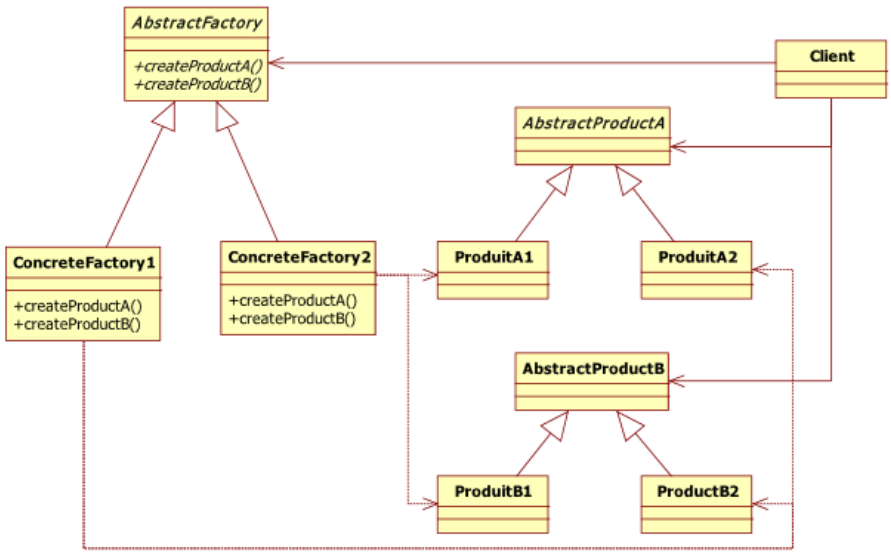
Un pattern est décrit par des propriétés qui expliquent sa raison d'être. Pour GOF, la liste suivante permet de structurer un pattern :

- Nom : Identifie le patron.
- Classification : Permet d'organiser les patrons selon deux critères. Le rôle qui traduit ce que font le patron (création, structuration, comportement), et le domaine qui précise si le patron s'applique en général à une ou plusieurs classes ou objets.
- Intention : Précise le problème de conception soulevé par le patron.
- Alias : Donne d'autres noms connus pour le patron.
- Motivation : Présente un scénario d'application du patron posant son problème dans un contexte particulier. Elle permet de mieux comprendre l'intérêt et la solution du patron.
- Indications d'utilisation : Reconnaît des situations dans lesquelles le patron peut être utilisé.
- Structure : Représente les classes participant dans la solution du patron et montre leurs interactions dans des diagrammes OMT.
- Constituants : Définit les classes et/ou objets participants ainsi que leurs responsabilités.
- Collaboration : Discute la manière dont les constituants collaborent pour assumer leurs responsabilités.
- Conséquences : Décrit comment le patron réalise ses objectifs et présente les résultats, compromis et impacts de l'application de la solution du patron.
- Implantation : Explique la technique d'implémentation du patron. Elle présente les pièges existants et propose des solutions types en fonction du langage utilisé, si elles existent.
- Exemple de code : Donne des parties de code illustrant la solution du patron.
- Utilisations remarquables : Présente des imitations du patron considéré dans des applications connues (dans des Framework, par exemple).
- Patrons apparentés : Référence d'autres patrons utilisant ou utilisés par celui-ci.

II.3.1.1 Abstract factory

Le design pattern Fabrique définit une interface pour la création d'un objet en déléguant à ses sous-classes le choix des classes à instancier.

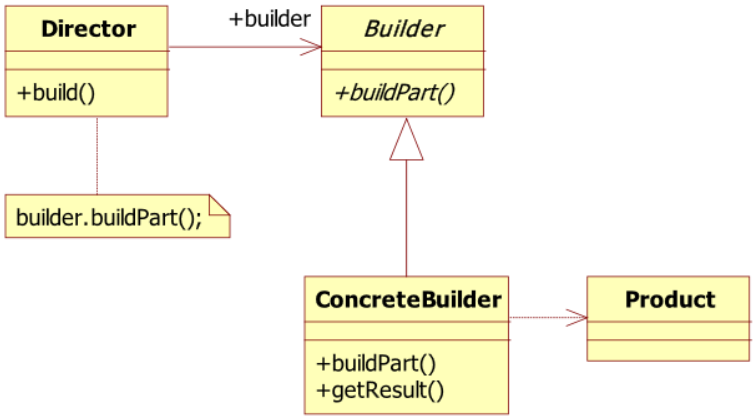
Tableau 2.3: Pattern Abstract factory

Abstract factory
Intention. Fournit une interface pour la création de familles d'objets apparentés ou interdépendants, sans qu'il soit nécessaire de spécifier leurs classes concrètes.
Indications d'utilisation. Le Patron Fabrique est utilisé lorsque : - un système doit être indépendant de la façon dont ses produits ont été créés, combinés, et représentés. - un système doit être constitué à partir d'une famille de produits, parmi plusieurs. - on souhaite renforcer le caractère communautaire d'une famille d'objets conçus pour être utilisés ensemble. - on souhaite fabriquer une bibliothèque de classes, en n'en révélant que l'interface et non l'implémentation
Structure. 
Participants - AbstractFactory : Déclare une interface contenant les opérations de création d'objets AbstractProduct - ConcreteFactory : Implémente les opérations de création d'objets produits concrets - AbstractProduct : Déclare une interface pour un type d'objet produit. - ConcreteProduct : Définit un objet produit qui doit être créé par la fabrique concrète correspondante. Implémente l'interface de AbstractProduct. - Client : N'utilise que les interfaces déclarées par les classes AbstractFactory et AbstractProduct**.**

II.3.1.2 Builder

Le Pattern monteur permet d'isoler des variations de représentations d'un objet. Il permet de créer des objets complexes à partir d'autres objets c'est-à-dire assembler plusieurs objets pour les « monter » et n'en faire qu'un.

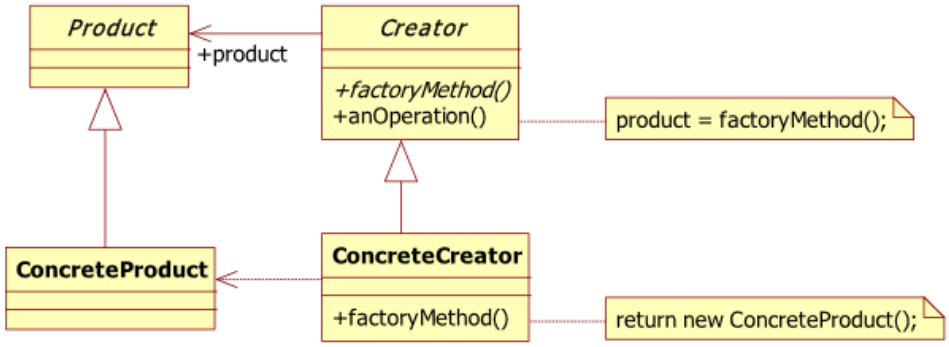
Tableau 2.4: Pattern Builder

Builder
Intention. Dissocie la construction d'un objet complexe de sa représentation, de sorte que le même processus de construction permette des représentations différentes.
Indications d'utilisation. Le Patron Monteur est utilisé lorsque : - l'algorithme de création d'un objet complexe doit être indépendant des parties qui composent l'objet et de la manière dont ces parties sont agencées, - le processus de construction doit autoriser des représentations différentes de l'objet en construction
Structure. 
Participants - Builder : Spécifie une interface abstraite pour la création de parties d'un objet Produit. - ConcreteBuilder : Construit et assemble des parties du produit par implémentation de l'interface Builder. Définit la représentation qu'il crée et en conserve la trace. Fournit une interface pour la récupération du produit final - Director : Construit un objet en utilisant l'interface de Builder. - Product : Surcharge la fabrique pour renvoyer une instance d'un ProduitConcret

II.3.1.3 Factory Method

Le pattern Méthode Fabrique permet l'instanciation d'objets non définis dans une classe concrète à partir d'une méthode d'instanciation issue d'une classe abstraite.

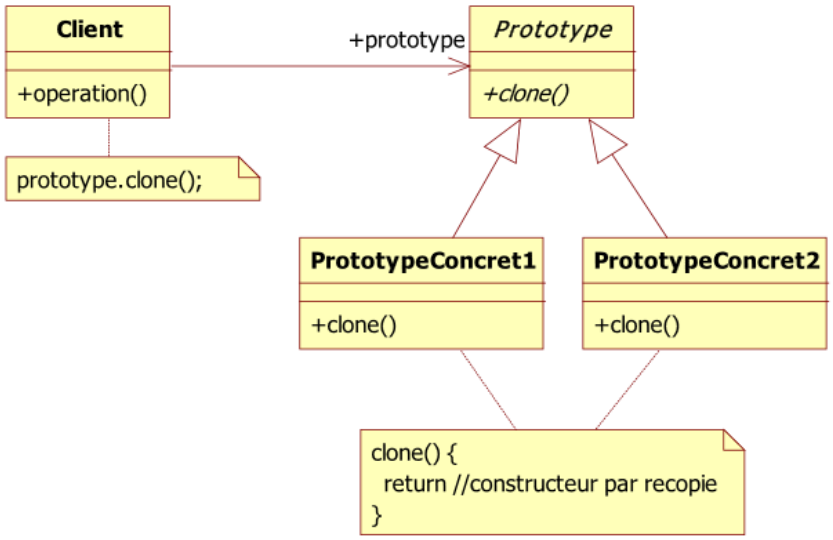
Tableau 2.5: Pattern Method

Factory Method
Intention. Définit une interface pour la création d'un objet, mais en laissant à des sous-classes le choix des classes à instancier. La Fabrique simple permet à une classe de déléguer l'instanciation à des sous-classes.
Indications d'utilisation. Le Patron Méthode Fabrique est utilisé lorsque : - une classe ne peut prévoir la classe des objets qu'elle aura à créer. - une classe attend de ses sous-classes qu'elles spécifient les objets qu'elles créent. - les classes délèguent des responsabilités à une de leurs nombreuses sous-classes assistantes, et que l'on veut disposer localement de l'information permettant de connaître la sous-classe assistante qui a reçu cette délégation
Structure. 
Participants - Product : Définit l'interface des objets créés par la fabrique. - ConcreteProduct: Implémente l'interface Produit. - Creator : Déclare une fabrique ; celle-ci renvoie un objet de type Produit. Le Créateur peut également définir une implémentation par défaut de la fabrique, qui renvoie un objet ConcreteProduct par défaut. Peut appeler la fabrique pour créer un objet Produit - ConcreteCreator : Surcharge la fabrique pour renvoyer une instance d'un ConcreteProduct.

II.3.1.4 Prototype

Pour de raison de performance (la duplication est plus rapide que l'instanciation) et par souci de réutilisation, ce Pattern offre un moyen de créer un nouvel objet en copiant un prototype.

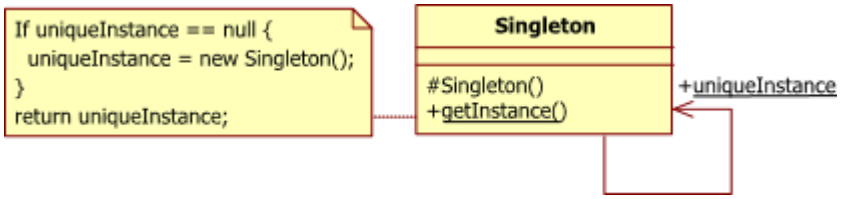
Tableau 2.6 : Pattern Prototype

Prototype
Intention. Spécifie le type des objets à créer à partir d'une instance de prototype, et crée de nouveaux objets en copiant ce prototype.
Indications d'utilisation. On utilise le modèle Prototype lorsqu'un système doit être indépendant de la manière dont ses produits sont créés, composés et représentés ; et : - si les classes à instancier sont spécifiées à l'exécution, par exemple, par chargement dynamique ; ou bien - pour éviter de construire une hiérarchie de classes de fabriques, qui réplique la hiérarchie de classes de produits ; ou encore - si les instances d'une classe peuvent prendre un état parmi un petit nombre de combinaisons. Il peut être plus approprié d'installer le nombre requis de prototypes et d'en faire des clones, plutôt que d'instancier chaque fois la classe manuellement avec l'état correspondant
Structure. 
Participants - Prototype : Déclare une interface pour se cloner lui-même. - PrototypeConcret : Implémente une opération capable de se cloner elle-même. - Client : Crée un nouvel objet en demandant à un prototype de se cloner

II.3.1.5 Singleton

Pour isoler l'unicité d'une instance, ce Pattern offre une classe qui empêche d'autres classes de l'instancier. Elle possède une seule instance d'elle-même et fournit la seule méthode permettant d'accéder à cette instance.
Un Singleton doit restreindre le nombre de ses propres instances à une et une seule. Son constructeur est privé : cela empêche les autres classes de l'instancier. La classe fournit la méthode statique getInstance() qui permet d'obtenir l'instance unique.

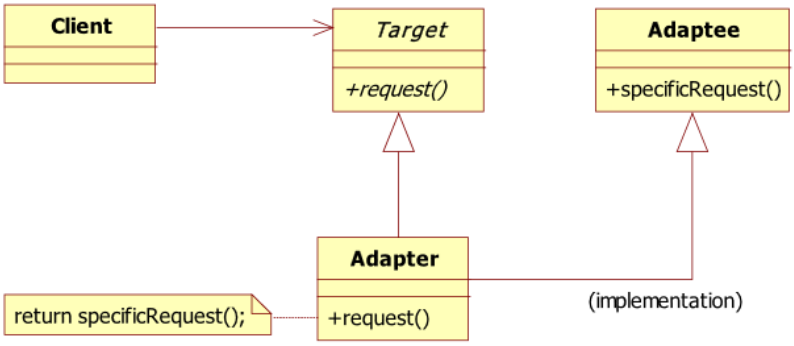
Tableau 2.7: Pattern Singleton

Singleton
Intention. Singleton garantit qu'une classe n'a qu'une seule instance et fournit un point d'accès de type global à cette classe.
Indications d'utilisation. Le Patron Singleton est utilisé lorsque : - s'il doit n'y avoir exactement qu'une instance d'une classe, qui, de plus, doit être accessible aux clients en un point bien déterminé, - si l'instance unique doit être accessible par dérivation en sous-classes, et si l'utilisation d'une instance étendue doit être permise aux clients, sans qu'ils aient à modifier leur code
Structure. 
Participants Singleton : Définit une opération instance() qui donne au client l'accès à son unique instance. Il s'agit d'une opération de classe. Singleton peut avoir la charge de créer sa propre instance unique

II.3.1.6 Adapter

Pour intégrer un sous-système qui a une interface non standard par rapport au système. Le Pattern Adaptateur permet de masquer cette interface non standard au système et de lui présenter une interface standard. La partie cliente utilise les méthodes de l'Adaptateur qui utilise les méthodes du sous-système pour réaliser les opérations correspondantes.

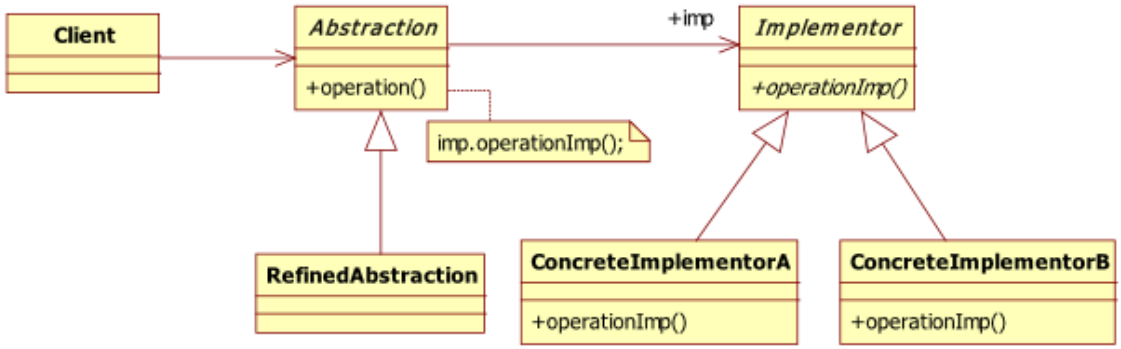
Tableau 2.8: Pattern Adapter

Adapter
Intention. Convertit l'interface d'une classe en une autre conformément à l'attente du client. L'Adaptateur permet à des classes de collaborer, alors qu'elles n'auraient pas pu le faire du fait d'interfaces incompatibles.
Indications d'utilisation. Le Patron Adaptateur est utilisé lorsque : - on utilise une classe existante, mais dont l'interface ne coïncide pas avec celle escomptée. - on souhaite créer une classe réutilisable qui collabore avec des classes sans relations avec elle et encore inconnues, c'est-à-dire avec des classes qui n'auront pas nécessairement des interfaces compatibles. - (pour le cas adaptateur d'objet seulement) vous avez besoin d'utiliser plusieurs sous-classes existantes, mais l'adaptation de leur interface par dérivation de chacune d'entre elles est impraticable. Un adaptateur objet peut adapter l'interface de sa classe parente.
Structure. 
Participants - Target : Définit une interface spécifique du domaine qu'utilise le client. - Client : Collabore avec les objets en se conformant à l'interface du Target. - Adaptee : Définit une interface existante qui demande adaptation. - Adapter: Adapte l'interface de l'adapté à l'interface Target

II.3.1.7 Bridge

Le Pattern Pont permet d'isoler le lien entre une couche de haut niveau et celle de bas niveau.

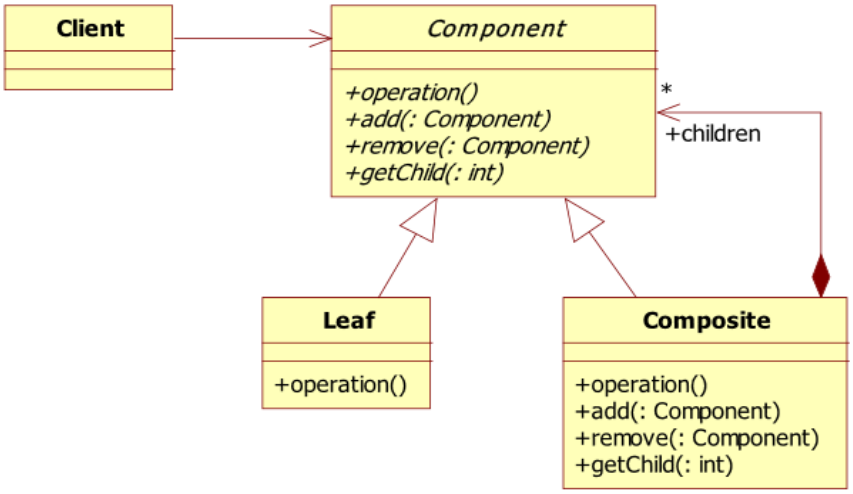
Tableau 2.9: Pattern Bridge

Bridge
Intention. Découple une abstraction de son implémentation afin que les deux éléments puissent être modifiés indépendamment l'un de l'autre.
Indications d'utilisation. Le Patron Pont est utilisé lorsque : - on souhaite éviter un lien définitif entre une abstraction et son implémentation. Ce peut être le cas, par exemple, lorsque l'implémentation doit être choisie parmi plusieurs ou échangée avec une autre lors de l'exécution. - Il faut que les abstractions et leurs implémentations puissent toutes deux être étendues par dérivation. Dans ce cas, le modèle du Pont permet de combiner les différentes abstractions et implémentations, et de les étendre indépendamment. - Il ne faut pas que les modifications apportées à l'implémentation d'une abstraction aient un impact sur le code client (il ne doit pas avoir à être recompilé). - on souhaite cacher complètement au client, l'implémentation d'une abstraction. - On est confronté à une prolifération de classes. Ce type de hiérarchie de classes est typique des cas où il est nécessaire d'éclater un objet en deux parties - On veut faire partager une même implémentation à plusieurs objets.
Structure. 
Participants - Abstraction : Définit l'interface de l'abstraction, gère une référence à un objet de type Implementor. - RefinedAbstraction : Spécialise l'interface définie par l'abstraction. - Implementor : Définit l'interface des classes d'implémentation. Cette interface ne doit pas nécessairement correspondre exactement à l'interface d'abstraction ; en fait, les deux interfaces peuvent être très différentes. En général, l'interface de l'implementor fournit uniquement des opérations primitives, et l'abstraction, des opérations de plus haut niveau, fondées sur ces primitives. - ConcreteImplementor: Implémente l'interface de l'implementor et réalise concrètement son implémentation

II.3.1.8 Composite

Un élément composite est une composition de plusieurs composants composites ou non ; ce type de Pattern offre un moyen pour présenter cette association comme un seul élément face à l'extérieur.

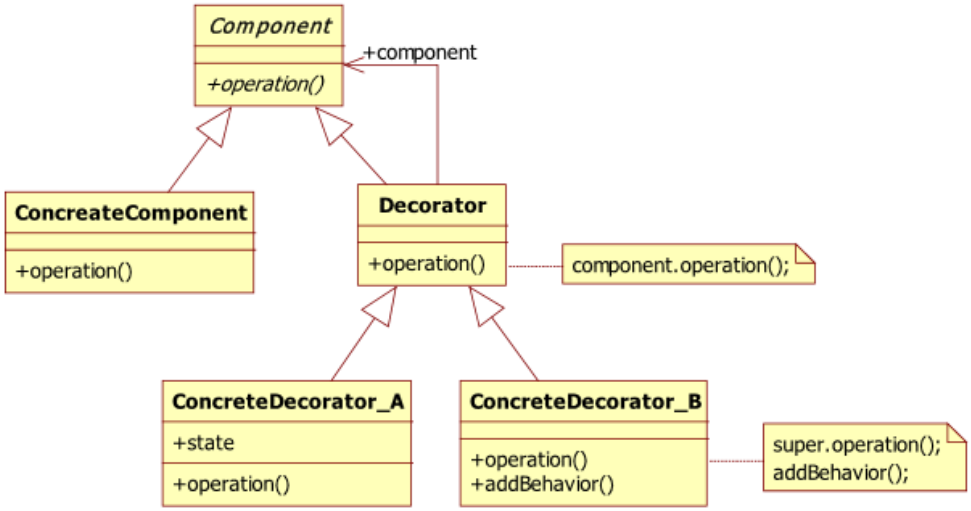
Tableau 2.10: Pattern Composite

Composite
Intention. Le patron Composite propose de composer des objets en des structures arborescentes pour représenter des hiérarchies composant/composé. Il permet au client de traiter de la même et unique façon les objets individuels et les combinaisons de ceux-ci. .
Indications d'utilisation. Le Patron Composite est utilisé lorsque : - on souhaite représenter des hiérarchies de l'individu à l'ensemble. - on souhaite que le client n'ait pas à se préoccuper de la différence entre objets composés et objets individuels. Les clients pourront traiter de façon uniforme tous les objets d'une structure
Structure. 
Participants - Component : Déclare l'interface des objets entrant dans la composition, implémente le comportement par défaut qui convient pour l'interface commune à toutes les classes, etc. - Leaf : Représente des objets feuille dans la composition. Une feuille n'a pas d'enfants. Elle définit le comportement d'objets primitifs dans la composition. - Composite : Définit le comportement des composants dotés d'enfants. Il stocke les composants enfants et implémente les opérations liées aux enfants dans l'interface Component. - Client : Manipule les objets de la composition à l'aide de l'interface composant**.**

II.3.1.9 Decorator

Le Pattern Décorateur permet d'étendre les responsabilités d'une classe sans avoir recours à la création de sous-classe.

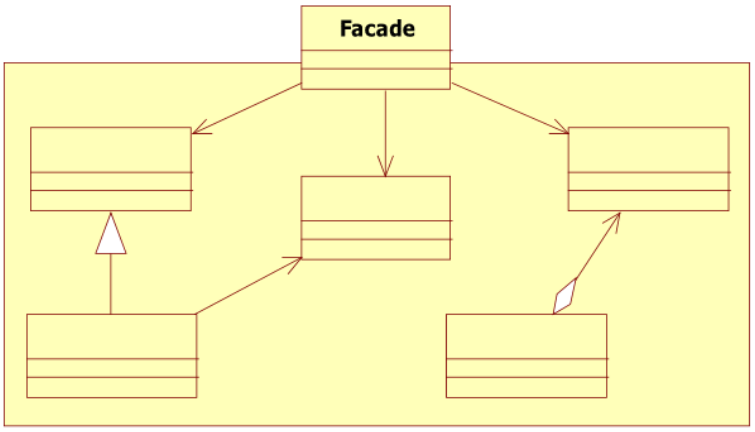
Tableau 2.11: Pattern Decorator

Decorator
Intention. Décorateur attache dynamiquement des responsabilités supplémentaires à un objet. Les décorateurs fournissent une alternative souple à la dérivation pour étendre les fonctionnalités.
Indications d'utilisation. Le Patron Décorateur est utilisé lorsque : - on ajoute dynamiquement des responsabilités à des objets individuels, ceci d'une façon transparente, c'est-à-dire, sans affecter les autres objets, - on rajoute des responsabilités qui peuvent être retirées, - l'extension par dérivation est impraticable. Il peut arriver parfois que l'on ait un grand nombre d'extensions indépendantes possibles. Il en résulte une prolifération explosive de sous classes pour permettre chaque combinaison. D'autres fois, la définition de classe pourra être cachée, ou encore inaccessible pour la dérivation.
Structure. 
Participants - Component : Définit l'interface des objets qui peuvent recevoir dynamiquement des responsabilités supplémentaires. - ConcreteComponent : Définit un objet auquel des responsabilités peuvent être adjointes. - Decorator : Gère une référence à un objet Component et définit une interface conforme à celle du Component. - ConcreteDecorator : Ajoute des responsabilités à un composant.

II.3.1.10 Facade

Le Pattern Façade permet de masquer des opérations disponibles dans des sous-systèmes afin de protéger ou isoler le système.

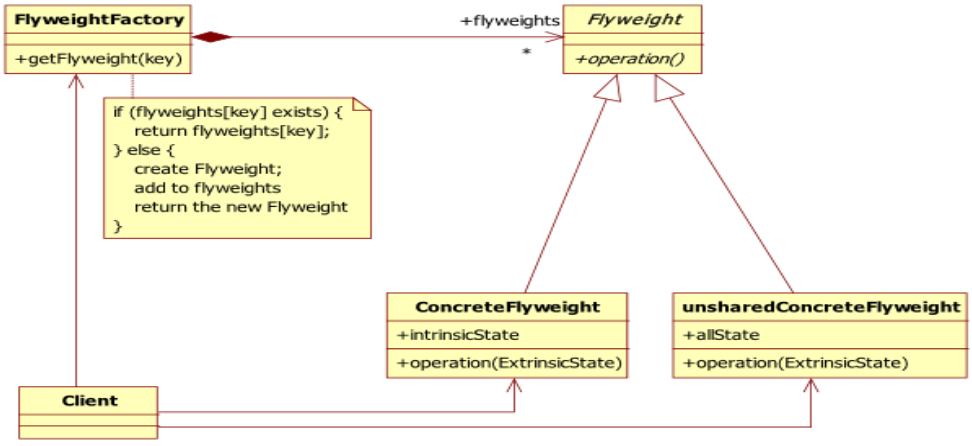
Tableau 2.12: Pattern Facade

Facade
Intention. Fournit une interface unifiée, à l'ensemble des interfaces d'un sous-système. La Façade fournit une interface de plus haut niveau, qui rend le sous-système plus facile à utiliser.
Indications d'utilisation. Le Patron Façade est utilisé lorsque : - on souhaite disposer d'une interface simple pour un sous-système complexe. - il y a beaucoup de relations de dépendance entre les clients et les classes d'implémentation d'une abstraction. On introduira une façade pour découpler le sous-système des clients et des autres sous-systèmes, favorisant ainsi l'indépendance et la portabilité du sous-système. - on cherche à structurer en niveaux un sous-système. On utilisera la façade pour définir un point d'entrée à chaque niveau du sous-système. Si des sous-systèmes sont interdépendants, on peut simplifier les relations entre eux en les faisant communiquer l'un avec l'autre, uniquement à travers leurs façades
Structure. 
Participants - Façade : Connaît les classes du sous-système compétentes pour une requête. - Délègue le traitement des requêtes clients aux objets appropriés du sous-système. - Classes du sous-système : Implémentent les fonctionnalités du sous-système. Gèrent les travaux assignés par l'objet Façade. Ne connaissent pas la façade ; c'est-à-dire qu'elles n'ont pas de références à celle-ci

II.3.1.11 Flyweight

En factorisant les propriétés des éléments de base, le Pattern Poids-mouche permet de créer une seule instance pour présenter ces éléments et alléger la mémoire.

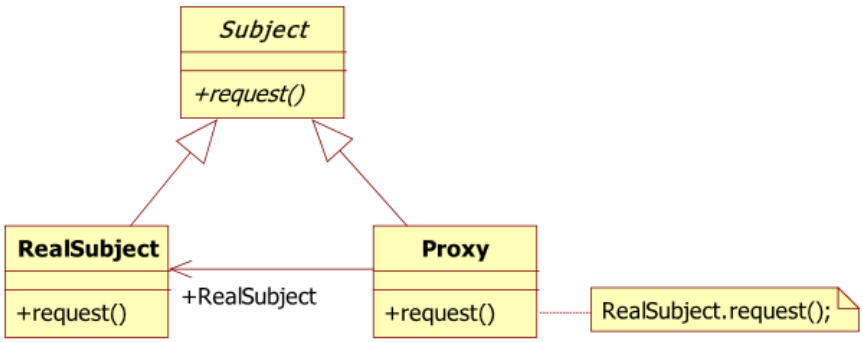
Tableau 2.13: Pattern Flyweight

Flyweight
Intention. Utilise une technique de partage qui permet la mise en œuvre efficace d'un grand nombre d'objets de fine granularité.
Indications d'utilisation. L'efficacité du *Flyweight* dépend beaucoup des conditions de son utilisation. Il faut appliquer le modèle lorsque toutes les conditions ci-après sont réunies. - l'application utilise un grand nombre d'objets. - les coûts de stockage sont élevés du fait d'une réelle quantité d'objets. - la plupart des états de l'objet peuvent être considérés comme extrinsèques. - plusieurs groupes d'objets peuvent être remplacés par un nombre relativement faible d'objets partagés, après que les états extrinsèques aient été retirés. - l'application ne dépend pas de l'identité des objets. Du fait que les objets *Flyweight* peuvent être partagés, des objets de conception distincts peuvent passer pour identiques lors de tests comparatifs.
Structure. 
Participants - Flyweight : Déclare une interface à travers laquelle les Flyweight peuvent recevoir les états extrinsèques et agir sur eux. - ConcreteFlyweight : Implémente l'interface d'un Flyweight et stocke éventuellement les états intrinsèques. Un objet ConcreteFlyweight doit être partageable. Tout état qu'il contient doit être intrinsèque ; c'est-à-dire être indépendant du contexte de l'objet. - UnsharedConcreteFlyweight : Toutes les sous-classes de Flyweight ne sont pas nécessairement partagées. L'interface Flyweight permet le partage, il ne l'impose pas. - FlyweightFactory : Crée et gère des objets Flyweight. S'assure que les objets Flyweight sont convenablement partagés. Si un client réclame un Flyweight, l'objet FlyweightFactory fournit une instance existante ou en crée une s'il n'y en a pas. - Client : Gère une référence aux Flyweight. Calcule ou mémorise l'état extrinsèque des Flyweight

II.3.1.12 Proxy

Le Design Pattern Proxy permet d'isoler le comportement d'un sous-système lors de l'accès à un objet. Cette fonctionnalité permet de sécuriser l'accès et d'inter-changer de fournisseur de service en cas de besoin.

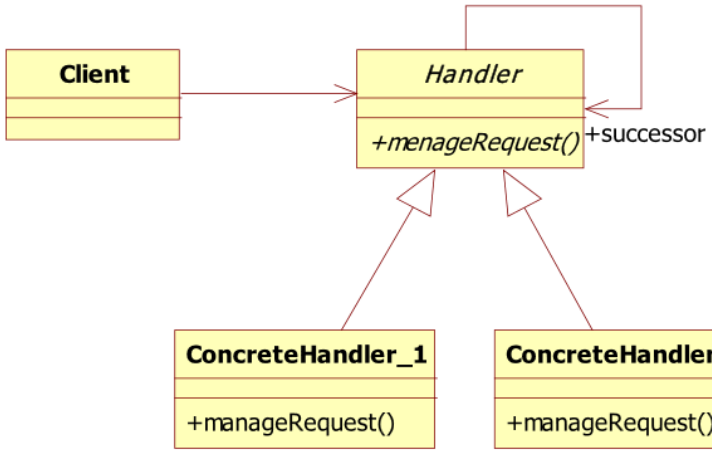
Tableau 2.14: Pattern Proxy

Proxy
Intention. Fournit à un tiers un mandataire ou un remplaçant, pour contrôler l'accès à cet objet.
Indications d'utilisation. L'utilisation du Proxy est indiquée quand on a besoin de références à un objet, qui soient plus créatives et plus sophistiquées qu'un simple pointeur. Suivent quelques situations courantes dans lesquelles le Proxy peut être employé. - une « procuration à distance » fournit un représentant local d'un objet situé dans un espace adresse différent. - une "procuration virtuelle" crée des objets lourds à la demande. - une "procuration de protection" contrôle l'accès à l'objet original. Les procurations de protection sont utiles quand les objets doivent satisfaire différents droits d'accès. - une "référence intelligente" est le remplaçant d'un pointeur brut, qui réalise des opérations supplémentaires, lors de l'accès à l'objet. Quelques utilisations typiques sont : - décompte du nombre des références faites à un objet réel, de sorte que celui-ci puisse être libéré automatiquement, dès qu'il n'y a plus de références. - charger en mémoire un objet persistant quand il est référencé pour la première fois - vérifier, avant d'y accéder, que l'objet réel est verrouillé, pour être sûr qu'aucun autre objet ne pourra le changer
Structure. 
Participants - Proxy : Gère une référence qui lui permet d'accéder au sujet réel. Un Proxy peut faire référence à un Subject, si les interfaces de RealSubject et de Sujet sont les mêmes - Subject : Définit une interface commune pour RealSubject et le Proxy, de sorte que Proxy puisse être utilisée partout où RealSubject est attendu - RealSubject : Définit l'objet réel représenté par le Proxy

II.3.1.13 Chain of responsibility

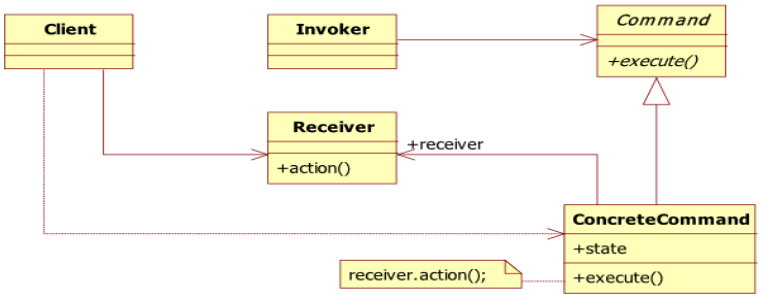
Le Pattern Chaîne de responsabilité permet de gérer le comportement d'un récepteur de requête lors du traitement complexe (piles des tâches).

Tableau 2.15: Pattern Chain of responsibility

Chain of responsability
Intention. Évite le couplage de l'émetteur d'une requête avec ses récepteurs, en donnant à plus d'un objet la possibilité d'entreprendre la requête. Chaîne les objets récepteurs et fait passer la requête tout au long de la chaîne, jusqu'à ce qu'un objet la traite.
Indications d'utilisation. Le Patron Chaîne des Responsabilité est utilisé lorsque : - une requête peut être gérée par plus d'un objet à la fois, et que le gestionnaire n'est pas connu a priori. Ce dernier doit être déterminé automatiquement. - on souhaite adresser une requête à un ou plusieurs objets, sans spécifier explicitement le récepteur. - l'ensemble des objets qui peuvent traiter une requête doit être défini dynamiquement
Structure. 
Participants - Handler : Définit une interface pour gérer les requêtes. Comporte (éventuellement) le code de liaison avec le successeur - ConcreteHandler : Gère les requêtes dont il a la charge. Traite la requête s'il le peut, sinon la transmet à son successeur - Client : Propose initialement la requête à un objet ConcreteHandler de la chaîne

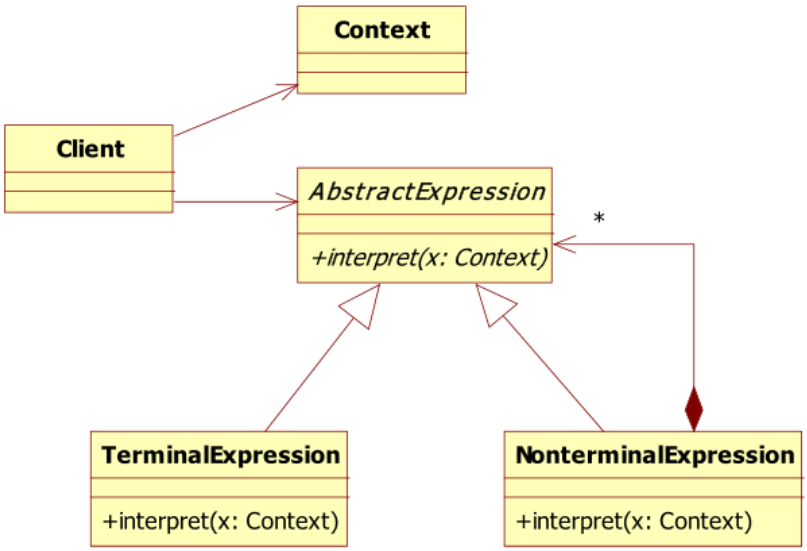
II.3.1.14 Command

Tableau 2.16: Pattern Command

Command
Intention. Encapsule une requête comme un objet, autorisant ainsi le paramétrage des clients par différentes requêtes, files d'attente et récapitulatifs de requêtes, et de plus, permettant la réversion des opérations.
Indications d'utilisation. Le Patron Command est utilisé pour : - introduire dans des objets, sous la forme de paramètres, des actions à effectuer. Un tel paramétrage peut s'exprimer en langage procédural par l'intermédiaire de fonctions callback, c'est-à-dire, des fonctions enregistrées dans un certain contexte pour être appelées ultérieurement. - spécifier, mettre en file d'attente, et exécuter les requêtes à différents instants. Un objet Commande peut avoir une durée de vie indépendante de la requête originale. - assurer des *Undo*. L'opération *Execute* de Commande peut stocker un état pour inverser son effet dans la commande elle-même. L'interface de Commande doit posséder une opération supplémentaire, *Undo*, qui supprime les effets d'un précédent appel à *Execute*. Les commandes exécutées sont stockées dans une liste historique. Un nombre indéterminé d'opérations *Redo* et Undo peuvent s'effectuer par la traversée de cette liste de façon rétrograde ou directe, par appels de *Undo* et *Execute* respectivement. - permettre une mémorisation de modifications, afin de pouvoir les appliquer à nouveau après un éventuel crash système. En ajoutant à l'interface de Commande les opérations charger et stocker, on peut réaliser un enregistrement persistant des modifications. Récupérer d'un crash, implique de recharger les commandes enregistrées depuis le disque et de les exécuter à nouveau à l'aide de l'opération Exécute. - structurer un système autour d'opérations de haut niveau, construites à l'aide de primitives. Une structure de ce type est courante dans les systèmes d'information qui autorisent les transactions. Une transaction encapsule un ensemble de modifications à effectuer sur des données.
Structure. 
Participants - Command : Déclare une interface pour exécuter une opération - ConcreteCommand : Définit une astreinte entre un objet récepteur et une action. Concrétise Execute par l'invocation des opérations adéquates du récepteur. - Client : Crée un objet ConcreteCommand et positionne son récepteur. - Invoker : Demande à la commande d'entreprendre la requête. - Receiver : Définit comment effectuer les opérations associées avec le traitement d'une requête. Tout type de classe peut servir de récepteur.

II.3.1.15 Interpreter

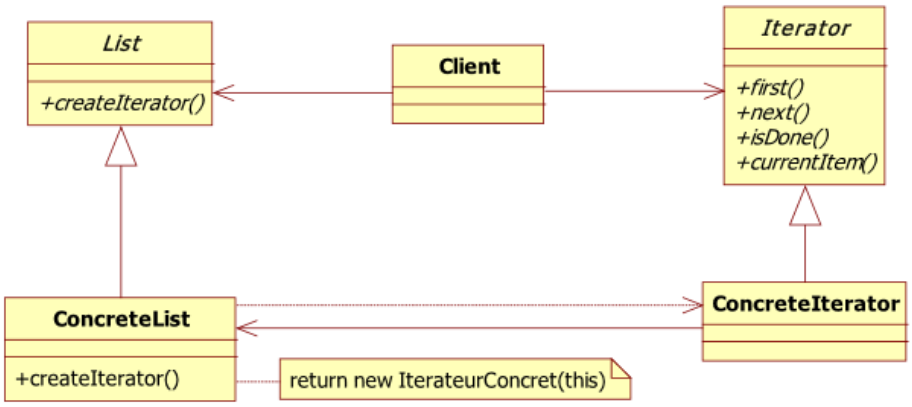
Tableau 2.17: Pattern Interpreter

Interpreter
Intention. Un système doit interpréter un langage. Ce langage possède une grammaire prédéfinie qui constitue un ensemble d'opérations qui peuvent être effectuées par le système. Une structure arborescente peut représenter la grammaire du langage. Elle permet d'interpréter les différents éléments du langage.
Indications d'utilisation. Il faut utiliser l'Interprète, lorsqu'il y a un langage à interpréter, et qu'on peut représenter les déclarations du langage sous forme d'arbres syntaxiques abstraits. L'Interprète fonctionne au mieux lorsque : - la grammaire est simple. Pour les grammaires complexes, la hiérarchie de classes grammaire devient grande et impossible à gérer. Dans de tels cas des outils tels que les générateurs syntaxiques sont de meilleures solutions. Il peut interpréter des expressions sans construire d'arbres syntaxiques abstraits, ce qui permet d'économiser de la place et probablement du temps. - l'efficacité n'est pas un souci majeur. Les Interprètes les plus efficaces ne sont pas généralement implémentés de façon à faire l'interprétation directe d'arbres syntaxiques, mais plutôt pour les traduire préalablement sous une autre forme. Par exemple, les expressions régulières sont fréquemment transformées en machines d'état. Mais même dans ce cas, le traducteur peut être implémenté avec l'Interprète, qui reste donc toujours applicable
Structure. 
Participants - AbstractExpression : Déclare une opération *Interprete* abstraite commune à tous les noeuds de l'arbre syntaxique abstrait - TerminalExpression : Implémente une opération interprète associée, dans la grammaire, à des symboles terminaux. Chaque symbole terminal de la phrase demande une instance de cette classe

- NonterminalExpression : Requise pour chaque règle R : := RI R2 ... Rn, de la grammaire. Gère des variables d'instance du type ExpressionAbstraite, pour chacun des symboles R1 à Rn. Implémente une opération interprète pour les symboles non terminaux de la grammaire. C'est une caractéristique de cette opération de s'appeler récursivement sur les variables représentant R1 à Rn.
- Context : Contient une information à caractère global pour l'interprète
- Client : Construit (ou reçoit) un arbre syntaxique abstrait représentant une phrase particulière dans le langage défini par sa grammaire. L'arbre syntaxique abstrait est constitué d'instances des classes ExpressionNonTerminale et ExpressionsTerminale. Invoque l'opération Interprete

II.3.1.16 Iterator

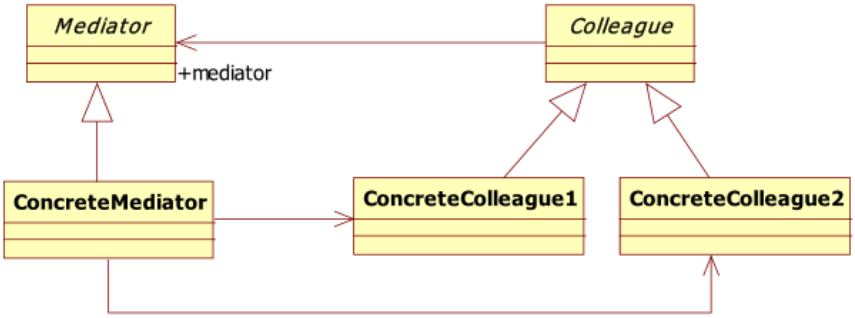
Tableau 2.18: Pattern Iterator

Iterator
Intention. Fournit un moyen d'accès séquentiel aux éléments d'un agrégat d'objets, sans mettre à découvert la représentation interne de celui-ci.
Indications d'utilisation. Le Patron d'Itération est utilisé pour : - accéder au contenu d'un objet d'un agrégat sans en révéler la représentation interne - gérer simultanément plusieurs parcours dans des agrégats d'objets - offrir une interface uniforme pour les parcours au travers de diverses structures agrégats (itération polymorphe).
Structure. 
Participants - Iterator : Définit une interface pour accéder aux éléments et les parcourir. - ConcreteIterator : Implémente l'interface Iterator. Assure le suivi de l'élément courant lors de la traversée d'un agrégat. - List : Définit une interface pour la création d'un objet Iterator. - ConcreteList: Implémente l'interface de création d'Iterator, afin de retourner l'instance adéquate de l'IterateurConcret

II.3.1.17 Mediator

Différents objets ont des interactions. Un événement sur l'un provoque une action ou des actions sur un autre ou d'autres objets. Le Pattern Médiateur permet de gérer la communication entre ces éléments.

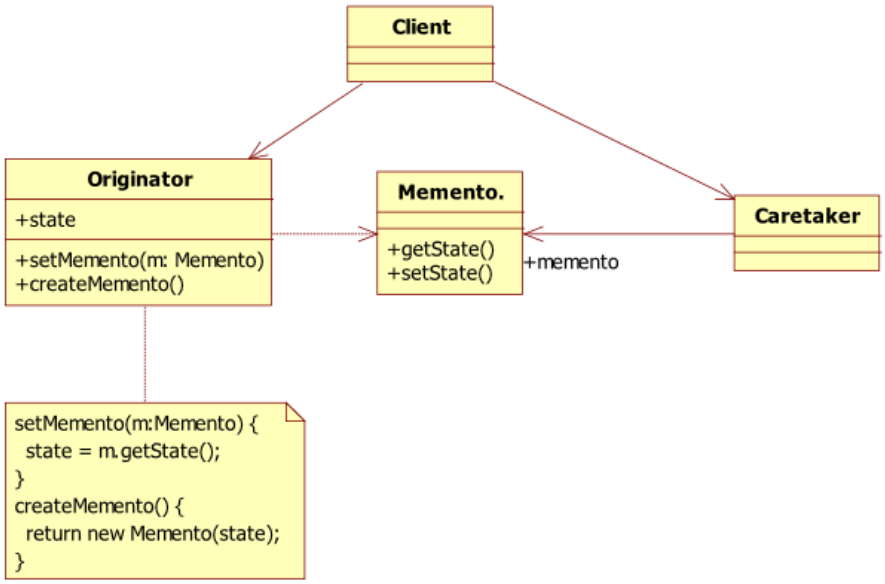
Tableau 2.19: Pattern Mediator

Mediator
Intention. Définit un objet qui encapsule les modalités d'interaction d'un certain ensemble d'objets. Le médiateur favorise le couplage faible en dispensant les objets de se faire explicitement référence, et il permet donc de faire varier indépendamment les relations d'interaction.
Indications d'utilisation. Le Patron Médiateur est utilisé lorsque : - les objets d'un ensemble communiquent d'une façon bien définie mais très complexe et que le résultat des interdépendances est non structuré et difficile à appréhender. - la réutilisation d'un objet est difficile, du fait qu'il fait référence à beaucoup d'autres objets et communique avec eux. - un comportement distribué entre plusieurs classes doit pouvoir être spécialisé sans une pléthore de dérivations.
Structure. 
Participants - Mediator : Définit une interface pour communiquer avec les objets collègues - ConcreteMediator : Réalise le comportement coopératif en coordonnant les objets Collegue. Il connaît et gère ses collègues - Colleague : Chaque classe Collegue connaît son objet Médiateur. Chaque collègue s'adresse à son médiateur pour communiquer avec un autre collègue.

II.3.1.18 Memento

Les informations de l'état interne d'un système sont conservées dans un memento. Un objet est défini dans le système comme étant le créateur du memento. Afin de respecter l'encapsulation, les valeurs du memento ne sont visibles que par son créateur. Ainsi, l'encapsulation de l'état interne est préservée. Un autre objet est chargé de conserver les « mementos » (gestionnaire d'annulation) : il s'agit du gardien.

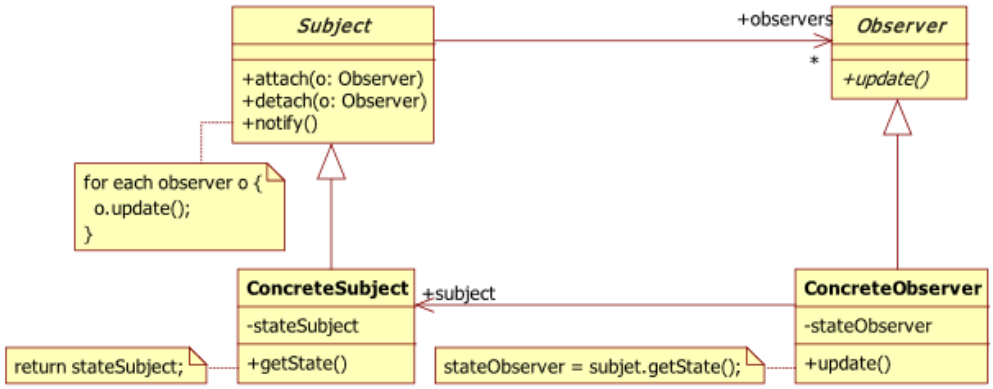
Tableau 2.20: Pattern Memento

Memento
Intention. Sans violer l'encapsulation, le Mémento permet de saisir et de transmettre à l'extérieur d'un objet l'état interne de celui-ci, dans le but de pouvoir ultérieurement le restaurer dans cet état.
Indications d'utilisation. Le Patron Memento est utilisé lorsque : - l'état d'un objet doit être sauvegardé de façon à pouvoir restaurer ultérieurement celui-ci - l'utilisation d'une interface directe pour atteindre l'état, conduirait à révéler des détails de l'implémentation et à rompre l'encapsulation.
Structure. 
Participants - Memento : Mémorise l'état interne de l'objet original. Il peut stocker autant d'informations d'état interne de l'original que nécessaire, à la discrétion de ce dernier. - Originator : Crée un objet Memento contenant un instantané de son état interne courant. Utilise Memento pour restaurer son état interne. - Caretaker : responsable de la sauvegarde de Memento. Il n'agit pas sur le Memento.

II.3.1.19 Observer

Ce Pattern a pour rôle d'écouter les changements d'état d'un système et de diffuser les informations vers les sous-systèmes si un évènement se présente.

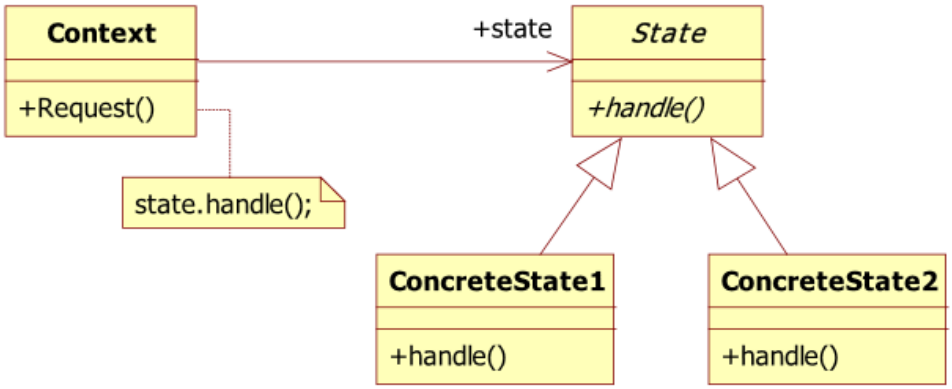
Tableau 2.21: Pattern Observer

Observer
Intention. Observateur définit une interdépendance de type un à plusieurs, de façon telle que, quand un objet change d'état, tous ceux qui en dépendent en soient notifiés et automatiquement mis à jour.
Indications d'utilisation. Le Patron Observateur est utilisé lorsque : - quand un concept a deux représentations, l'une dépendante de l'autre. - quand la modification d'un objet nécessite de modifier les autres, et que l'on ne sait pas combien ils sont. - quand un objet doit être capable de faire une notification à d'autres objets sans faire d'hypothèse sur la nature de ces objets.
Structure. 
Participants - Subject : Un nombre quelconque d'observateurs peut observer un sujet. Il fournit une interface pour attacher et détacher les objets observateurs. - Observer : Définit une interface de mise à jour pour les objets qui doivent être notifiés de changements dans un sujet. - ConcreteSubject : Mémorise les états qui intéressent les objets ConcreteObserver. Il envoie une notification à ses observateurs lorsqu'il change d'état. - ConcreteObserver : Gère une référence sur un objet ConcreteSubject et mémorise l'état qui doit rester pertinent pour le sujet. Il fait l'implantation de l'interface de mise à jour de l'observateur pour conserver la cohérence de son état avec le sujet.

II.3.1.20 State

Un objet a un fonctionnement différent selon son état interne. Son état change selon les méthodes appelées. Les différents états internes sont chacun représenté par une classe état. Les états possèdent des méthodes permettant de réaliser (ou bloquer) les opérations et de changer d'état. Le Pattern Etat présente les opérations à la partie cliente.

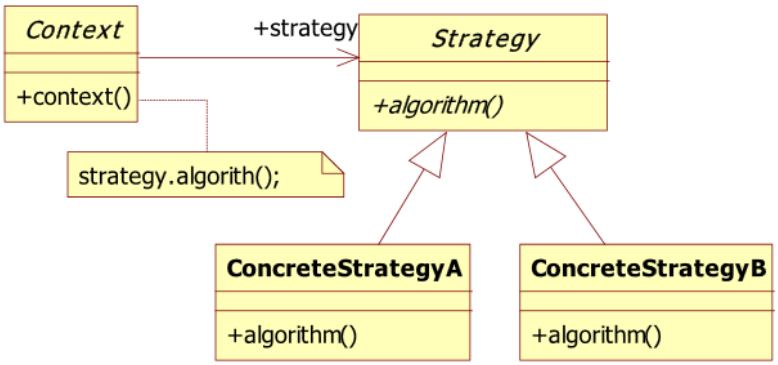
Tableau 2.22: Pattern State

State
Intention. Permet à un objet de modifier son comportement, quand son état interne change. Tout se passe comme si l'objet changeait de classe.
Indications d'utilisation. Le Patron Etat est utilisé lorsque : - le comportement d'un objet dépend de son état, et son comportement doit changer dynamiquement, en fonction de cet état. - les opérations comportent de liste des déclarations conditionnelles dépendant de l'état de l'objet. Cet état est généralement désigné par une ou plusieurs constantes d'énumération
Structure. 
Participants - Context : Gère une instance d'une sous-classe ConcreteState qui définit l'état en cours - State : Définit une interface qui encapsule le comportement associé avec un état particulier de Contexte - ConcreteState : Chaque sous-classe implémente un comportement associé avec l'état de contexte

II.3.1.21 Strategy

Si un objet doit varier une partie de son algorithme. Le Pattern Stratégie offre une interface pour modéliser les différents algorithmes possibles afin qu'on puisse les inter-changer.

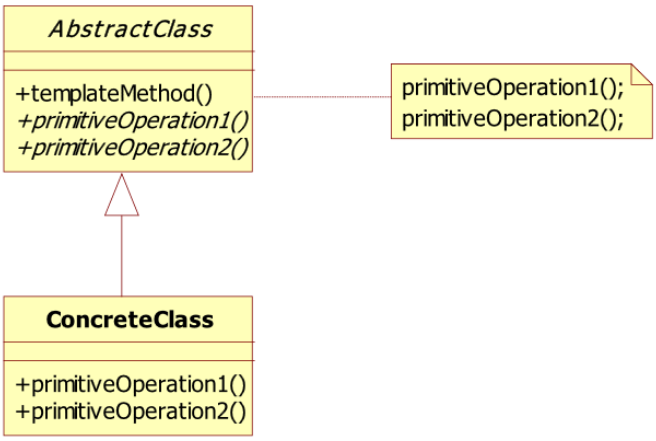
Tableau 2.23: Pattern Strategy

Strategy
Intention. Définit une famille d'algorithmes, encapsule chacun d'entre eux et les rend interchangeables. La Stratégie permet aux algorithmes d'évoluer indépendamment des clients qui les utilisent.
Indications d'utilisation. Le Patron Stratégie est utilisé lorsque : - plusieurs classes apparentées ne diffèrent que par leur comportement. - on a besoin de diverses variantes d'un algorithme. Les Stratégies peuvent être utilisées quand ces variantes sont implémentées sous la forme d'une hiérarchie de classe d'algorithmes. - un algorithme utilise des données que les clients n'ont pas à connaître. Utiliser la Stratégie dispense d'avoir à révéler des structures complexes de données spécifiques des algorithmes. - une classe définit de nombreux comportements, qui figurent dans ses opérations sous la forme de déclarations conditionnelles multiples. Plutôt que laisser subsister ces expressions conditionnelles, on déplacera les sections correspondantes aux différentes branches, dans des classes Stratégie qui leurs sont propres
Structure. 
Participants - Strategy : Déclare une interface commune à tous les algorithmes représentés. Contexte utilise cette interface pour appeler l'algorithme défini par une ConcreteStrategy - ConcreteStrategy : Implémente l'algorithme, en utilisant l'interface Stratégie - Context : Définir une interface qui permette à Stratégie d'accéder à ses données

II.3.1.22 Template method

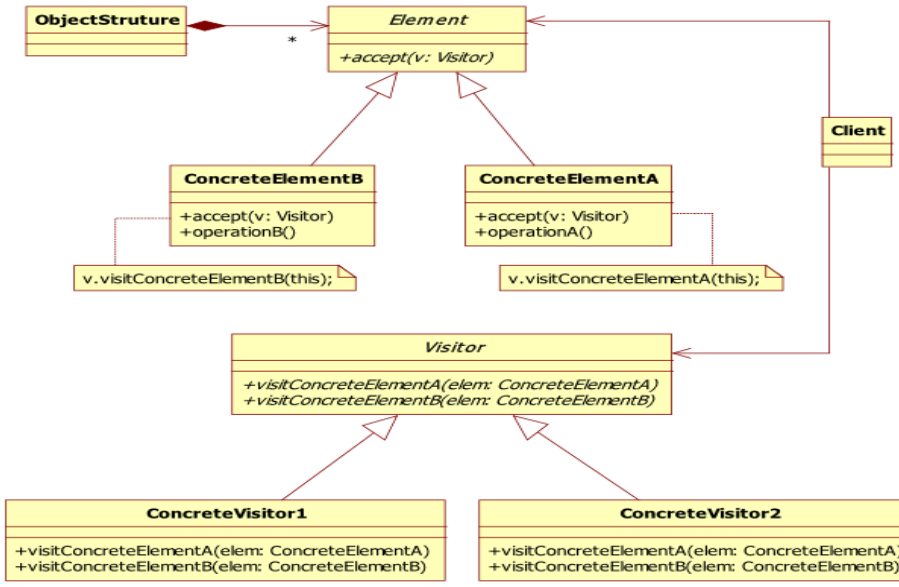
Pour séparer les traitements génériques et les traitements spécifiques liés à la nature de l'objet, le Pattern Patron utilise deux types de classe pour pouvoir utiliser la classe « concrète » derrière le patron.

Tableau 2.24: Pattern Template Method

Template method
Intention. Définit, dans une opération, le squelette d'un algorithme, en en déléguant certaines étapes à des sous-classes. Le Patron de méthode permet de redéfinir par des sous-classes, certaines parties d'un algorithme, sans avoir à modifier la structure de ce dernier.
Indications d'utilisation. Le Patron de méthode est utilisé lorsque : - on laisse aux sous-classes le soin d'implémenter les parties dont le comportement est conçu comme modifiable. - on évite la duplication de code, isoler le facteur commun des comportements de sous-classes, et l'implémenter dans une classe commune. - on veut contrôler des extensions de sous-classes.
Structure. 
Participants - AbstractClass : Implémente une méthode primitiveOperation définissant le squelette d'un algorithme qui fait appel aux méthodes primitives abstraites. - ConcreteClass : Implémente les opérations primitives qui assurent l'exécution de chaque étapes de l'algorithme définit dans la super classe

II.3.1.23 Visitor

Tableau 2.25: Pattern Visitor

Visitor
Intention. Visiteur réalise la représentation d'une opération applicable aux éléments d'une structure d'objet. Il permet de définir une nouvelle opération, sans qu'il soit nécessaire de modifier la classe des éléments sur lesquels elle agit.
Indications d'utilisation. On utilisera le modèle Visiteur dans les circonstances suivantes. - une structure d'objets contient beaucoup de classes différentes d'interfaces distinctes, et l'on veut réaliser des opérations sur ces objets qui dépendent de leurs classes. - il s'agit d'effectuer plusieurs opérations distinctes et sans relation entre elles, sur les objets d'une structure, et ceci en évitant de polluer leurs classes avec des opérations. Le visiteur permet de regrouper toutes les opérations du même type dans une seule classe, quand la structure est partagée par plusieurs applications qui seules en ont besoin. - les classes qui définissent la structure d'objets changent rarement, mais on doit souvent définir de nouvelles opérations sur cette structure. Modifier les classes de la structure impose de redéfinir l'interface avec tous les visiteurs, ce qui peut être coûteux. Si les classes de la structure d'objets changent souvent, il est sans aucun doute préférable de définir les opérations dans des classes.
Structure. 
Participants - Visitor : Déclare une opération visitConcreteElementX() pour chaque classe élément concret ConcreteElementX. Le nom de l'opération et sa signature identifient la classe émettrice de la requête vers le visiteur, qui peut ensuite accéder à l'élément directement, au travers de son interface particulière. - ConcreteVisitor : Concrétise par codage chaque opération déclarée par la classe Visitor. - Element : Définit une opération accept() qui prend pour argument un visiteur. - ConcreteElement : Réalise le codage d'une opération accept().

II.3.2 Portée des Patterns GOF

La portée d'un pattern définit le niveau d'utilisation d'un pattern dans la conception de logicielle. On a deux niveaux de portée applicable pour les patterns de GOF :

- niveau classes
- niveau objets

Les patterns au niveau des classes s'occupent des relations entre les classes et les sous-classes. Ces relations sont fixées statiquement à la compilation.

Exemple : Les patterns de création au niveau des classes délèguent une partie de la création des objets aux sous-classes.

Les patterns au niveau des objets s'occupent des relations entre les objets, lesquels sont plus dynamiques et peuvent être modifiés à l'exécution

Exemple : Les patterns de création au niveau des objets délèguent la création à d'autres objets.

Tableau 2.26 : Portée des Pattern de Gang of Four

		Catégorie		
		Création	Structure	Comportement
Portée	Classe	Factory	Method Adapter	Interpreter
				Template Method
	Objet	Abstract Factory	Adapter	Chain of Responsibility
		Builder	Bridge	Command
		Prototype	Composite	Iterator
		Singleton	Decorator	Mediator
			Facade	Memento
			Flyweight	Observer
			Proxy	State
				Strategy
				Visitor

Les Design Patterns de *Gang of Four* permettent d'améliorer la qualité de développement et d'en diminuer la durée. En effet, leur application réduit les couplages (points de dépendance) au sein d'une application, apporte de la souplesse, favorise la maintenance et d'une manière générale aide à respecter de « bonnes pratiques » de développement.

II.4 Pattern Architectural

II.4.1 Architecture logicielle

L'architecture logicielle décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs relations et leurs interactions.

L'architecture logicielle possède plusieurs styles. Parmi les styles d'architecture proposés, les plus utilisés et prouvés sont :

- architecture en couches
- architecture centrée sur les données
- architecture en flot de données
- architecture orientée objets
- architecture orientée agents
- architecture multi-tiers
- architecture SOA

II.4.1.1 Architecture en couches

Ce type de style est organisé de manière hiérarchique, chaque couche effectue un service pour la couche de dessus et sert de client pour la couche de dessous. Dans certains systèmes, les couches intérieures sont cachées de toutes les couches sauf des couches adjacentes.
Les connecteurs sont les protocoles qui déterminent la façon dont les couches interagissent.

II.4.1.2 Architecture centrée sur les données

Cette architecture sépare clairement les données (serveurs) des traitements et de la présentation (clients) et permet ainsi une très grande intégrabilité, en effet, des clients peuvent être ajoutés sans affecter les autres clients. Par contre, tous les clients sont dépendants de l'architecture des données qui doit rester stable et qui est peu extensible.

II.4.1.3 Architecture en flot de données

Cette architecture est composée de plusieurs composants logiciels reliés entre eux par des flux de données. L'information circule dans le réseau et est transformée par les différents composants qu'elle traverse.

II.4.1.4 Architecture orientée objets

Cette architecture est basée sur les propriétés d'un objet qui intègre les données et les opérations de traitement. Cette architecture est décrite par les principes de conception orientée objet :

- protection des variations : Identifier les points de variation et d'évolution, et séparer ces aspects de ceux qui demeurent constants
- faible couplage : Réduire l'impact des modifications en affectant les responsabilités de façon à minimiser les dépendances entre classes
- forte cohésion : Faciliter la compréhension, gestion et réutilisation des objets en concevant des classes à but unique

- indirection : Limiter le couplage et protéger des variations en ajoutant des objets intermédiaires
- composer au lieu d'hériter : Limiter le couplage en utilisant la composition (boite noire) au lieu de l'héritage (boite blanche) pour déléguer une tâche à un objet

II.4.1.5 Architecture orientée agents

Ce type de style réorganise les objets (composant passif) pour créer un composant projectif (service) :
L'agent logiciel, utilise de manière relativement autonome, avec une capacité d'exécution propre, les autres agents pour réaliser ses objectifs : il établit des dialogues avec les autres agents, il négocie et échange de l'information, décide à chaque instant avec quels agents communiquer en fonction de ses besoins immédiats et des disponibilités des autres agents.

II.4.1.6 Architecture multi-tiers

Cette architecture est une manière de segmenter une application en plusieurs blocs (le N dans N-tiers fait référence au nombre de blocs) plus ou moins indépendants qui communiquent entre eux par le biais d'interfaces et d'objets de transfert.

La subdivision de base est de trois tiers :

- données
- logique de métier
- IHM

II.4.1.7 Architecture SOA

Définition 2.3 :

Un service est une valeur livrée à quelqu'un d'autre à travers une interface bien définie, disponible pour une communauté. Un service a comme résultat un travail fait par l'un et fourni à l'autre.
L'architecture SOA permet aux systèmes de fournir des services entre eux. Cette organisation aide le concepteur à séparer la question de ce qu'il faut faire de la question de comment le faire, où, par qui ou par quoi.

II.4.2 Pattern Architectural

Définition 2.4 :

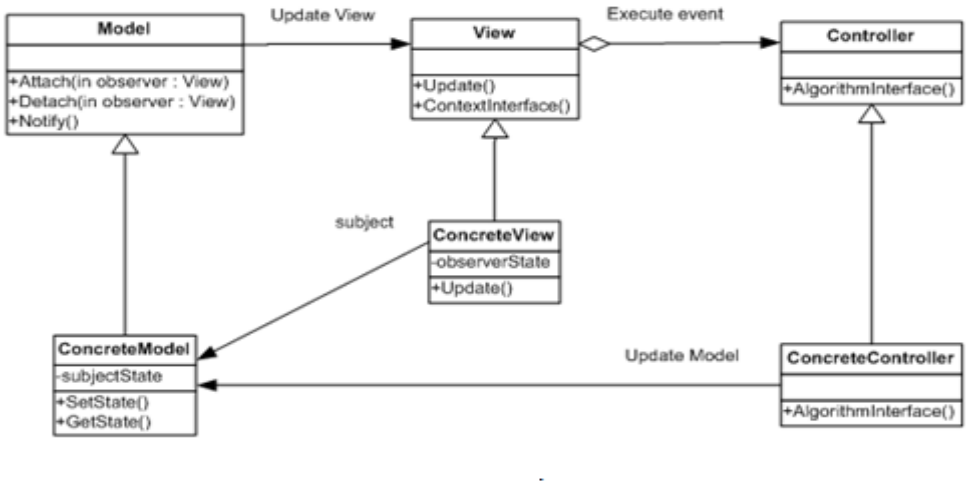
Un pattern architectural définit le schéma de la structure des systèmes logiciels. Il fournit un ensemble de sous-systèmes prédéfinis, spécifie leurs responsabilités ainsi que des règles et des directives pour l'organisation des relations entre eux.

Pour répondre au problème d'organisation des entités logicielles, l'expérience de modélisation de système offre les patterns suivants :

- Model-view-controller pattern
- Layers pattern
- Client-server pattern
- Master-slave pattern
- Pipe-filter pattern
- Broker pattern
- Peer-to-peer pattern
- Event-bus pattern
- Blackboard pattern
- N-tiers pattern
- Service-Oriented pattern

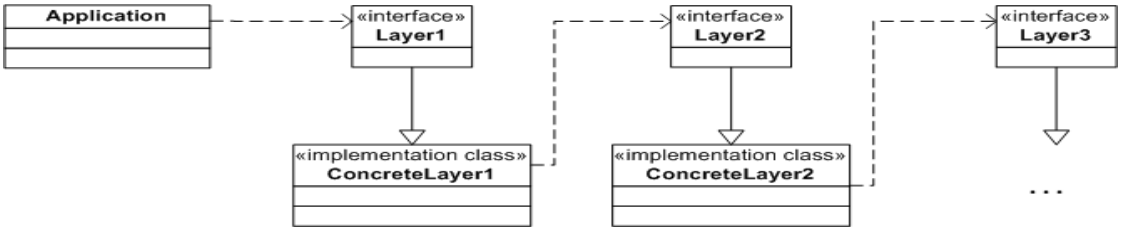
II.4.2.1 MVC

Tableau 2.27: Pattern MVC

MVC
Intention. Organiser les objets graphiques, les objets métier et les données dans une application
Indications d'utilisation. On utilise cette architecture pour gérer les interactions dans les applications qui possèdent et ou utilisent des interfaces graphiques.
Structure. 
Participants - le modèle qui contient la logique et l'état de l'application ; - la vue qui représente l'interface utilisateur ; - le contrôleur qui gère la synchronisation entre la vue et le modèle - concreteX : implémentation des traitements

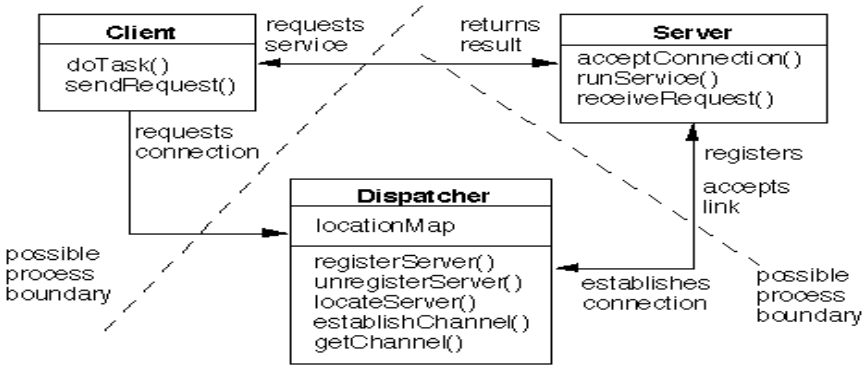
II.4.2.2 Layers

Tableau 2.28: Pattern Layers

Layers
Intention.: Mettre en place un niveau de traitement et de conception hiérarchisé sur un seul axe.
Indications d'utilisation. On utilise cette architecture pour : - modéliser des Framework - gérer des communications inter-système
Structure. 
Participants - Layer : Interface qui définit la responsabilité du niveau - ConcreteLayer : Traitement : acquisition – transformation – livraison

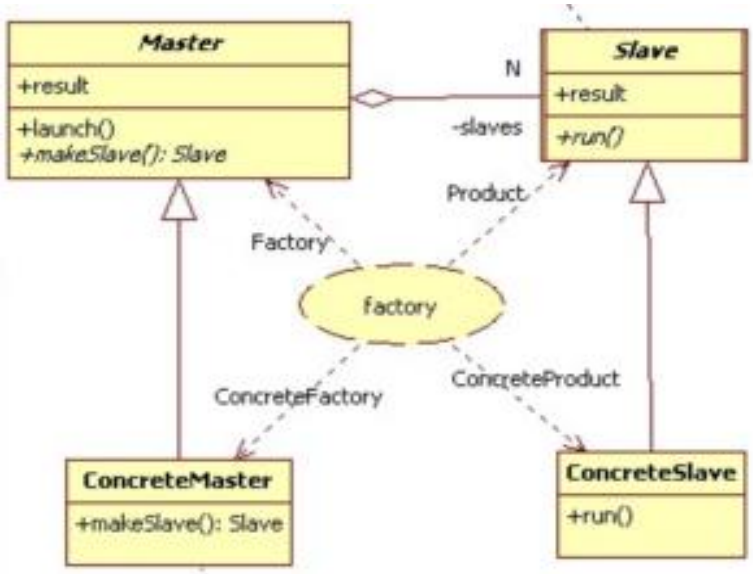
II.4.2.3 Client-server

Tableau 2.29: Pattern Client-Server

Client-server
Intention.: Organiser les traitements dans un seul point, et accessibles via une requête.
Indications d'utilisation. On utilise ce style pour délivrer des services à des clients distants et de centraliser les traitements.
Structure. 
Participants - Client : Demandeur de service - Server : Fournisseur de service - Dispatcher : Gestionnaire des serveurs.

II.4.2.4 Master-slave

Tableau 2.30: Pattern Master-Slave

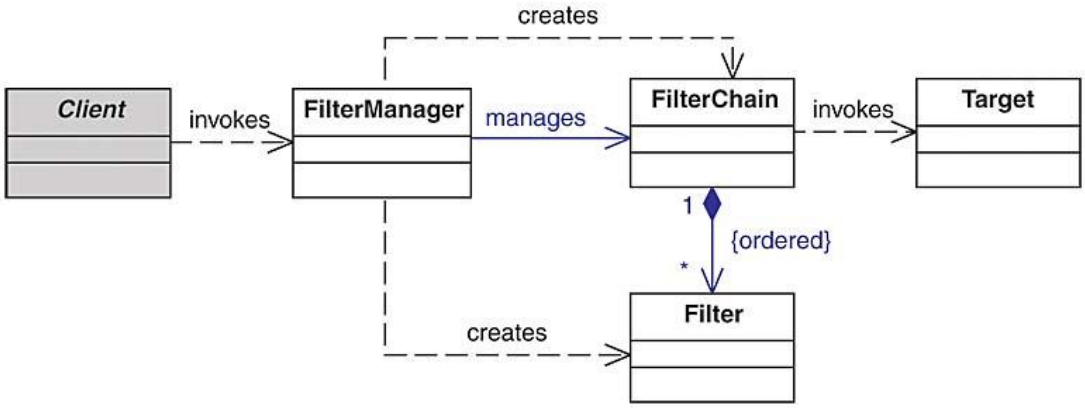
Master-slave
Intention : Organiser plusieurs serveurs de même nature afin qu'ils puissent être accessibles via une seule entité.
Indications d'utilisation. On utilise cette architecture pour mettre en place des traitements parallèles avec le principe de délégation des tâches : - éviter la rupture de traitement en cas d'indisponibilité d'un serveur - éviter les limitations en ressource
Structure. 
Participants - Master : Gère la relation avec le client et le Slave - Slave : Interface qui définit le traitement - ConcreteMaster : Implémente Master pour appeler les Slaves - ConcreteSlave : Entité qui réalise la demande du client .

Cette technique de duplication de serveur (un maitre et plusieurs esclaves) permet d'assurer le traitement dans un système où l'indisponibilité du service entraine un chaos total :

- authentification distribué
- base de données
- commande engin automatique
- etc.

II.4.2.5 Pipe-filter

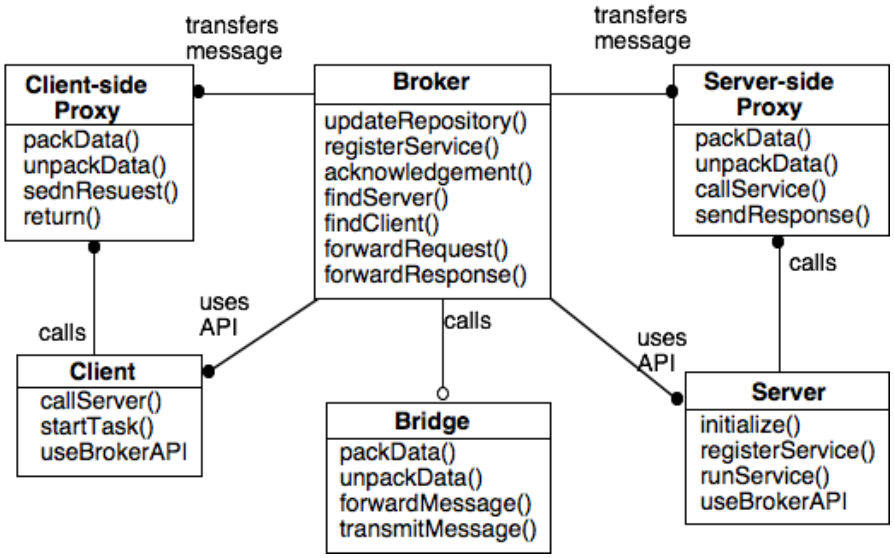
Tableau 2.31: Pattern Pipe-filter

Pipe-filter
Intention : Organiser les traitements comme un enchainement de tuyau de pipeline
Indications d'utilisation. On utilise ce style pour décomposer le traitement de flux de données par des séquences de filtres uniformes : - pour gérer les ressources partagées - pour subdiviser une tâche complexe en une série de tâches simples
Structure. 
Participants - Client : Générateur de message - FilterManager : Gère une liste de filtre - FilterChain : Interface pour uniformiser les filtres - Filter : implémente le traitement du message - Target : destination du message .

Ce Pattern représente le principe général de la gestion de métier en processus et le traitement en services. La décomposition nécessite la maitrise des propriétés commune dans le *filter* et le gestionnaire de la liste des éléments obtenus.

II.4.2.6 Broker

Tableau 2.32: Pattern Broker

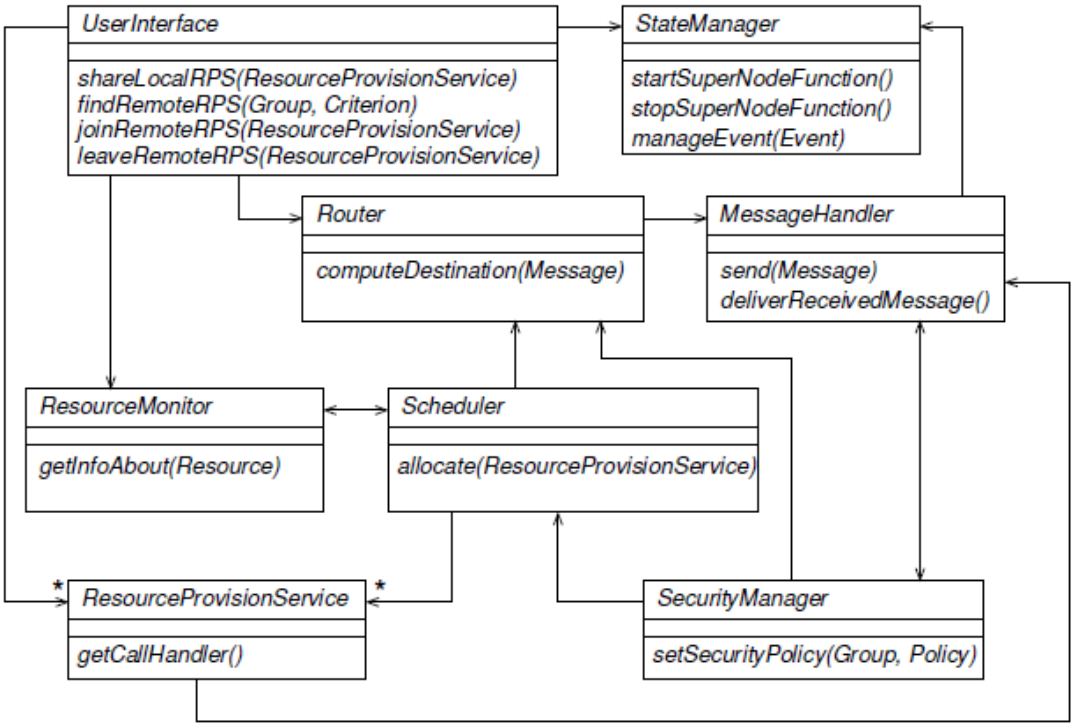
Broker
Intention : Organiser le système pour pouvoir distribuer les services via le partage des objets.
Indications d'utilisation. On utilise cette architecture pour dispatcher des requêtes dans un système de composants distribués.
Structure. 
Participants - Server : Offre un service - Client : Demandeur de service via une requête - Broker : Joue l'intermédiaire (trader) entre le client et le serveur - Bridge : une couche qui permet de relier des brokers - Server-sideProxy : Une interface qui permet d'accéder au serveur

En utilisant ce Pattern, un annuaire de service peut gérer les instances de traitement en cours et optimiser sa relation avec le Bus événement :

- identifier les serveurs où s'exécutent les services
- créer les événements
- gérer l'aboutissement de requête
- gérer le proxy
- gérer la structure des données

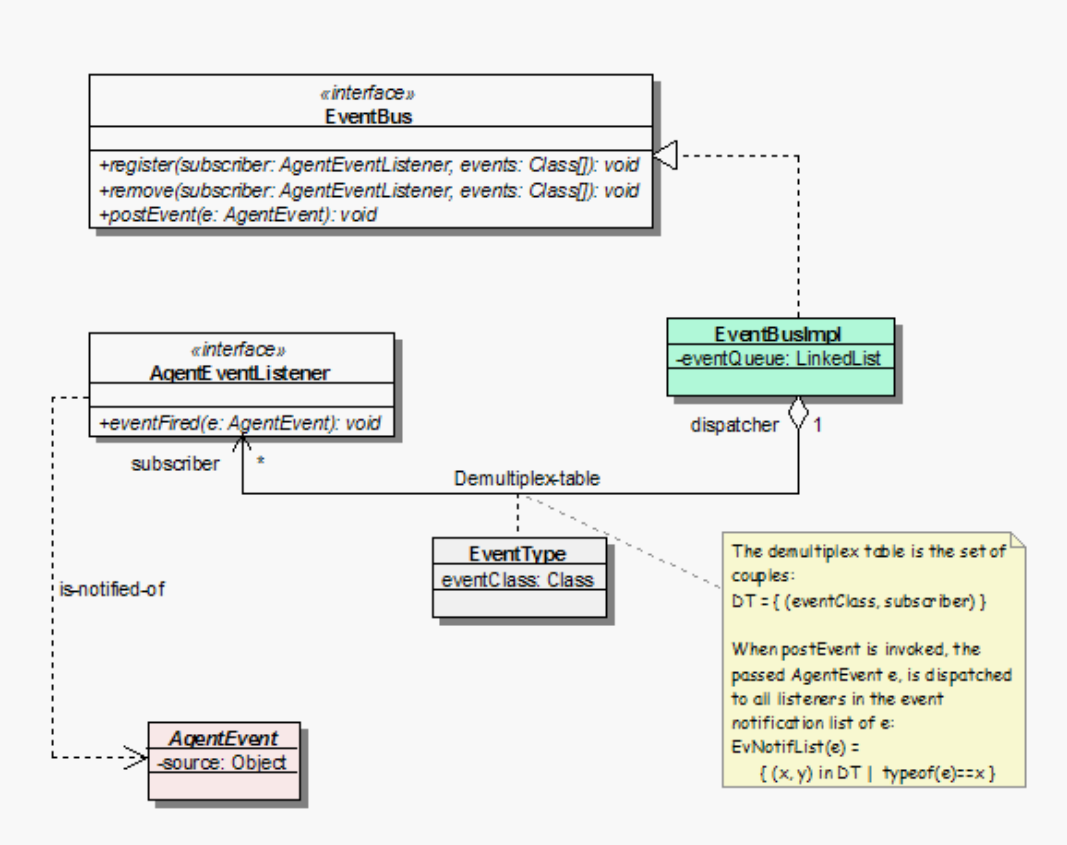
II.4.2.7 Peer-to-peer

Tableau 2.33: Pattern Peer to peer

Peer-to-peer
Intention : Créer une organisation qui collabore pour déployer, publier, découvrir et consommer des ressources hétérogènes.
Indications d'utilisation. On utilise cette architecture pour diffuser et partager des informations avec réplication entre plusieurs systèmes client/serveur.
Structure. 
Participants - StateManager : Gère les événements et la communication asynchrone entre système - UserInterface : Définit le point d'accès pour gérer les ressources pour un système - Router : Définit le moyen pour relier les systèmes - MessageHandler : Observateur d'événement au niveau de la communication - Scheduler: Plannifier les demandes - ResourceMonitor: Donne l'accès aux ressources - ResourceProvisionService : Donne l'accès aux traitements sur les ressources - SecuritManager: Gère les privilèges

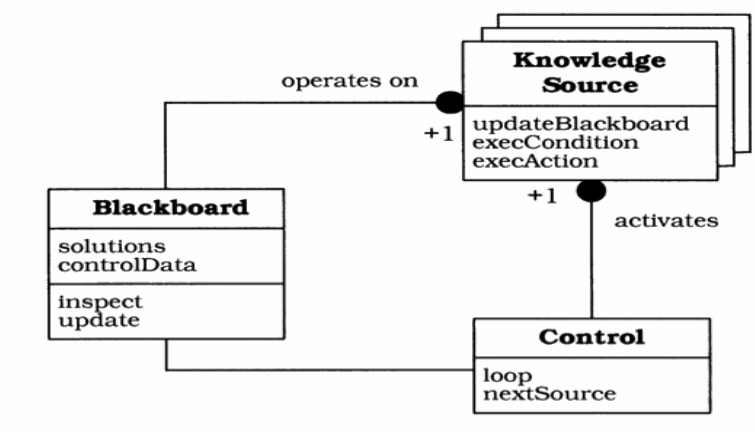
II.4.2.8 Event-bus

Tableau 2.34: Pattern Event Bus

Event-bus
Intention : Organiser le système afin de permettre l'appel d'un service sans référencement du système qui l'offre.
Indications d'utilisation. On utilise cette architecture dans un système dynamique où les modules peuvent être ajoutés ou supprimés selon le besoin.
Structure. 
Participants - EventBus : Définit l'interface pour accéder au service bus. - AgentEvent : Interface qui représente un événement - AgentEventListener : Observateur qui permet de diffuser l'événement - EventBusListener Classe qui implémente EventBus .

II.4.2.9 Blackboard

Tableau 2.35: Pattern Blackboard

Blackboard
Intention : Créer un système capable de sélectionner et sauvegarder des savoir-faire (service) approuvé.
Indications d'utilisation. Cette architecture peut être utilisée pour construire un système expert qui est capable de sélectionner des solutions fiables.
Structure. 
Participants - Blackboard : Gère les données communes et les présente un composant Control et sources de connaissances disponibles - Knowledge Sources : o Evalue l'état actuel de la progression de la solution o Produit des résultats sur la base de ses informations et l'écrit sur le Blackboard - Control : Coordonne, sélectionne les sources de connaissances à l'aide des stratégies prédéfinies .

Le modèle *Blackboard* est un modèle de conception, utilisé dans l'ingénierie logicielle, pour coordonner des systèmes distincts et distribués qui doivent travailler ensemble, ou en séquence, en priorisant continuellement les acteurs (ou les sources de connaissances).

Il est défini comme un modèle de conception comportementale car il affecte quand et comment les programmes réagissent et exécutent.

Le tableau se compose d'un certain nombre de magasins ou de « variables globales », comme un référentiel de messages auxquels on peut accéder par des processus autonomes séparés, susceptibles d'être physiquement séparés. Un « contrôleur » surveille les propriétés sur le *Blackboard* et décide quels acteurs (ou sources de connaissances) priorisent.

II.5 Conclusion

Notre objectif dans cette étude est de fournir un modèle d'application entreprise pour gérer de façon optimale un SI à partir de design pattern. La démarche consiste à comprendre les différents types des patterns existants pour construire une bibliothèque de solution et de modéliser un à un les composants du système.

On a spécifié deux catégories de patterns dans ce chapitre afin de définir leur domaine et la portée de leur solution. Il est indispensable de maitriser les patterns de *Gang Of Four (*Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides) pour modéliser nos composants logiciels et le Framework.

Mais ces composants seront inutiles si on n'arrive pas à trouver un moyen pour les organiser. Les patterns architecturaux nous permettent alors de choisir les combinaisons des architectures logicielles à mettre en place selon le besoin du système.

Les patterns décrits dans ce chapitre constituent le socle des autres patterns et l'ensemble de notre solution. Chaque composant ou entité que l'on va modéliser devrait faire référence d'au moins un de ces patterns.

Pour les deux prochains chapitres, on va continuer à spécifier des patterns (problèmes/solutions) qui vont agrandir notre catalogue de pattern. Ces nouveaux patterns font partie de business pattern qui sont liés au processus basic du SI.

CHAPITRE 3 : Business Pattern sur la gestion des ressources TIC

III.1 Introduction

Les ressources en matériels et les systèmes informatiques sont des investissements énormes au sein d'un SI. La disponibilité de ces ressources varie selon la capacité de l'organisation mais il y a un minimum de ressources nécessaire pour faire fonctionner correctement le SI.

Le gain obtenu par cette relation entre système d'information et ressources TIC est linéaire. Le nombre et la qualité des ressources mis en jeu et la fiabilité de service que l'on attend du système sont interdépendants. Le système devrait réutiliser autant que possible les services offerts par ses ressources pour prouver les investissements.

Notre objectif dans ce chapitre est de modéliser les ressources et les services offerts par le TIC qui est nécessaire pour faire fonctionner un SI. Cette modélisation consiste dans un premier temps de définir les métiers qui font appel à des outils TIC et dans second temps de concevoir une structure standard pour accéder à ses services.

Définition 3.1 :

Un service informatique est un service qui fonctionne ou fourni par des appareils électroniques et de réseau de communication dotés de capacité de traitements numériques.

Définition 3.2 :

Un pattern de métier est une description générique des règles de travail pour réaliser des solutions propres à un métier.

III.2 Service Informatique

Le traitement de l'information par un appareil électronique et de télécommunication englobe tout le type de service de SI en allant de l'acquisition jusqu'au pilotage :

- acquisition
- analyse et modélisation
- stockage et restitution
- communication et transport
- calcul et transformation
- visualisation et pilotage
- sécurisation

Toutes les chaines de traitement du SI (Paragraphe 1.2) nécessitent au moins l'utilisation d'un service informatique. Ces services sont souvent basic et brutes (ne contient pas de contrôle niveau métier) et nécessitent une plateforme organisationnelle et logicielle pour optimiser leurs interactions.

III.2.1 Acquisition

Ce service consiste à fournir des données numériques au SI et qui peuvent être obtenues de la manière suivante :

- manuellement : saisie, import, extraction
- automatiquement : capteurs, logiciel d'extraction, réseaux, système intelligent

Pour l'acquisition manuelle, les composants d'interface qui sont utilisés par les formulaires jouent un rôle important sur la nature et la qualité des données obtenues ainsi que la satisfaction des utilisateurs. Le processus défini par la figure 3.1 montre les différents filtres avant la sauvegarde des données :

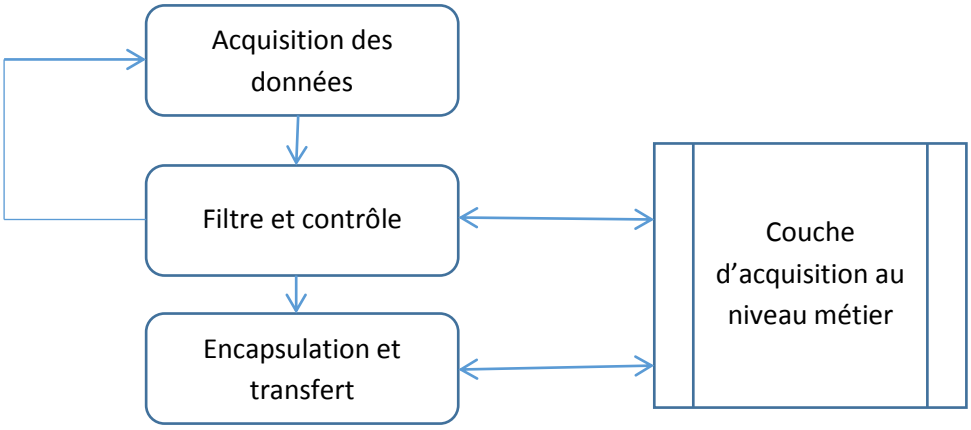


Figure 3.1 : Processus d'acquisition des données pour SI

III.2.1.1 Contrôle au niveau de l'acquisition

Généralement, on ne traite pas toutes les données directement. Chaque système d'acquisition contient un filtre qui permet de choisir de bonne candidate à envoyer au service métier pour être traitée :

- contrôle au niveau du terminal (élément principal vide, valeur non permise, etc.,)
- contrôle qui fait appel aux services

III.2.1.2 Adaptation au terminal

La présentation des données et le mode de transfert utilisé dépendent entièrement de la nature du terminal utilisé pour acquérir ces données :

- chaque terminal devrait avoir une interface unique dans la couche métier
- le choix d'un terminal est validé par sa capacité d'inclure les contrôles suffisant pour le système

III.2.2 Analyse et modélisation des données

Les données devront être stockées d'une manière permanente ou temporaire dans une source de données pour pouvoir les manipuler. Le modèle de cette source de données impose un large impact sur le mode de fonctionnement des services subséquents.

La modélisation et la structuration des données est un service quasiment manuel. La compréhension et la perception des données et ses contraintes permettent de définir la forme du conteneur qui va être utilisé pour stocker les données.

III.2.2.1 Niveau de modélisation des données

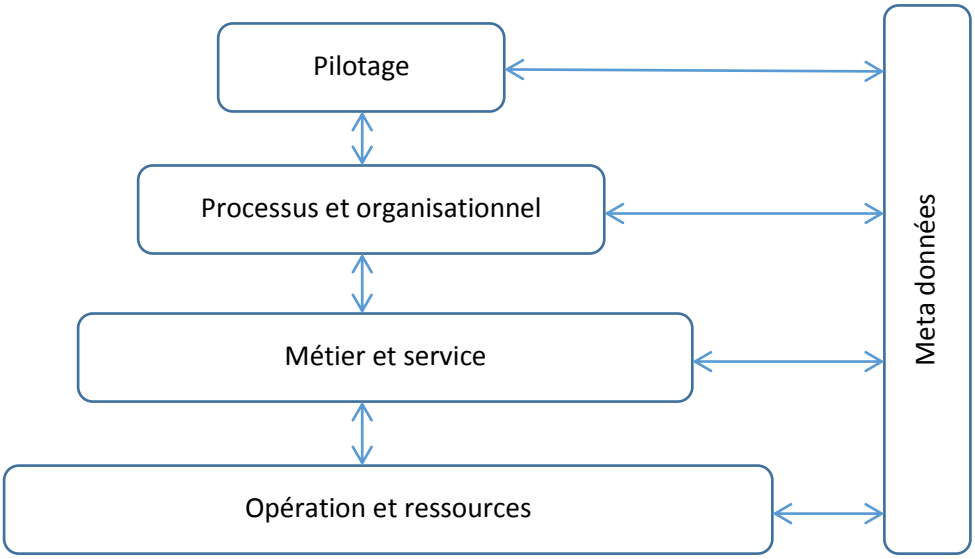


Figure 3.2 : Niveau de compréhension et modélisation des données

En se basant de flux d'information d'un SI, on a quatre niveaux de définition des données (Figure 3.2) :

- donnée de pilotage : spécifie les catégories des données nécessaires pour orienter l'action du système vers un objectif bien défini. C'est à partir des données contenues dans ce niveau qu'on constitue la liste des processus du système
- donnée organisationnelle et de processus : définit la structure du système du point vue organisationnelle et de traitement. Cette structure permet de stocker la forme du système à un moment donné par sa liste de processus, son organisation et ses ressources
- donnée métier et de service : définit la connaissance et le savoir-faire du système. Ce niveau de spécification de données permet de définir la bibliothèque de service disponible dans le système ainsi que leurs structures de données.
- donnée opérationnelle : spécifie les différentes évolutions de la donnée selon les traitements qu'elle a subis et les ressources qu'on a utilisées.

III.2.2.2 Meta données

Les services d'extraction et de transformation de données de la chaine de pilotage du SI ont besoin des données clés pour accéder à chaque niveau des données. La structure verticale Meta données permet de stocker les catalogues des données, les différentes sources de données et les indicateurs pour servir un point d'entrée au système selon le niveau désiré.

III.2.3 Stockage et restitution des données

Ce service est essentiel au cycle de vie d'un système. Le système tout entier repose sur le stockage des données acquises ou calculées et la restitution de ces informations.

Le volume des données d'un système d'information évolue exponentiellement selon le rythme de son utilisation. Ce service nécessite des ressources physiques importantes pour s'assurer la pérennité du système.

III.2.3.1 Support physique des données

Techniquement, les données numériques sont toujours stockées dans un fichier. La gestion du fichier et le mode d'accès séparent les types de support des données :

- source simple: gérer par le système d'exploitation
- source de données relationnelles : gérer par un SGDB
- source de données partagées : gérer par un serveur de fichier
- source de données intelligentes : gérer par un système autonome

III.2.3.2 Serveur des données

Un serveur de données est une ressource qui encapsule les traitements suivants :

- gestion de structure des données
- service de stockage de données
- gestion des accès
- gestion des transactions
- archivage

Chaque source de données implémente ses propres règles pour gérer les données et les transactions avec le système. Pour accéder à ses ressources, le système doit implémenter des interfaces capables d'interagir avec chaque type de serveur.

III.2.4 Communication et transport

Le SI est un système distribué qui véhicule des données dans des réseaux interconnectés. Ces services et technologies permettent au système d'acheminer les informations vers chaque terminal et de contrôler l'ensemble via un seul point : le noyau du SI.

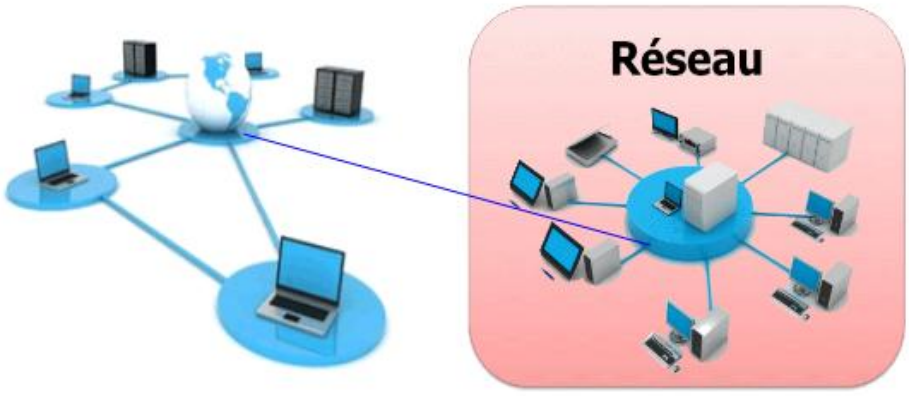


Figure 3.3 : Support de partage et de communication sur réseau

Les services de communication et de transport (Figure 3.3) sont sensibles aux intrusions et aux attaques. Le système qui utilise ce service devrait ajouter d'autre couche de sécurité afin de personnaliser le mode de communication.

III.2.4.1 Réseau informatique

Le modèle de traitement distribué défini dans cette étude ne fonctionne pas sans un support réseau. L'infrastructure et la technologie utilisées par le réseau informatique sont complètement transparentes lors de la modélisation d'un SI mais pendant l'implémentation, son influence agrandit selon le rythme d'utilisation et d'installation des services.

La relation entre le noyau du système et le réseau informatique est dictée par les transactions :

- la sécurité
- la disponibilité

Dans le cas pratique, cette relation apparait sur l'organisation du système au niveau géographique (subdivision par zone) et au niveau fonctionnel (subdivision par traitement).

III.2.4.2 Protocole de la couche application

Le noyau du SI et réseau définissent leurs interactions spécialement au niveau de couche d'application du réseau informatique. Avec le concept de réutilisation, l'intégration des technologies TIC dans le système se traduit par l'utilisation de son service (applicative) sans pour autant se soucier de son système.

Parmi les protocoles de la couche application, les protocoles qui implémentent des services utilisateurs sont les cibles de nos patterns :

- service de transfert de fichiers : FTP, NFS, SMB, etc.
- service de messagerie : SMTP, POP, IMAP
- service de communication client/serveur : HTTP
- service annuaire : LDAP

III.2.5 Calcul et transformation

Les données stockées dans un SI sont modélisées sous plusieurs formats selon les technologies disponibles et le choix de l'analyste. Ce service offre au système le moyen de fournir d'autres données sous un autre modèle et format après un calcul et une conversion.

La compatibilité des systèmes (outils) interconnectés nécessite la transformation des données de part et d'autre des bouts de ligne de communication. Le système a besoin d'un service de conversions susceptible de répondre à ces besoins diverses d'adaptation.

Les métiers de base pour ces services sont :

- conversion (format, conteneur, codage, langue)
- triage
- filtrage
- évaluation (arithmétique, agrégation, statistique)
- planification (temps, espace, ressources)

III.2.6 Visualisation et pilotage

Les acteurs du SI interagissent avec le système à l'aide de ce service. Il est souvent associé au service acquisition des données et ils partagent les mêmes composants de présentation.

L'interface homme machine est en constante évolution, il dépend de la technologie disponible, la culture et le niveau d'adaptation de l'homme.

L'implémentation de ce service dans le SI occupe environ 80% de ressources logicielles du système.

Ce service utilise plusieurs types de ressources pour assurer la communication entre le système et l'acteur :

- dispositif d'entrée : capteur (position, direction vitesse, tactile, optique, etc.), terminal mobile
- dispositif de sortie : écran, hologramme, optique, son, imprimante, terminal mobile

L'interaction en temps réel entre le système et l'acteur combine ses dispositifs via une interface graphique offerte par le système. La diversité des versions et de type de dispositif rend encore plus difficile la conception et l'adaptation des interfaces graphiques aux ressources TIC existantes.

III.2.7 Sécurisation

La plupart des composants TIC offre un moyen pour sécuriser l'accès à son service. Cette fonctionnalité devient systématique pour le cas de stockage des données, gestion des transactions et les formulaires d'interaction.

Comme l'implémentation de ce service est indépendante au système (dépendant de la nature de ressource TIC), elle ne garantit pas la sécurisation efficace du SI.

Exemple :

- l'implémentation du service LDAP pour authentifier les utilisateurs dans un réseau local n'empêche pas les comptes internes de chaque machine de s'authentifier localement.
- La mise en place d'un firewall ne peut pas empêcher un Cheval de Troie d'utiliser les canaux ouverts pour se communiquer avec l'extérieur.

III.3 Pattern Adapter aux ressources TIC

L'utilisation directe des ressources et des services TIC dans le SI induit une dépendance absolue de ce dernier. Pour assurer la compatibilité du système avec la plupart des appareils et technologie disponibles, on doit créer des interfaces qui contiennent le maximum de propriétés et des méthodes du service attendu.

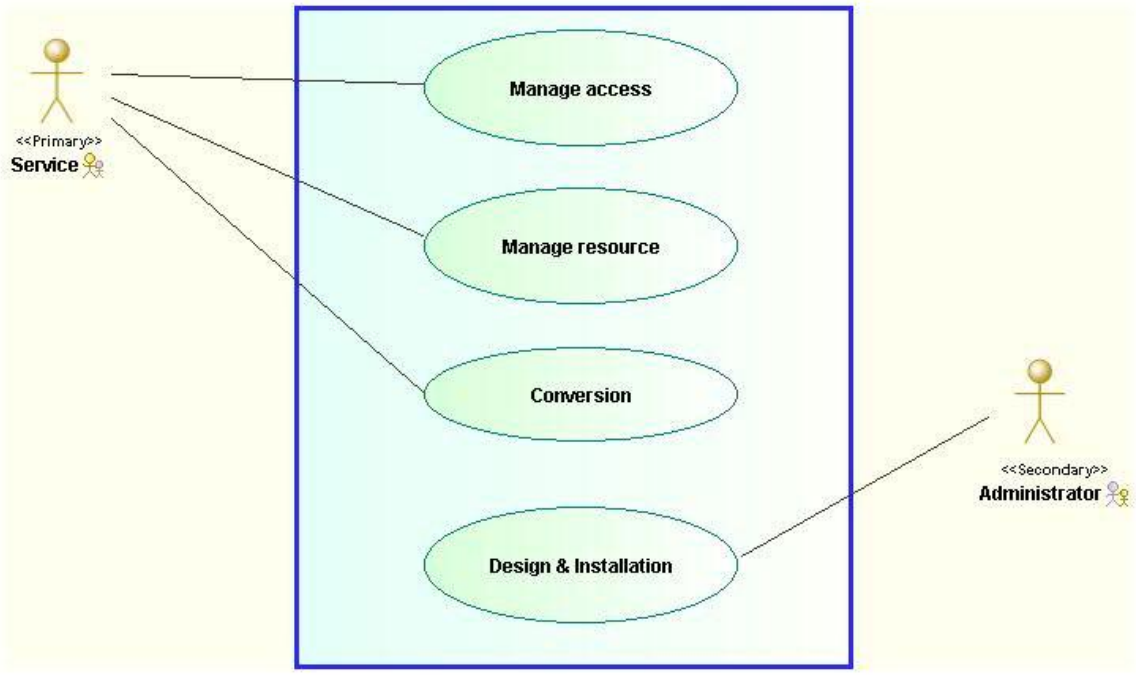


Figure 3.4 : Diagramme de cas d'utilisation des ressources TIC

Le principal acteur qui utilise les ressources TIC est le service métier (Figure 3.4) du système. Cette utilisation se déroule en général en trois étapes :

- l'accès aux ressources : établissement de la connexion, authentification
- la gestion des ressources : paramétrage et configuration, gérer les transactions
- adaptation des données : conversion des formats de données

III.3.1 Pattern sur l'IHM

L'interface homme machine est un système complexe, la compréhension et l'interprétation de l'homme d'un graphe ou d'une forme varie selon sa culture et de ses connaissances.

Mais avec l'apparition de concept Windows et la standardisation des écrans tactiles des terminaux mobiles (exemple disponible sur la figure 3.5), la modélisation de cette couche du système devient de plus en plus accessible.

Figure 3.5 : Exemple d'IHM

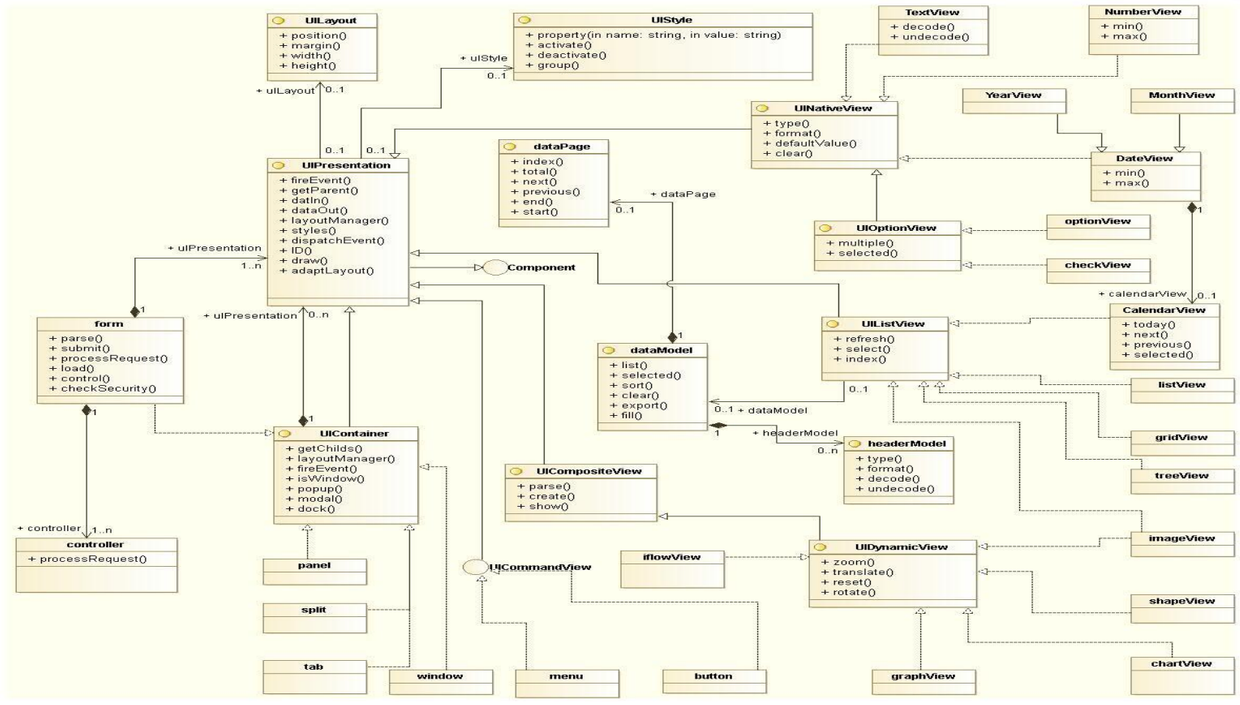
Le pattern IHM consiste à modéliser l'interaction du système avec les composants graphique standard :

- éléments du formulaire de saisie (texte, nombre, liste, page, tableau, etc.)
- graphes
- Shapes (forme géométrique planes)

III.3.1.1 Graphical User Interface 2D

Avec l'apparition du pattern MVC, la plupart de technologie de présentation des données et d'interaction sépare la vue de celui des données.

Tableau 3.1: Pattern GUI 2D

GUI 2D
Intention : Créer un système capable de présenter des informations
Indications d'utilisation. Cette structure peut être utilisée pour afficher ou saisir des données sur un écran 2D
Structure. 

Participants principaux
- UILayout : Définit la règle de positionnement de chaque composant
- UIPresentation :
  o gérer les interactions avec l'utilisateur
  o interface connue par le contrôleur
- UIContainer : classifie les composants selon leur parent
- dataModel : source des données temporaires
.

L'affichage d'un composant graphique est défini par :

- sa nature
- son conteneur (emplacement et nature)
- son style (liste des propriétés)
- sa structure de données
- sa liste d'action (méthode)
- ses événements

III.3.1.2 Office

Les suites bureautiques sont des alternatives de formulaire pour présenter et traiter les données. L'autonomie de ces applications vis-à-vis du système présente à la fois des avantages et des inconvénients :

- avantage : portable, accessible indépendamment du réseau et du système, convivial et familier
- inconvénient : nécessite d'autre application pour l'import-export des données

Pour pallier à ce problème de migration des données, le pattern de conversion permet de créer des interfaces qui puissent interconnecter ces outils aux systèmes.

III.3.2 Pattern sur les ressources numériques

Les données manipulées par un système numérique sont catégorisées en trois types :

- données de contrôle
- données d'identification
- données à traiter

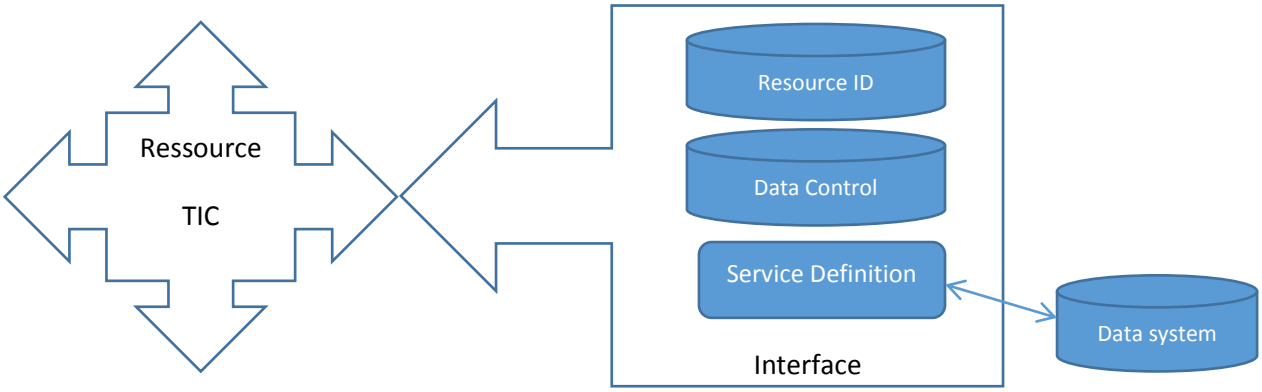


Figure 3.6 : Interface d'accès aux ressources TIC

Chaque composant TIC définit son modèle de données en combinant ces types de données dans son système (Figure 3.6). L'interface qui utilise la même combinaison de données peut se communiquer avec le système du composant.

Les patterns sur les ressources numériques consistent à modéliser les différentes formes des données nécessaires pour relier le système avec la ressource et de spécifier les services de base offerts par le composant TIC.

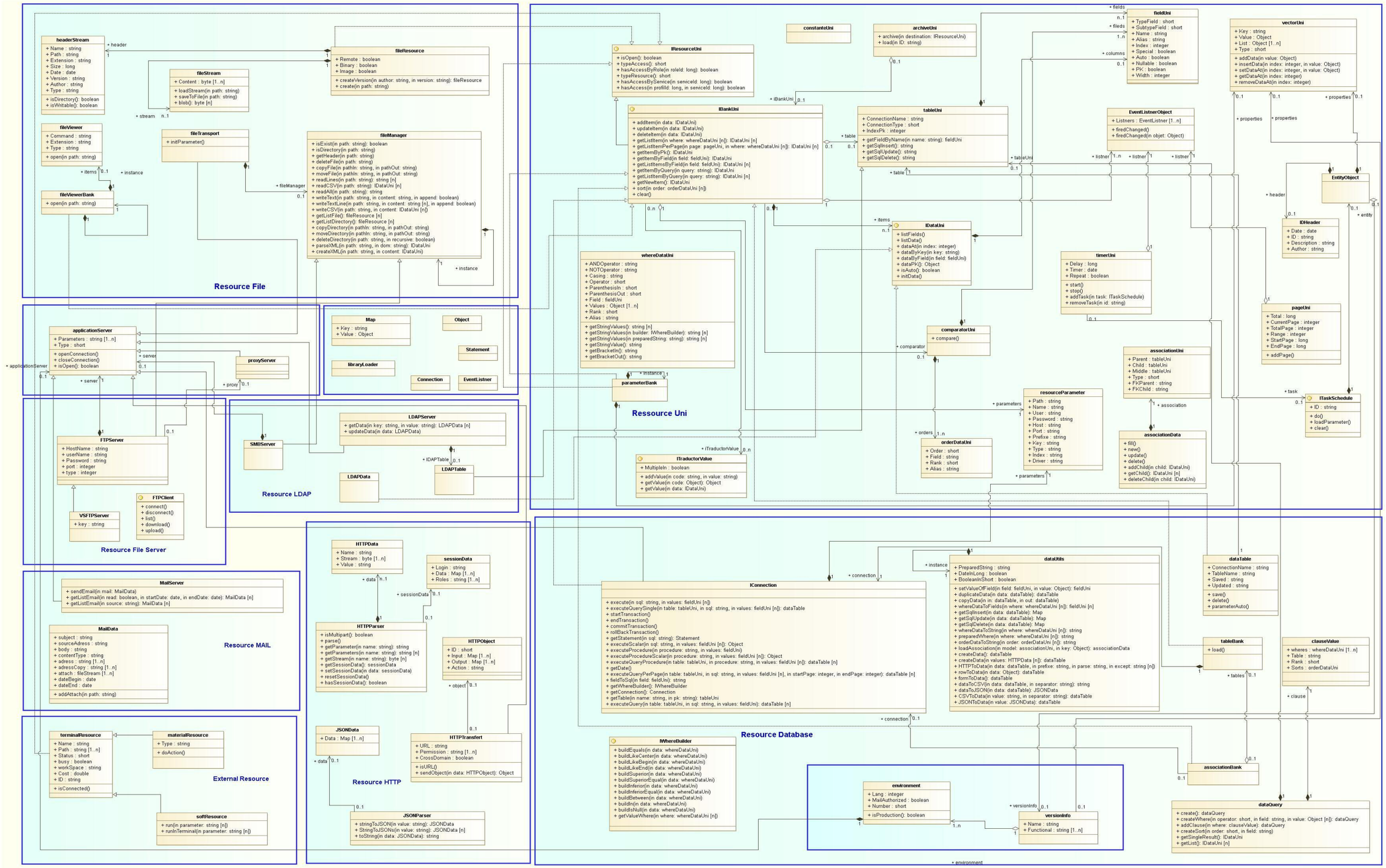


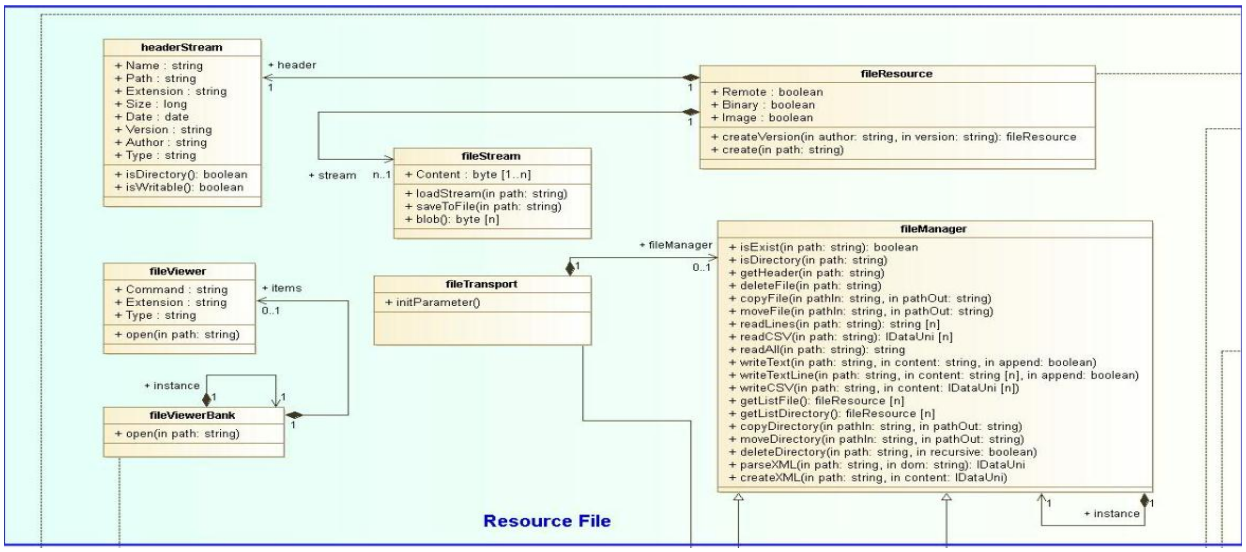
Figure 3.7 : Diagramme de classe des ressources TIC

Le bloc « Resource Uni » définit un modèle standard pour toutes les ressources tandis que les sept autres blocs dans ce diagramme définissent les interfaces pour accéder aux ressources des principaux composants TIC :

- ressource fichier : pour accéder aux fichiers d'un système d'exploitation directement
- ressources données relationnelles pour accéder aux données d'une base de données
- ressource liée à serveur FTP : pour utiliser le service FTP
- ressource liée à un serveur http : pour utiliser le service client serveur
- ressource liée à un serveur Mail : pour envoyer et recevoir d'email
- ressource liée à un serveur annuaire : pour accéder aux annuaires de l'organisation
- ressource liée à un autre type traitement externe : pour lier le système à d'autres matériels ou application

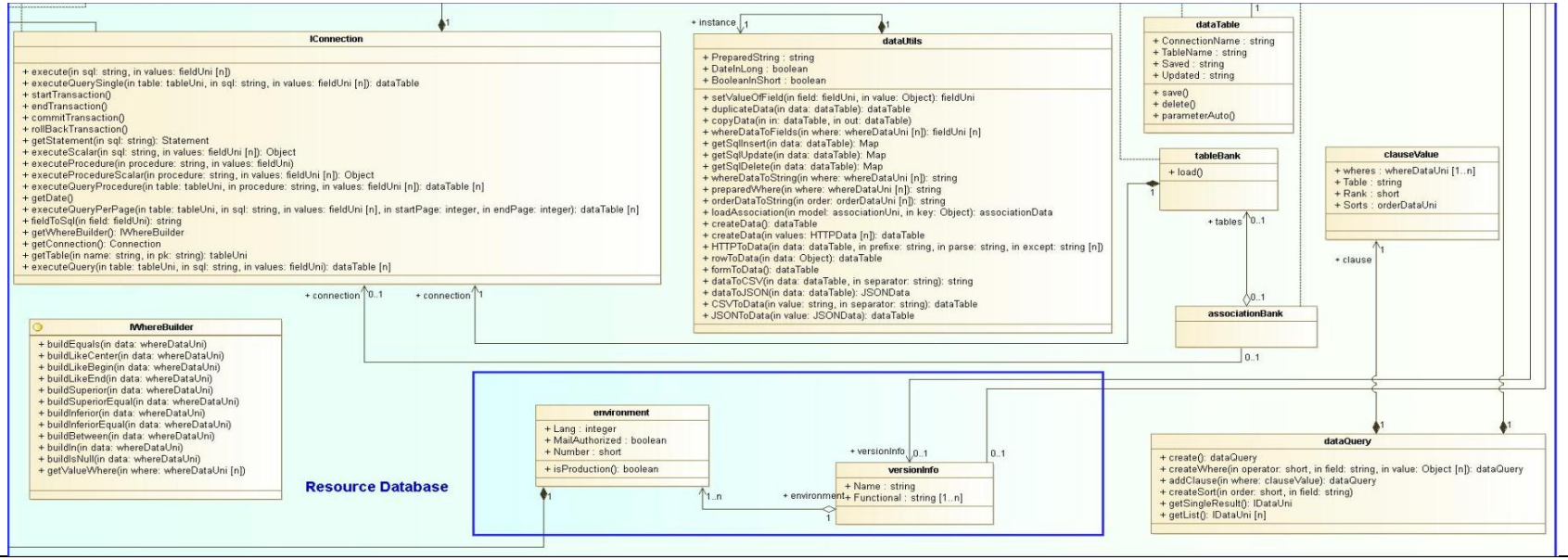
III.3.2.1 Ressource fichier

Tableau 3.2: Pattern File Resource

File resource
Intention : présenter les données et les interfaces pour manipuler les fichiers.
Indications d'utilisation. Service de conversion, dataModel d'un IHM.
Structure 
Participants principaux - fileManager : Service qui manipule le fichier - fileStream : contenu du fichier - headerStream : identification du fichier.

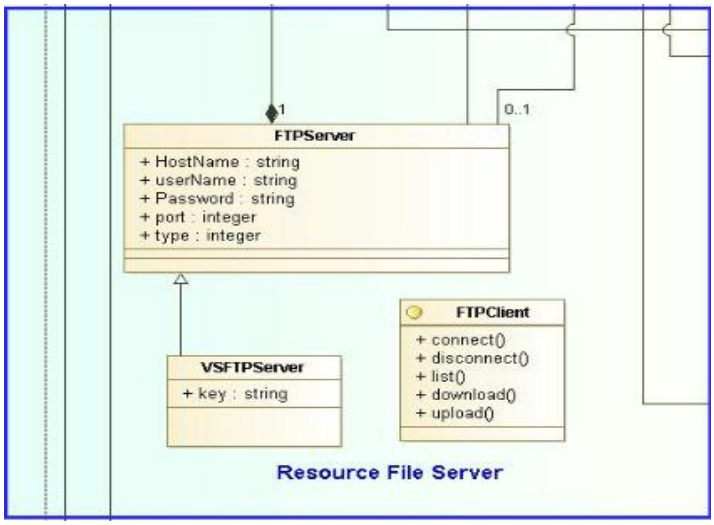
III.3.2.2 Ressources données relationnelles

Tableau 3.3: Pattern GUI 2D

Database Resource
Intention : présenter les données et les interfaces utilisées pour manipuler des données relationnelles.
Indications d'utilisation. Service de conversion, dataModel d'un IHM.
Structure 
Participants principaux - Iconnection : service qui manipule les données sur une connexion avec une base de données - dataTable : un enregistrement - tableUni : structure d'une table - fieldUni : structure d'un champ

III.3.2.3 Ressource liée à un serveur FTP

Tableau 3.4: Pattern File server *resource*

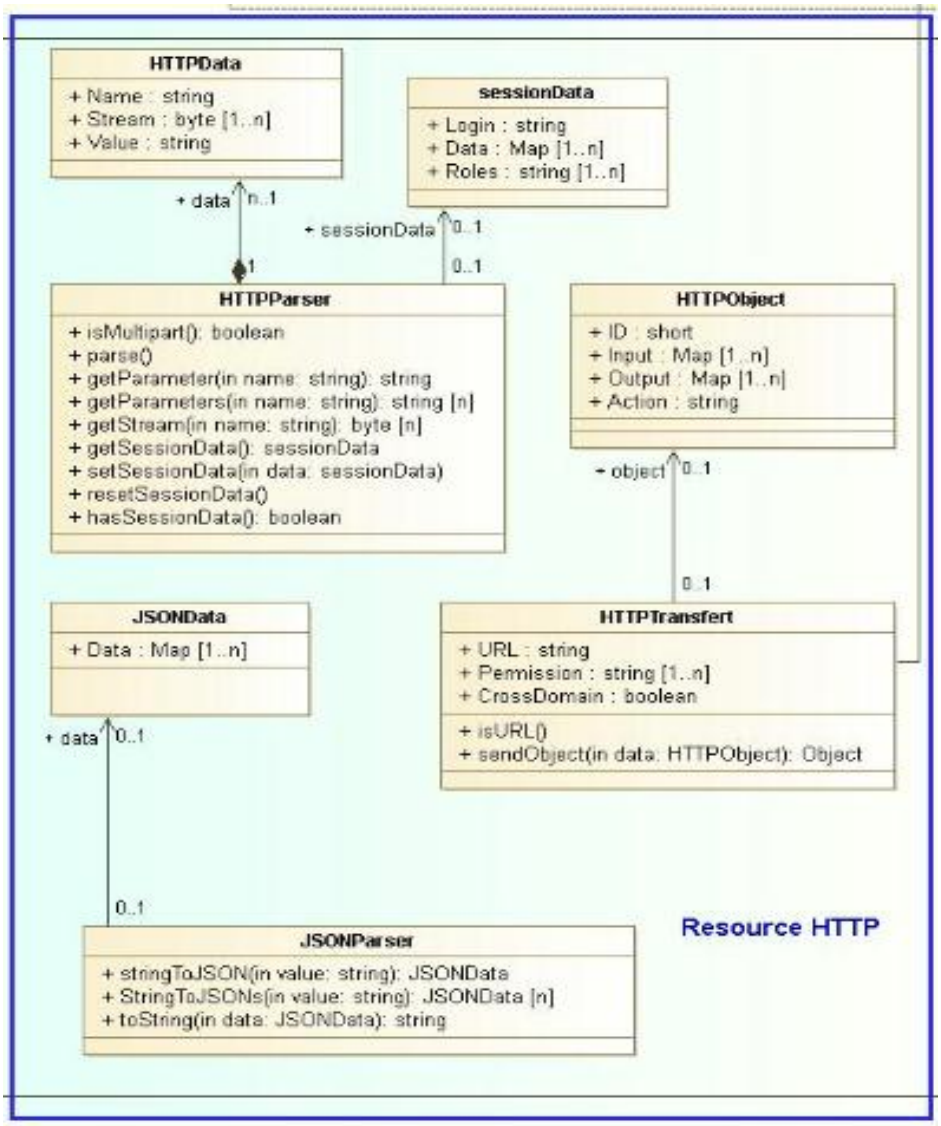
File Server Resource
Intention : présenter les données et les interfaces utilisées par un serveur de transfert de fichier.
Indications d'utilisation. Service de traitement.
Structure 
Participants principaux - FTPServeur : identifiant du serveur - fileStream : fichier à transférer - FTPClient : service qui manipule les connexions

Actuellement, on a plusieurs versions de technologie de transfert de fichier : SFTP, FTPS, FTP, etc. Ce Pattern permet d'offrir une interface pour représenter les services communs de ces technologies afin qu'on puisse les adapter dans le Framework :

- lister les fichiers disponibles
- télécharger un fichier
- mettre à jour un fichier distant
- envoyer un fichier vers le serveur

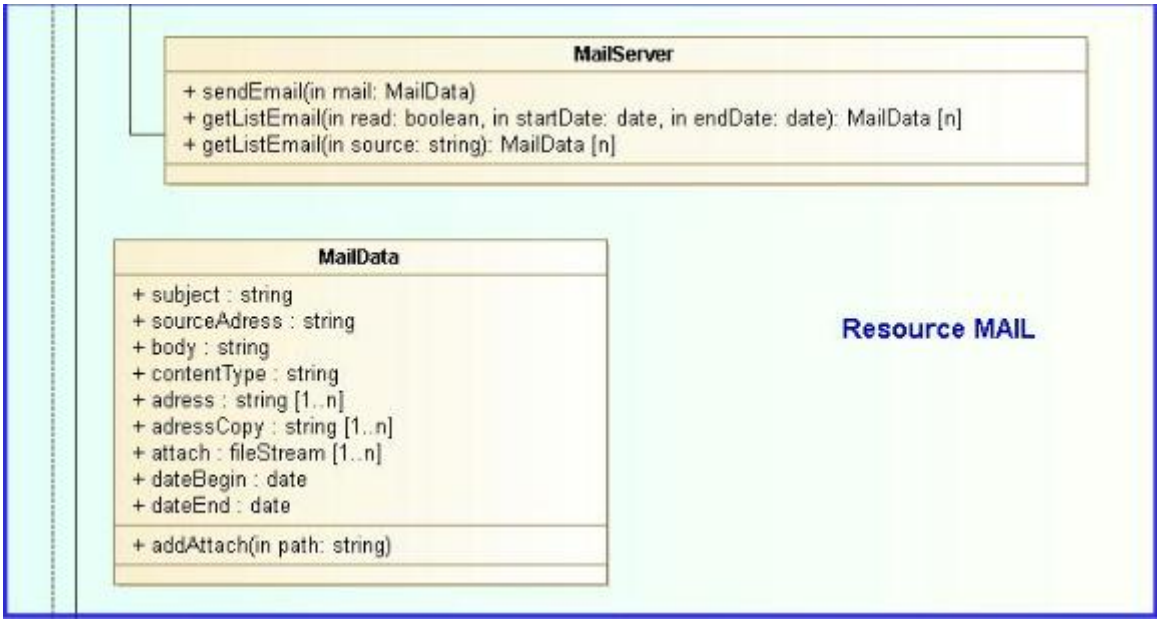
III.3.2.4 Ressource liée à un serveur HTTP

Tableau 3.5: Pattern HTTP *resource*

HTTP Resource
Intention : présenter les données et les interfaces utilisées par un serveur HTTP.
Indications d'utilisation. Service de traitement, contrôleur d'un IHM.
Structure 
Participants principaux - HTTPTransfert : manipule les transactions - HTTPParser : rétablir les données reçues - HTTPData : données à sérialiser et à transférer - JSONData : une autre structure de HTTPData

III.3.2.5 Ressource liée à un serveur Mail

Tableau 3.6: Pattern Mail *resource*

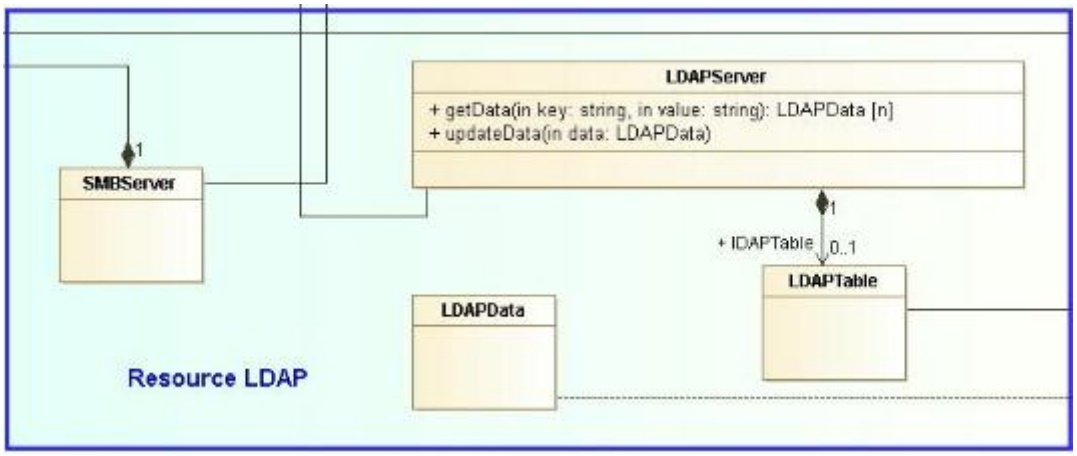
Mail Resource
Intention : présenter les données et les interfaces utilisées pour commander un serveur d'envoi et réception d'email
Indications d'utilisation. Service de conversion, dataModel d'un IHM
Structure 
Participants principaux - MailServer : service qui manipule la connexion au serveur mail - dataMail : présente les données contenues dans un email

Le courrier électronique est considéré comme étant le service le plus utilisé sur Internet. Pour envoyer et recevoir (prendre les mails dans le serveur), la version courante de service de messagerie utilise trois protocoles :

- envoie : SMTP (Simple Mail Transfer Protocol) est le protocole standard permettant de transférer de courrier d'un serveur à un autre avec une connexion point à point.
- réception : POP3 (Post Office Protocol), IMAP (Internet Message Access Protocol) est un protocole alternatif au protocole POP3 mais offrant beaucoup plus de fonctionnalités

III.3.2.6 Ressource liée à un serveur annuaire

Tableau 3.7: Pattern LDAP Resource

LDAP Resource
Intention : présenter les données et les interfaces utilisées pour exécuter de requête dans le serveur annuaire
Indications d'utilisation. Service d'authentification, dataModel d'un IHM
Structure 
Participants principaux - LDAPServer : service qui manipule la connexion avec le service d'annuaire - LDAPData : présente les données d'un nœud dans l'annuaire - LDAPTable : définit la structure des données à échanger avec le serveur

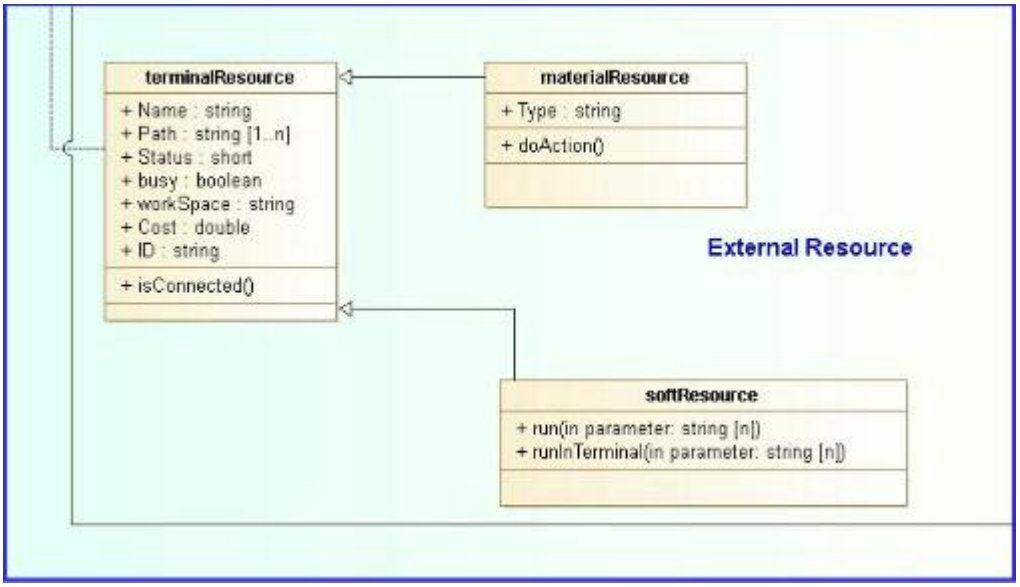
Les annuaires LDAP (Lightweight Directory Access Protocol) se situent au cœur des fonctions de communication et de collaboration de l'entreprise à travers son intranet car ils en simplifient la gestion et l'administration.
La mise en œuvre d'un annuaire LDAP au sein du système d'information apporte une gestion optimale des utilisateurs et de leurs profils, des ressources, et la possibilité de partager ce référentiel avec l'ensemble.

Le protocole d'échange LDAP comprend neuf opérations élémentaires :

- Les opérations d'interrogation : recherche et la comparaison
- Les opérations de mise à jour : ajout, suppression, modification et changement de nom
- Les opérations d'identification et de contrôle de la session : ouverture de la session et l'identification, fermeture de la session, abandon d'une requête en cours

III.3.2.7 Ressource liée à un autre type traitement externe

Tableau 3.8: Pattern External resource

External Resource
Intention : présenter les données et les interfaces utilisées pour commander un serveur d'envoi et de réception d'email
Indications d'utilisation. Service de conversion, dataModel d'un IHM
Structure 
Participants principaux - terminalResource : présente l'identifiant d'une ressource externe - materialResource : service qui simule l'utilisation d'un matériel - softResource : service qui lance l'exécution d'une autre application

III.3.3 Pattern de conversion

Les différentes couches du système (interne ou externe) traitent les données selon leurs règles. Pour passer d'une couche à l'autre, on a besoin souvent de redéfinir la structure des données. Les services de conversion dont le diagramme de classe est défini par la figure 3.8 permettent de converger en un seul point du système toute opération de conversion pour faciliter la maintenance et l'intégration d'une nouvelle structure.

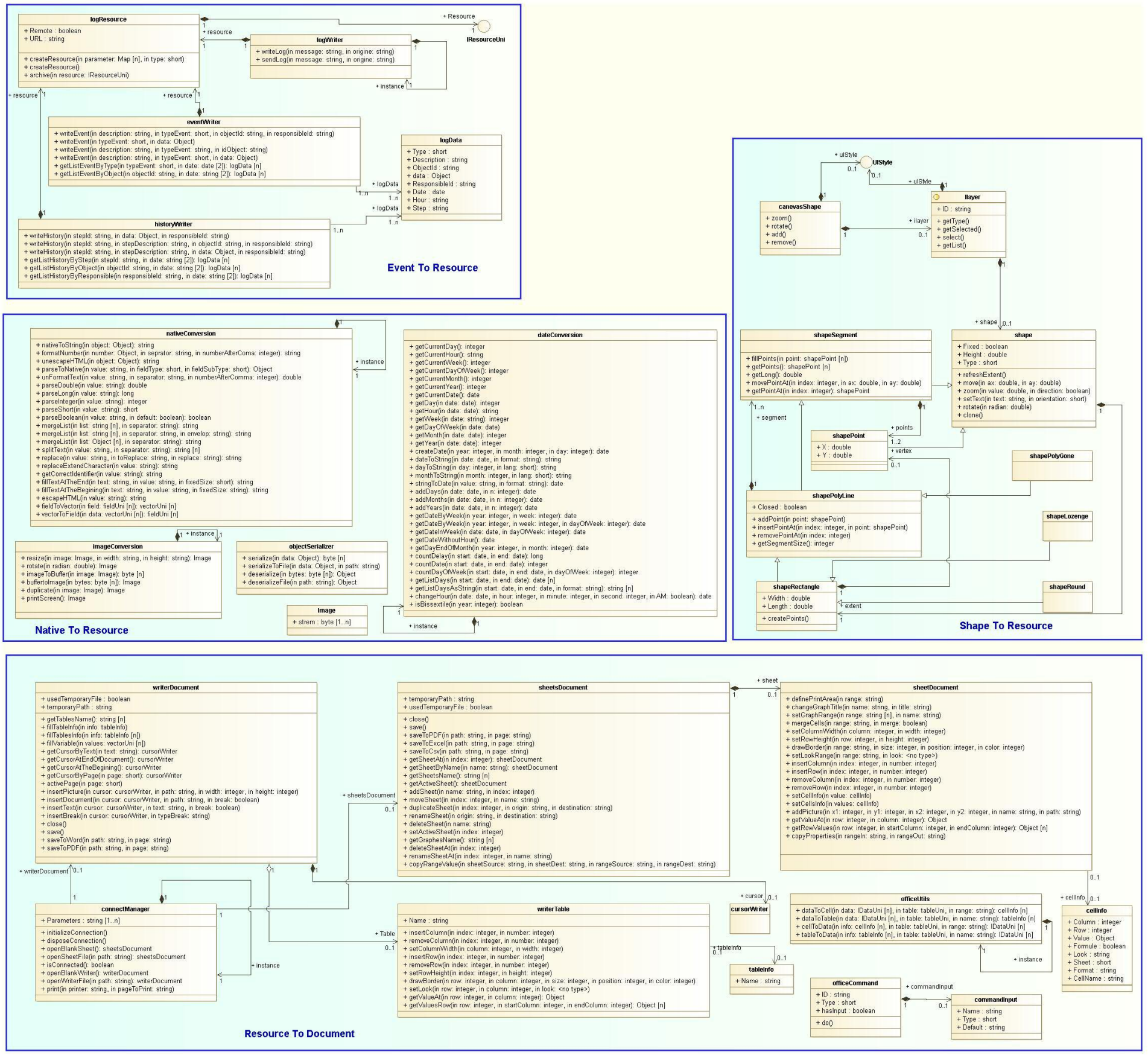
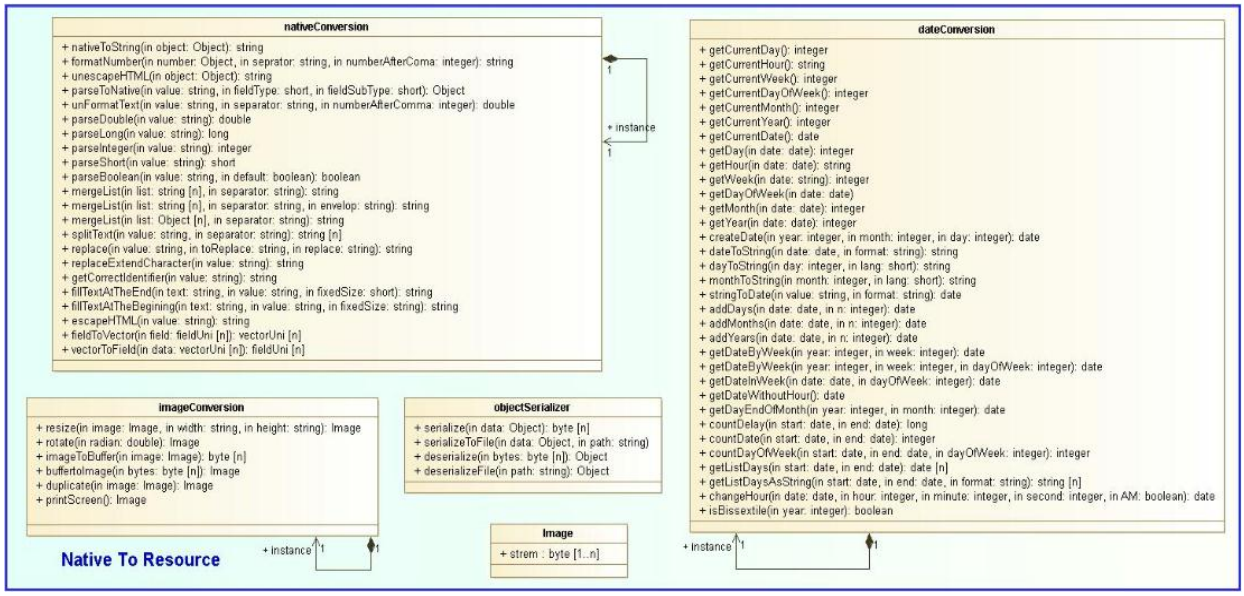


Figure 3.8 : Diagramme de classe des services de conversion des ressources TIC

III.3.3.1 Service de conversion des données primitives

Chaque plateforme de développement possède des déclarations des données primitives propres à son langage. Ce service permet de traiter ces types de données.

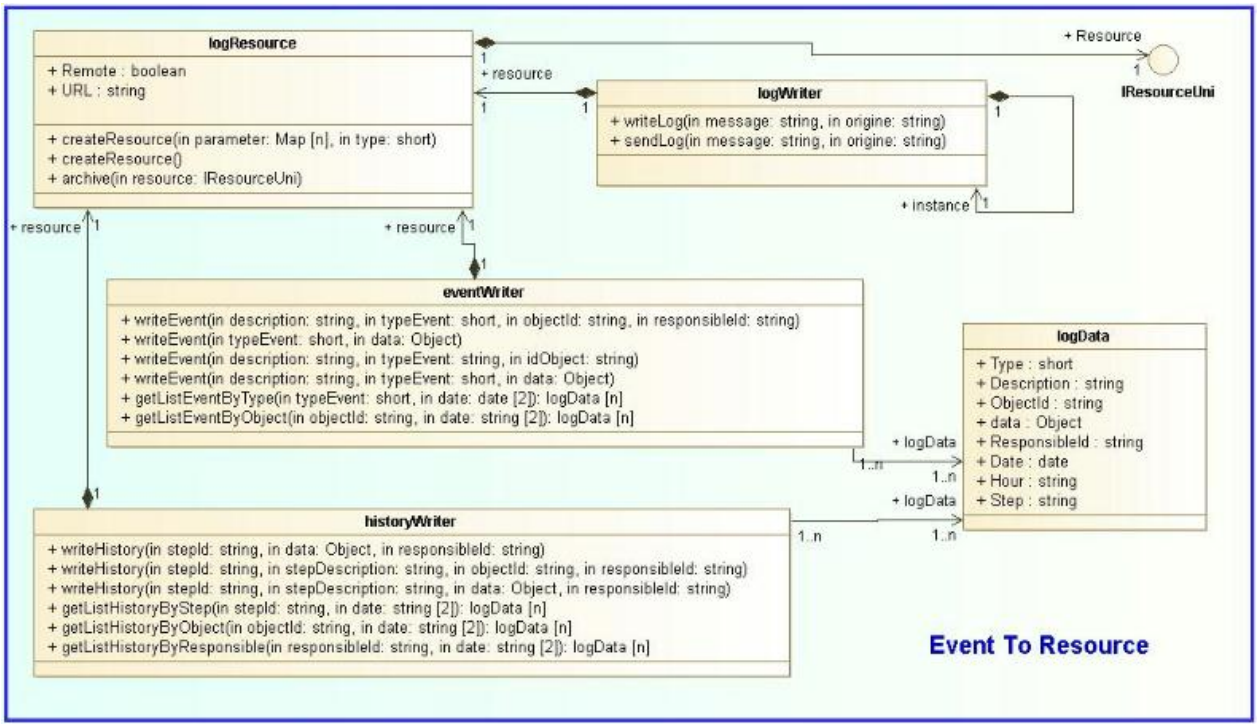
Tableau 3.9: Pattern Service conversion

Service native conversion
Intention : Convertir le format des données ou extraire une partie de données pour en créer d'autres.
Indications d'utilisation. Service de traitement, dataModel d'un IHM
Structure 
Participants principaux - NativeConversion : service qui manipule les données liées à la plateforme de développement - ImageConversion : service qui manipule les objets images - DateConversion : service qui manipule les objets dates - ObjectSerializer : service qui manipule les objets sérialisables

III.3.3.2 Service de stockage d'événement

La gestion des évènements est un point essentiel dans le cycle de vie des données et des traitements. Ce service permet de stocker les données relatives à un évènement dans un serveur de fichier ou de base de données.

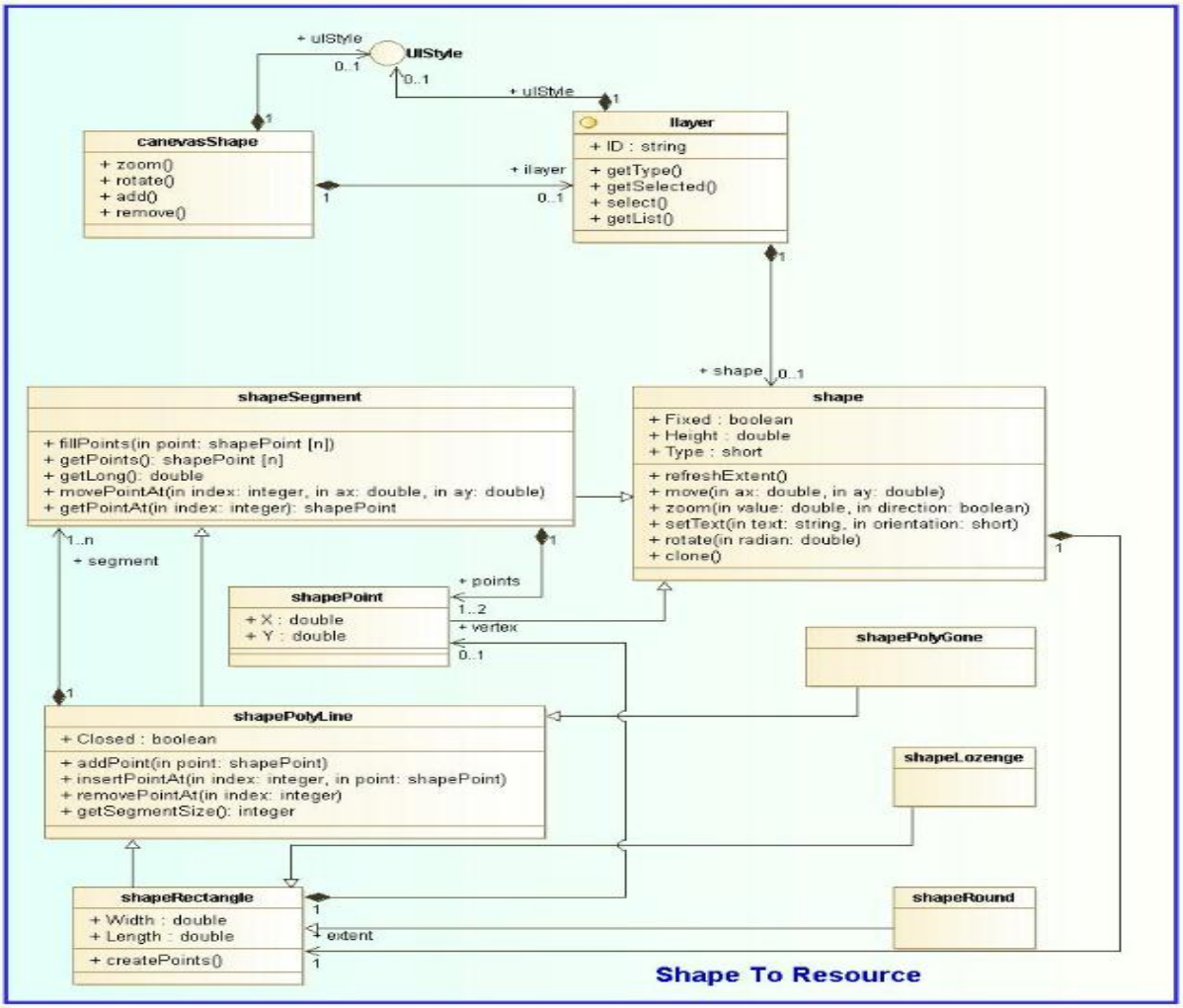
Tableau 3.10: Pattern Service *event conversion*

Service event conversion
Intention : Convertir les événements du système en données.
Indications d'utilisation. Service
Structure 
Participants principaux - LogData : présente les données équivalentes à un événement - LogWriter : service qui trace les événements liés à des algorithmes et codes - EventWriter : service qui trace les traitements (processus) - historyWriter : service qui trace les actions d'un utilisateur

III.3.3.4 Service de traitement des informations vectorielles

Une information vectorielle est basée sur une présentation graphique d'un ensemble de point. Le service Shape conversion permet de convertir des coordonnées en images vectorielles de deux dimensions et vice versa.

Tableau 3.11: Pattern Service Shape conversion

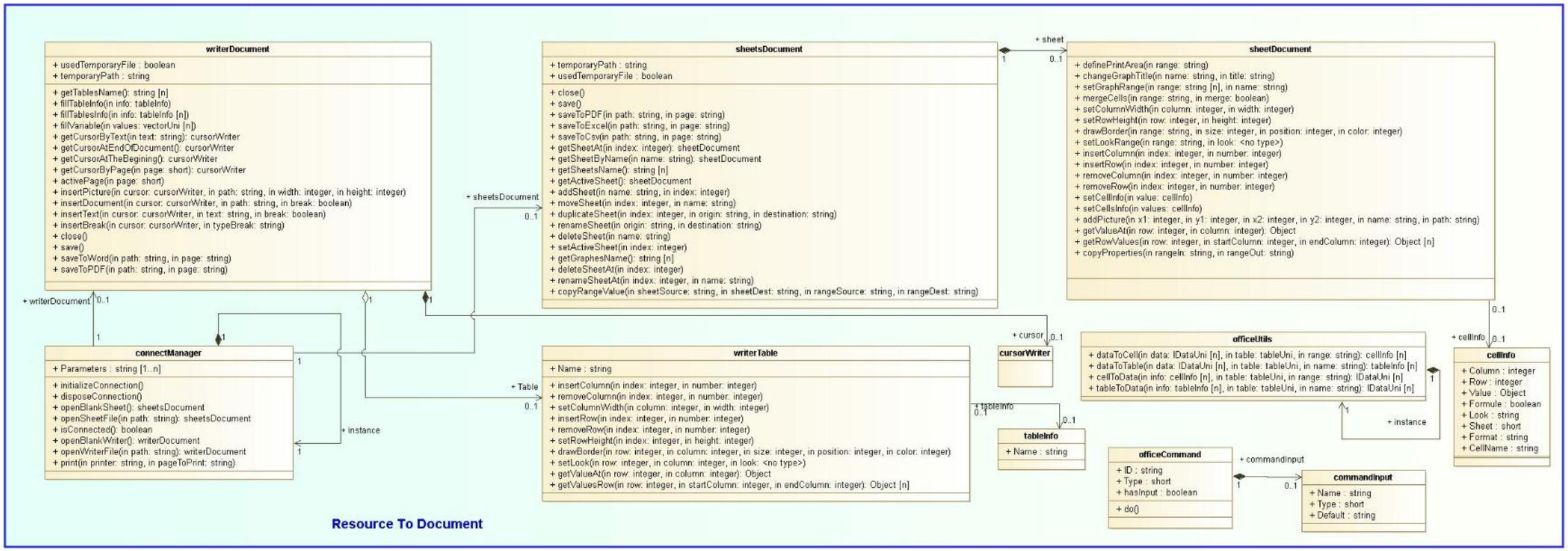
Service Shape conversion
Intention : gérer les données vectorielles.
Indications d'utilisation. Service de conversion, dataModel d'un IHM
Structure 
Participants principaux - canevasShape : service qui assure l'affichage des informations - Ilayer : présente une couche des données de même type - shape : présente la structure d'une information

III.3.3.5 Service de traitement des documents bureautiques

Le format de document Word et Excel sont les plus utilisés pour saisir et échanger des informations. Ce service permet de manipuler des modèles de documents pour en extraire des données ou pour en remplir.

Tableau 3.12: Pattern Service document conversion

Service document conversion
Intention : échanger des données avec des modèles documents
Indications d'utilisation. Service d'import et export
Structure



Participants principaux

- writerDocument : manipule les données sur modèle de format Word
- sheetsDocument : manipule les données sur un modèle de format Excel
- connectManger : service qui gère la connexion avec un document
- cellInfo : présente une structure de donnée équivalente à une cellule
- tableInfo : présente une structure de donnée équivalente à un tableau

III.4 Dépendance avec les versions

Chaque composant d'un système d'information évolue avec le temps. Les éléments TIC n'échappent pas à cette règle ; un composant électronique ou un logiciel devient inutile ou obsolète après quelque cycle de vie et cet événement ne devrait pas impacter le bon déroulement du système.

Pour faire face à cet ensemble des contraintes, on doit :

- l'utilisation au maximum des interfaces (au lieu de classe) dans le modèle pour faciliter l'intégration d'une nouvelle version
- la distribution des traitements par des services : un traitement susceptible de changer dans le temps et indépendant de version de la plateforme et des ressources nécessite une création de service
- l'utilisation de serveur de code source et de librairies : les services puissent continuer d'utiliser des anciennes versions de Framework

III.5 Conclusion

Les acteurs du SI agissent sur le système en utilisant des ressources matérielles et logicielles munies des interfaces homme machine. L'ensemble de ces ressources constituent à la fois la porte et le fondement du système tout entier.

En factorisant les services offerts par chaque composant et les contraintes techniques de traitement numérique, on a défini le modèle du socle du système qu'on va bâtir. Il est spécialement constitué de structure de données figées et des services de traitement de base ouverts (peut s'adapter à des systèmes externes).

A l'aide de cette structure et un cahier de charge bien défini, on peut construire des applications robustes mais la réutilisation s'arrête uniquement au niveau des composants logiciels.

Pour augmenter le degré de la réutilisation des éléments du système, on va redéfinir d'abord le système de gestion des processus et ses patterns et après spécifier techniquement le modèle et la structure de la chaine opérationnelle et chaine décisionnelle de notre outil de gestion de système d'information.

CHAPITRE 4 : Business Pattern sur la gestion de processus métier

IV.1 Introduction

Le processus métier offre une vision de haut niveau de mode de fonctionnement de l'organisation. Le processus représente un langage graphique pour décrire la séquence de traitement dans un système complexe et la plupart des acteurs (interne et externe) sont capables d'interpréter la logique d'un processus à cause de sa présentation claire et limitée.

Cet aspect simpliste de processus nous permet de définir des patterns sur le pilotage et la gestion d'une entreprise via ses métiers.

Le management par processus a pour objectif de s'assurer que le système utilise et gère ses ressources pour réaliser ses orientations stratégiques. Cette solution offre aux organisations la possibilité d'accroitre leur productivité, de saisir plus vite des opportunités sur le marché et d'être plus agile et flexible afin de mieux faire face à l'instabilité de l'environnement.

IV.2 Processus métier

Définition 4.1 :

Un processus est un ensemble d'activités organisées dans le temps qui transforme des éléments d'entrée en éléments de sortie (Figure 4.1).

Un processus métier est un enchaînement d'actions réalisées par différents acteurs collaborant pour délivrer un résultat tangible et une valeur ajoutée métier pour l'entreprise.

Figure 4.1 : Processus métier

Selon la structure fonctionnelle du SI (Figure 1.16), on a quatre catégories de processus :

- processus de pilotage ou de management qui ont pour but d'organiser les objectifs stratégiques.
- processus opérationnels (métier) qui ont pour fonction d'accomplir une mission dans un domaine donné et utilise plusieurs fonctions.
- processus de support qui sont périphériques au métier de l'entreprise et ne participent qu'indirectement à l'accomplissement d'un objectif métier.
- processus de mesure qui fournissent les métriques nécessaires à l'évaluation des processus et à leur amélioration continue

IV.2.1 Activité

Définition 4.2 :

Une activité est un ensemble de tâches identifiables du processus aux entrées et sorties clairement définies et dont la valeur ajoutée est mesurable.

IV.2.2 Modélisation de processus

Un processus métier doit être encadré, de façon à le positionner dans une vision métier globale au sein du SI :

- l'évènement déclencheur,
- le (ou les résultats) attendus,
- les objectifs poursuivis

La Figure 4.2 présente l'ordre chronologique dans lequel les différents types de modèles de processus ont été introduits. Les nouveaux modèles de processus ne remplacent pas les anciens ; les méthodes agiles par exemple, apparues dans les années 2000, sont basées sur des modèles de processus orientés activité, introduits dans les années 70.

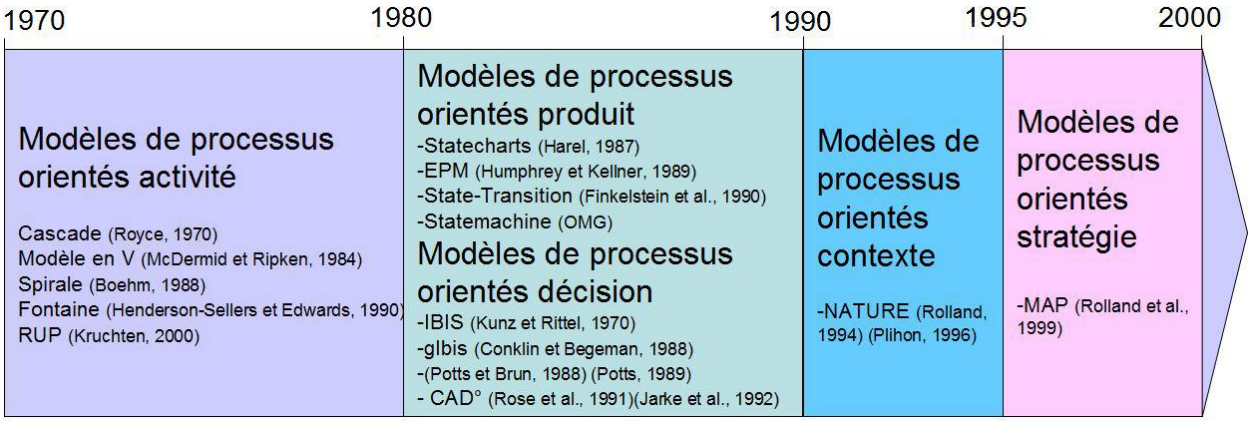


Figure 4.2 : Les différents types de modèles de processus

La modélisation d'un processus est indépendante du système d'information ; elle requiert :

- l'axe de modélisation de l'organisation : activité, produit, stratégie, etc.
- la compréhension et la maitrise du métier (Business Pattern)
- l'existant en termes de ressources

Chaque langage de modélisation associe le processus avec son environnement selon le contexte de l'organisation. Mais cette approche de regroupement rend souvent le modèle complexe et fermé.

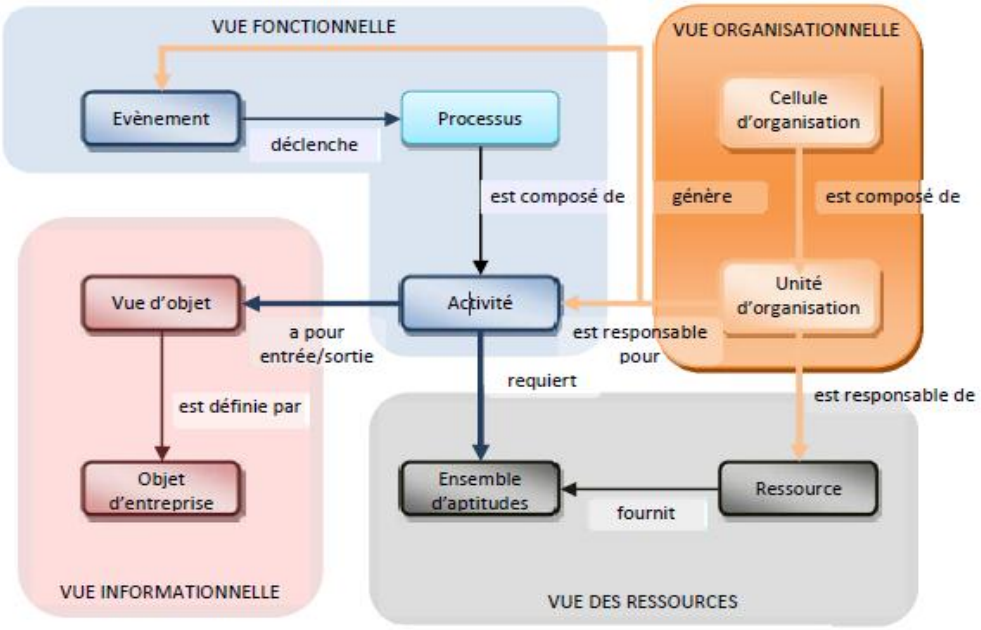


Figure 4.3 : Interaction des processus avec les éléments de l'entreprise

A l'aide du graphe de la figure 4.3, une activité d'un processus interagit avec le système à partir :

- de l'aptitude de la ressource disponible
- de l'unité de l'organisation responsable du métier
- les événements fonctionnels

Dans ce contexte, la modélisation d'un processus n'est pas une simple fonction de transformation mais un ensemble de procédure et de traitement en boucles.

La démarche processus se déroule en plusieurs étapes :

- étude du système en analysant ses objectifs et son organisation afin d'être en mesure de décomposer l'ensemble de son activité en processus métier.
- modélisation des processus métiers, c'est-à-dire représenter informatiquement un modèle le plus proche possible de la réalité,
- mise en œuvre : adopter la solution de BPM, reliée au système d'information de l'entreprise (applications et bases de données)
- exécution : phase opérationnelle où la solution de BPM est mise en œuvre.
- pilotage : consiste à analyser l'état des processus à travers des tableaux de bord présentant les performances des processus
- optimisation : proposer des solutions permettant d'améliorer les performances des processus métiers.

IV.2.2.1 Model de processus métier

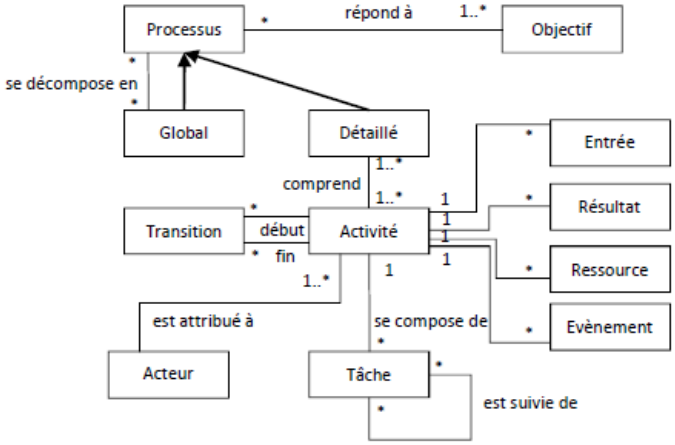


Figure 4.4 : Méta modèle du processus métier

En se basant du modèle de base de la figure 4.4 et la figure 4.5, le processus métier interagit avec trois sous-systèmes :

- gestion de tableau de bord (objectif, entrée, résultat)
- gestion de ressources (ERP) : acteurs, ressources
- gestion de services (SOA et Event Bus) : évènement, transition

IV.2.2.2 Cycle de vie de processus métier

Les différentes étapes de cycle de vie de processus sont réalisées dans deux domaines bien distincts : le domaine métier et le domaine informatique. Les étapes appartenant au domaine métier sont relatives à l'analyse, la modélisation et l'optimisation du processus. Le domaine informatique permet l'automatisation de la gestion de processus et le prélèvement de données pendant son exécution.

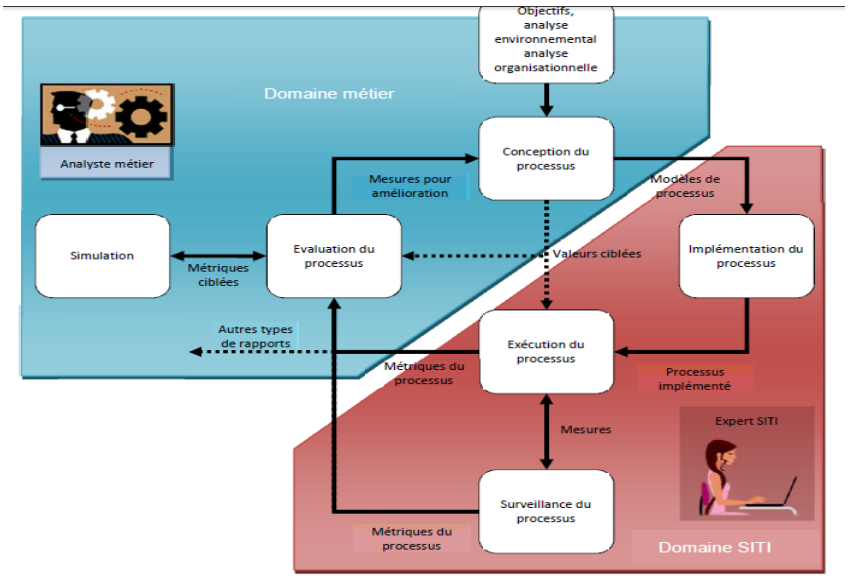


Figure 4.5 : Cycle de vie du processus métier

Les conditions qui permettent de faire tourner ce cycle de vie sont :

- le spécialiste du métier doit capable de comprendre la forme du processus sur l'outil de modélisation
- chaque modèle de processus est mesurable en termes de ressources, productivité, durées et qualités (satisfaction, qualité et fiabilité de produit, etc.)
- le support informatique est disponible pour tous les acteurs

IV.3 Management par les processus

Le management par les processus s'appuie sur un trois piliers principaux :

- les ressources humaines
- une gestion de l'actif processus
- un système de gouvernance associé à des processus de gestion de la performance

IV.3.1 Ressources humaines

Cette ressource a pour mission :

- d'optimiser l'intégration des processus de l'entreprise en utilisant des méthodes et des outils cohérents avec l'ensemble des composants de l'entreprise.
- de structurer les processus
- de définir les rôles associés distribués au sein de l'organisation

Tableau 4.1 : Responsabilité des acteurs pour la gestion des processus

Acteur	**Responsabilité**
Dirigeante	- Définit les objectifs - Définit la démarche à mettre en œuvre (approche, organisation, moyens, périmètre, services proposés, plan de déploiement, plan de communication, ...) - Met en place l'organisation et s'assure du niveau de compétences des acteurs - Définit et met en place le système de gouvernance - Définit et met en place les modalités de suivi de la performance des processus (et des interfaces processus) - Définit et suit les critères de mesure du retour sur investissement - Assure le *reporting* sur des performances - Définit et met en œuvre la politique de communication - Est garant du référentiel processus de haut niveau
Experts processus Responsable qualité processus	- Participe aux travaux de définition de processus - Définit et maintient le méta modèle du référentiel processus et les conventions de modélisation - Participe à la mise en en œuvre et à la maintenance des outils (modélisation, mesure performance…) - Assiste les responsables processus et leurs équipes dans le

	déploiement de la démarche processus - Contribue à la définition et à la mise en œuvre des outils en matière de communication
Propriétaire de processus	- Assiste aux revues de processus, et leur cycle d'amélioration continue - Définit les critères de la performance concernée et coordonne la mise en place des outils de mesure - Suit la performance du processus et diffuse les tableaux de bord auprès de sa ligne hiérarchique - Est responsable de la gestion de la connaissance et de la compétence en matière de processus auprès de son entité - S'assure de la disponibilité des ressources nécessaire et à la maîtrise des risques - Est un interlocuteur privilégié de l'entité de gestion de SI en matière d'industrialisation du processus
Architecte processus	- Conduit les travaux d'analyse et d'optimisation des processus (animation d'ateliers, définition de processus cibles, déploiement, …) - Suit et analyse la performance du processus - Echange avec les architectes des autres processus pour travailler sur des interfaces de processus optimisées.

IV.3.2 Système de gestion des processus

Ce système est l'outil principal utilisé par l'ensemble de ressources humaines pour gérer l'ensemble des processus :

- création et modélisation
- utilisation
- mesure
- mise à jour et adaptation
- cartographie et recherche

Cet outil permet de prendre en compte à la fois les aspects comportement, connaissance, communication, temporel, et environnement de travail. Notre démarche consiste à utiliser cette modélisation pour simuler les activités collaboratives afin de proposer une assistance à ces pratiques.

IV.3.3 Système de gouvernance par performance

Les priorités stratégiques et opérationnelles sont les facteurs principaux qui forcent le SI de mettre à jour ses processus. Disposer un système de mesure de la performance de processus permet d'ajuster le processus et d'apporter de nouvelle valeur ajoutée à l'organisation.

Le pilotage des processus implique la mise en place d'une démarche instrumentée pour mesurer les performances. Le management par les processus s'impose comme un véritable mode de gestion collaboratif et offre un levier puissant d'amélioration des performances de l'entreprise.

La gestion des processus métiers est aussi un processus cyclique d'amélioration continue et qui comporte d'au moins trois activités fondamentales (Figure 4.6) :

- élaboration des processus métiers : Cette activité recourt généralement à la description de modèles de représentation facilitant la compréhension et la communication des informations structurées entre toutes les parties prenantes de l'organisation.
- mise en œuvre des processus métiers est une transposition des modèles de processus métiers théoriques dans leur environnement de production.
- supervision et amélioration des processus

Le système de gestion des processus implique la mise en place des trois autres types de processus (autre que les quatre types définis dans le paragraphe IV.2) :

- processus de conception de processus
- processus de simulation de processus
- processus d'analyse et d'amélioration de processus

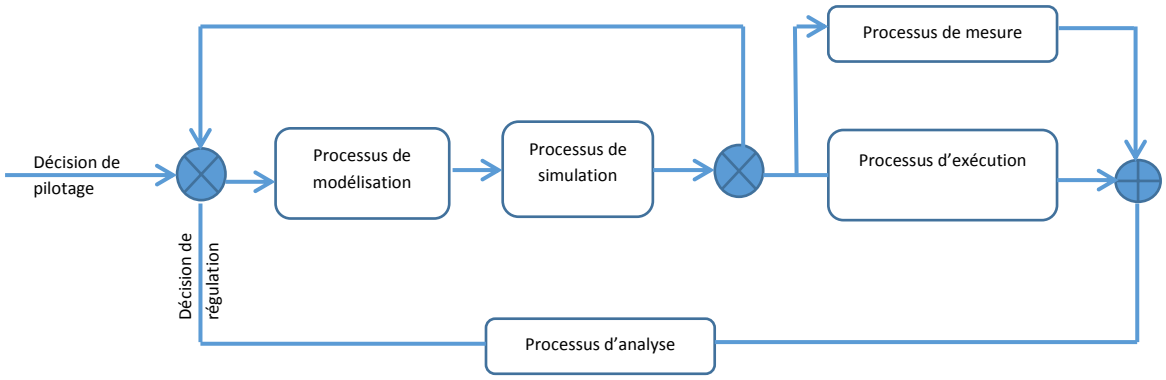


Figure 4.6 : Schéma synoptique de système de gestion des processus

Le management de l'organisation possède trois niveaux : régulation, pilotage et planification (Cf. paragraphe I.2.4.3). Dans le concept métier, la conception des processus fait partie de l'axe stratégique de pilotage de l'organisation tandis que l'amélioration des processus fait partie de l'axe de la régulation de l'organisation. Elle permet à l'aboutissement d'une décision structurée qui peut être à la suite traduite à une nouvelle cartographie de processus et une nouvelle organisation.

IV.3.3.1 Niveau de maturité de traitement

Un traitement ou une suite de traitement peut être classifié en quatre niveaux selon la maitrise du système. Cette maitrise consiste à associer le concept lié au traitement avec un environnement de contrôle.

Tableau 4.2 : Niveau de maturité de traitement

Niveau	**Description**
Fonctionnement de base	« les choses se passent » Existence des produits entrant et sortant
Défini, planifié, suivi	Système qualité formalisé autour de procédure écrites Pas ou peu de mesure
Maitrisé	Avec des mesures de résultats
Optimisé	Avec optimisation des ressources
Amélioration permanente	Insertion des outils de prévention pour activer la mise à jour

IV.3.3.2 Démarche d'amélioration

Une amélioration implique toujours des changements, des objectifs et des ressources. En pratique, la gestion de changement se fait par étape pour pouvoir contourner différents problèmes d'intégrations :

- contrainte de l'existant et des ressources
- contrainte d'adaptation humaine et formation
- les imprévus de l'analyse

Plusieurs techniques sont élaborées pour mener à bien des changements mais le processus le mieux connus sont la démarche DMAIC *Six Sigma* et la démarche d'amélioration continue PDCA

a. Démarche DMAIC

Pour les décisions programmables et structurées, qui concernent des plans purement techniques, on peut adopter la démarche DMAIC pour générer des objectifs à moindre écart. Cette démarche peut éliminer les variations et les défauts du processus par une répétition d'une boucle de traitement.

La méthodologie de processus *Six Sigma* DMAIC est un système qui apporte une amélioration mesurable et significative aux processus existants qui tombent au-dessous des spécifications. La méthodologie DMAIC peut être utilisée lorsqu'un produit ou un processus est en vigueur dans le système mais ne satisfait pas les attentes.

La mise en œuvre d'une démarche Six Sigma se fait selon les étapes suivantes (Figure 4.7) :

- définir : l'équipe de pilotage identifie un projet d'amélioration basé sur les objectifs opérationnels et les besoins des clients
- mesurer : dans cette phase, l'équipe commence avec les paramètres appropriés. Des mesures essentielles qui sont nécessaires pour évaluer la réussite du projet sont identifiées et déterminées. La capacité initiale et la stabilité du projet est déterminée afin d'établir une base de mesure. Des mesures valides et fiables pour surveiller la progression du projet sont établies pendant la phase de mesure
- analyser : l'équipe peut déterminer les causes du problème qui a besoin d'amélioration et la façon d'éliminer l'écart entre le rendement actuel et le niveau de performance souhaité. Cela consiste à découvrir pourquoi les défauts sont générés en identifiant les variables clés qui sont plus susceptibles de créer des variations.
- améliorer : la phase d'amélioration consiste à rechercher, proposer et faire appliquer des solutions adaptées
- contrôler : A l'issue de la phase de contrôle, le suivi est transféré à l'équipe propriétaire du processus pour réévaluer le nouveau système

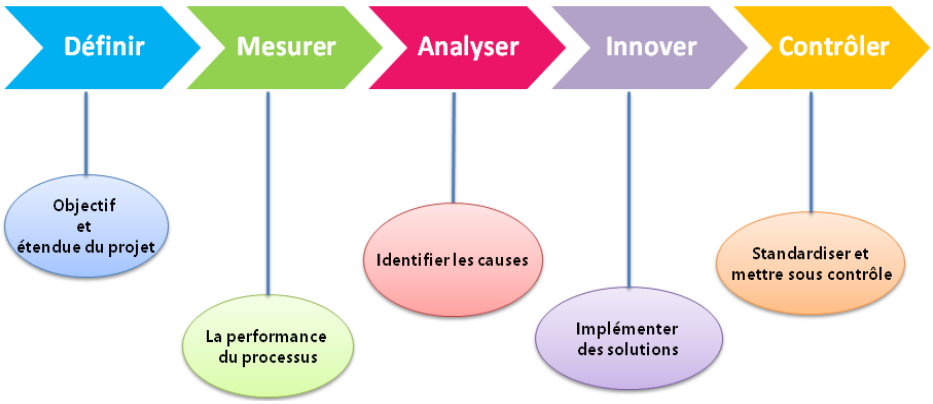


Figure 4.7: Méthodologie DMAIC

Cette démarche repose sur 4 principes de mise en œuvre :

- une méthodologie rigoureuse qui propose une imbrication logique de plusieurs outils pour résoudre les problèmes
- une prise de décision reposant sur des faits réels et mesurés et non sur des opinions
- une implication d'une équipe dirigée par un expert de la méthodologie
- centré le processus de résolution de problèmes et la satisfaction client

b. Démarche PDCA

La méthode PDCA est une démarche cyclique (Figure 4.8) d'amélioration qui consiste, à la fin de chaque cycle, à remettre en question toutes les actions précédemment menées afin de les améliorer :

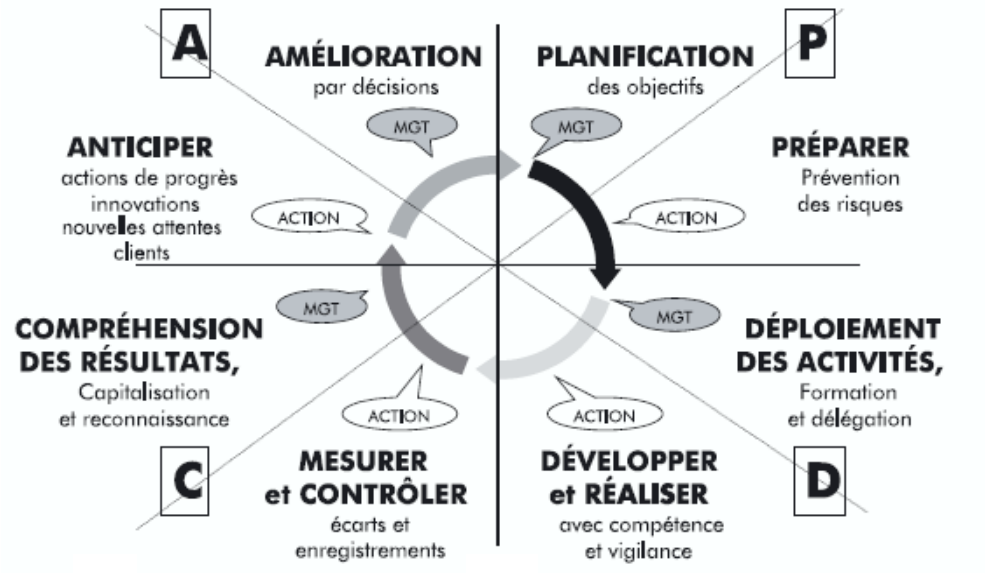


Figure 4.8 : Management en PDCA

❖ Plan

La première étape du cycle « Planifier », consiste à préparer et planifier ce que l'on va réaliser : définir le cahier des charges et établir un planning.

Objectif :

Identifier le vrai problème, rechercher les causes racines et planifier la mise en œuvre des actions correctives :

- construire une équipe multidisciplinaire et organiser un travail de groupe
- identifier clairement et formaliser le problème
- rechercher les causes racines
- analyser et visualiser les causes
- classer et hiérarchiser les causes
- valider les causes principales
- comparer la situation actuelle à la situation
- définir l'objectif et créer des indicateurs de mesure
- rechercher les solutions possibles
- sélectionner les solutions optimales;
- déterminer les moyens de collecte des informations
- planifier la mise en œuvre des actions correctives

❖ Do

La seconde étape du cycle « faire » est la construction, la réalisation, de l'œuvre. Elle commence toujours par une phase de test.

Objectif :

Exécuter le plan d'action, déployer les ressources nécessaires et mettre en œuvre toutes les opérations correctives mentionnées dans le plan, toutes les solutions retenues :

- définir la zone d'expérimentation
- tailler un échantillon représentatif pour les tests
- appliquer les actions correctives définies dans le plan
- vérifier le résultat et le mesurer à l'aide des indicateurs d'activité
- s'il est satisfaisant, étendre les solutions à l'ensemble de la population.

❖ Check

La troisième étape « vérifier » consiste à contrôler que les ressources mises en œuvre dans l'étape précédente (Do) et les résultats obtenus correspondent bien à ce qui a été prévu (Plan). Divers moyens de contrôle sont alors déployés (Tableau de bord : indicateurs de performance ...).

Objectif :

Contrôler que les ressources mises en œuvre dans l'étape précédente (Do) et les résultats obtenus correspondent bien à ce qui a été prévu (Plan) :

- mesurer les résultats obtenus sur l'ensemble après la fin de l'étape « Do » ;
- comparer ces résultats à la situation initiale (Mesure de l'amélioration) ;
- comparer ces résultats aux objectifs fixés dans l'étape « Plan » (Mesure de la performance) ;
- identifier les causes des dérives entre les réalisations et les objectifs attendus.

❖ Act

Enfin la dernière étape du cycle « agir », consiste à ajuster les écarts à rechercher des points d'améliorations. Ce qui amènera un nouveau projet à réaliser, donc une nouvelle planification à établir. Et ce sera le début d'un nouveau cycle.

Objectif :

Ajuster les écarts, vérifier l'efficacité des solutions mises en place, rechercher des points d'améliorations tant que le niveau attendu n'est pas atteint :

- identifier les causes de non performance
- cibler les nouveaux points d'intervention
- reprendre pour ces points les étapes « Do » et « Check » si nécessaire
- mettre en place des systèmes anti-erreur
- formaliser les solutions une fois que le niveau attendu est atteint
- encadrer le groupe de travail afin de préserver les acquis

IV.3.4 Gestion de la performance

D'une manière générale, la performance est un résultat chiffré obtenu dans le cadre d'une compétition.

Au niveau d'un SI, la performance exprime le degré d'accomplissement des objectifs poursuivis. Un système performant doit être à la fois efficace et efficiente :

- il est efficace lorsqu'elle atteint les objectifs qu'elle s'est fixés.
- il est efficient lorsqu'elle minimise les moyens mis en œuvre pour atteindre les objectifs qu'elle s'est fixés.

IV.3.4.1 Comment mesurer la performance

La performance se mesure avec des critères (ou indicateurs) qualitatifs ou quantitatifs de résultat. Pour mesurer l'efficacité, on utilise un critère qui exprime un rapport entre le résultat obtenu et l'objectif visé. Pour mesurer l'efficience, on utilise un critère qui exprime un rapport entre le résultat obtenu et les moyens mis en œuvre.

Pour évaluer la performance d'une entreprise, il est nécessaire d'effectuer des mesures au niveau :

- des données métiers (lier aux exigences du monde externe qui nécessite d'autres outils d'interfaçage)
- des données de traitements BPM (mesure interne du SI)

Selon la figure 1.7, le traitement de ces indicateurs est un processus qui évolue dans le système en terme de traitement mais non pas en terme de structure.

IV.3.4.2 Exemple de performances métiers

Les niveaux de métier dans une entreprise sont tous susceptibles d'être mesurés : financier, économique, social, organisationnel et sociétal.

a. La performance financière

Techniquement, on mesure la performance financière à l'aide des indicateurs ROI et ROE. Mais avec l'économie moderne, on utilise en plus l'indicateur EVA.

- ROI (Return On Investment) : ce ratio mesure la rentabilité économique du capital utilisé par l'entreprise. C'est le rapport entre le résultat d'exploitation et les capitaux investis.
- ROE (Return On Equity) : ce ratio mesure la rentabilité financière des capitaux apportés par les propriétaires de l'entreprise. C'est le rapport entre le résultat net et les capitaux propres.
- EVA (Economic Value Added) : ce ratio permet de mesurer la création de valeur pour l'actionnaire. C'est la différence entre le résultat opérationnel et les capitaux investis.

b. La performance économique

Il s'agit de mesurer les composantes de la compétitivité de l'entreprise : la compétitivité-prix et la compétitivité-hors prix.

- compétitivité-prix : désigne la capacité d'un produit à attirer des clients au détriment des produits concurrents du fait de son prix. Sa mesure permet de situer la place de l'entreprise sur le marché par rapport à ses concurrents.
- compétitivité hors-prix : désigne la capacité d'un produit à attirer des clients au détriment des produits concurrents du fait des éléments indépendants du prix. Elle est obtenue grâce à des éléments comme la qualité des produits, l'innovation, le service, le design.

c. La performance organisationnelle

Il s'agit de mesurer la performance de l'entreprise au niveau de la qualité de la production, de la flexibilité, des délais.

d. La performance sociale

Le bilan social récapitule les principales données chiffrées permettant d'apprécier la performance sociale et les rapports sociaux au sein d'une entreprise. Parmi les nombreux indicateurs sociaux, on peut citer : le montant des rémunérations, le nombre d'accidents de travail, les maladies professionnelles.

e. La performance sociétale

Elle indique l'engagement de l'entreprise dans les domaines environnementaux, humanitaires, culturels. Les outils de la responsabilité sociétale de l'entreprise (RSE) peuvent être utilisés pour apprécier le niveau de performance de l'entreprise.

IV.3.5 Pattern sur les processus de contrôle

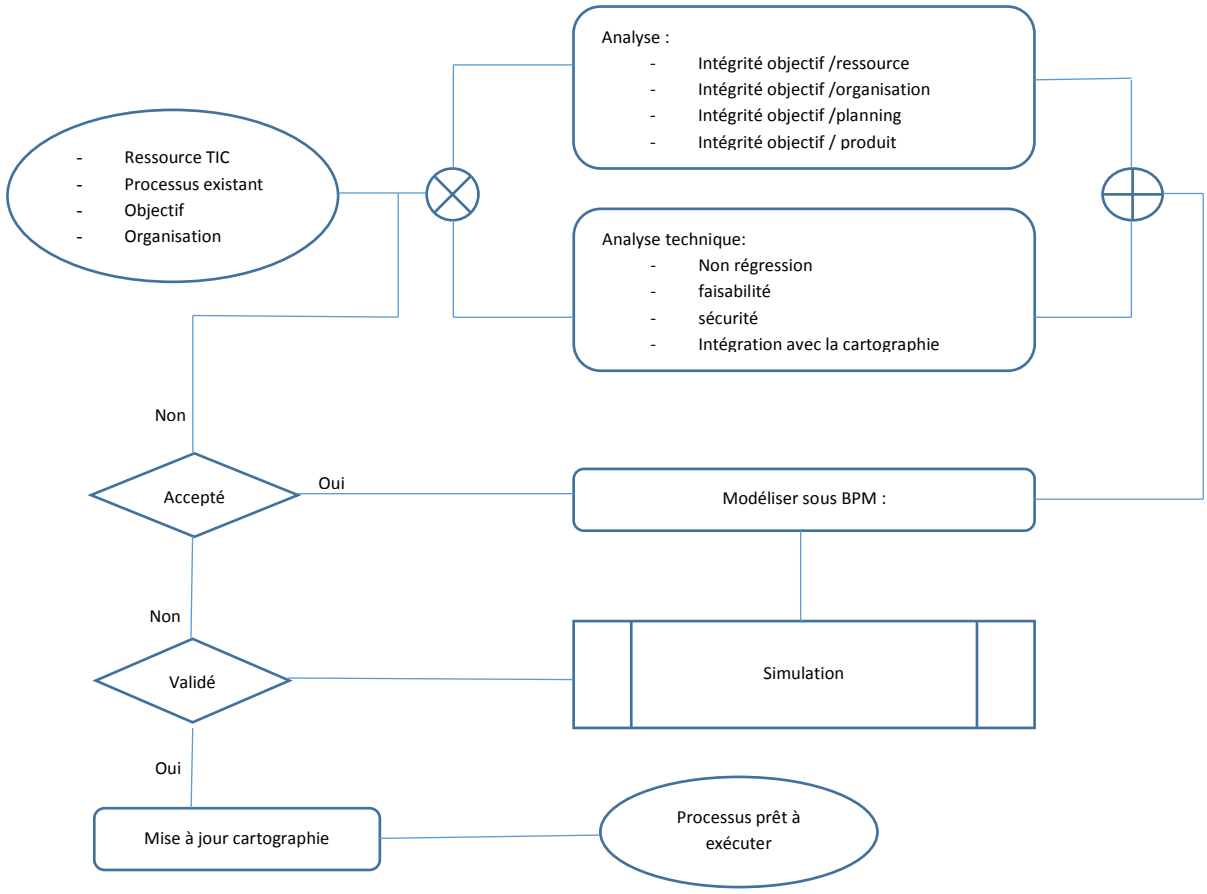
En utilisant les deux démarches DMAIC et PDCA, on peut conclure que pour maitriser l'amélioration des processus dans un système, il faut :

- effectuer une boucle de traitement
- mesurer
- analyser
- modéliser

En adoptant le processus en boucle fermé défini dans le Paragraphe IV.3.3, chaque système doit incorporer ces quatre processus de contrôle statiques comme étant socle de traitement d'amélioration continue pour les processus métiers.

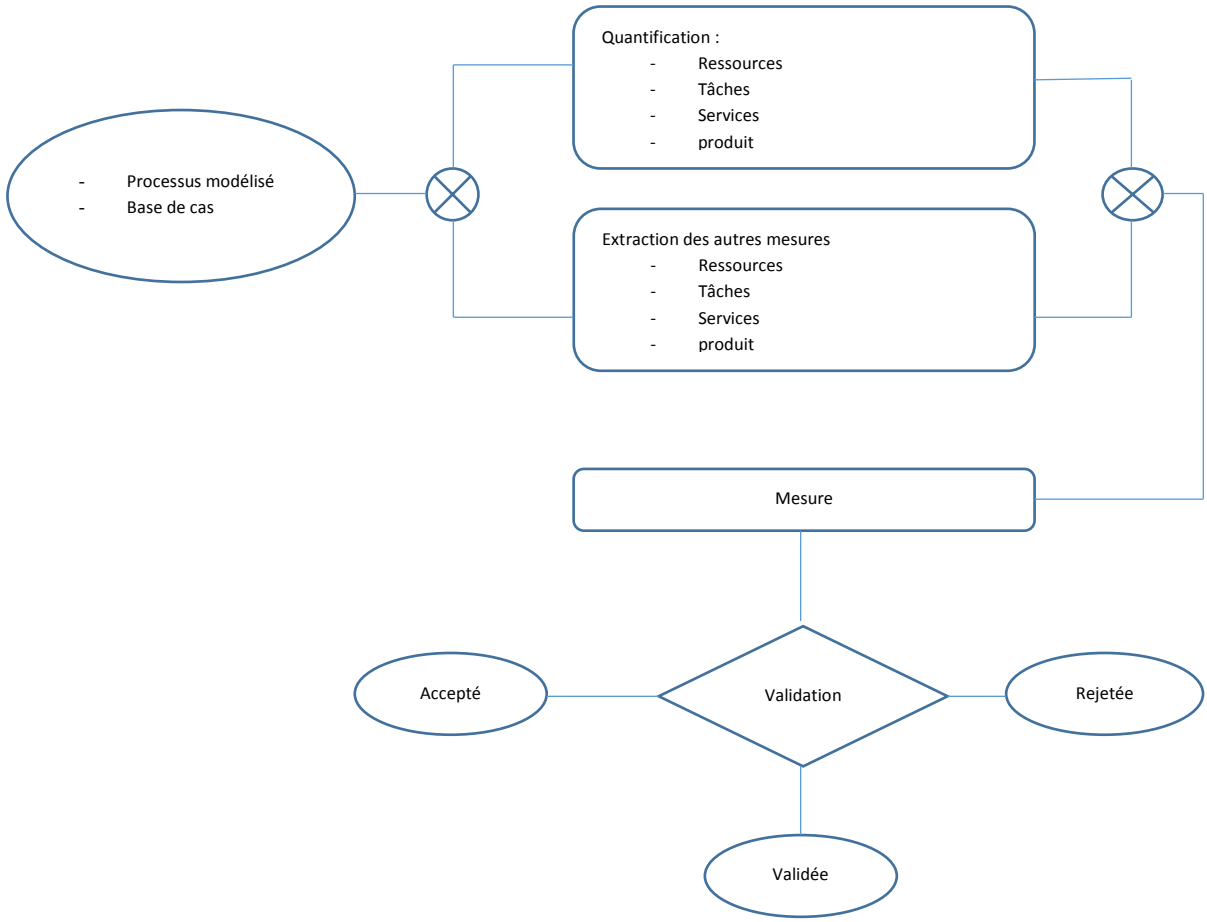
IV.3.5.1 Processus de modélisation

Tableau 4.3: Pattern *Process design*

Process design
Intention : convertir objectif et traitement en processus.
Indications d'utilisation. Service workflow.
Structure 
Participants principaux - Analyse fonctionnel : définit les contraintes liées à l'objectif - Analyse technique : définit les contraintes liées à l'implémentation - Cartographie : ensemble de processus existants - BPM : outil de gestion des processus - Simulation : sous processus

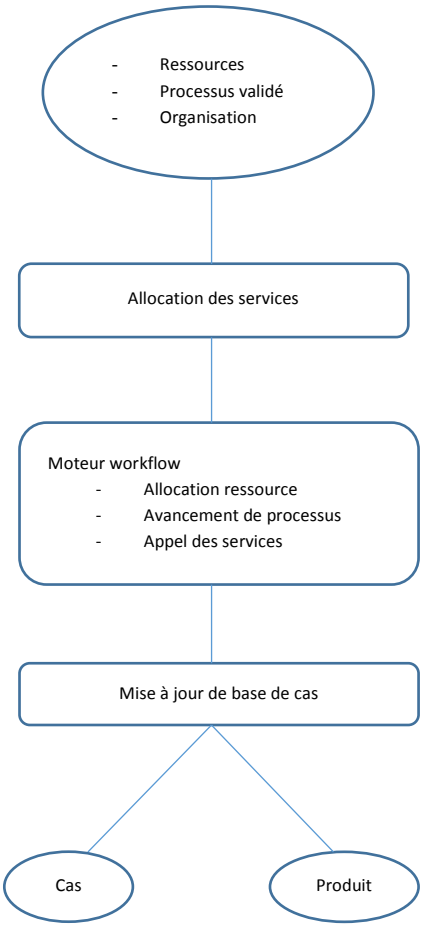
IV.3.5.2 Processus de simulation

Tableau 4.4 : Pattern *Process evaluation*

Process evaluation
Intention : évaluer le coût et la qualité du processus.
Indications d'utilisation. Service workflow
Structure 
Participants principaux - Base de cas : données liées aux traitements antérieurs - Quantification : outil de gestion des valeurs par unité - Mesure : application de la quantification sur le processus

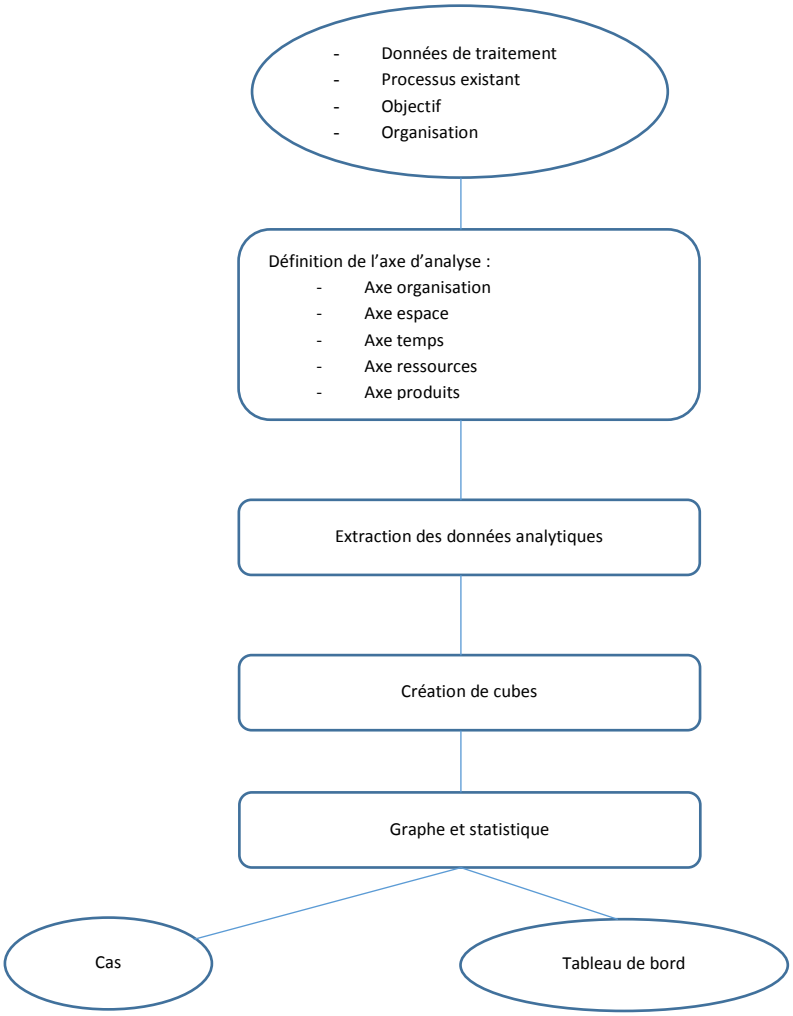
IV.3.5.3 Processus d'exécution

Tableau 4.5 : Pattern *Process execution*

Process execution
Intention : Utiliser le processus validé dans le cas réel (production).
Indications d'utilisation. Service workflow
Structure 
Participants principaux - Gestionnaire des services : fournisseur de service - Moteur workflow : instanciation dans l'outil BPM - Base de cas : données liées aux traitements antérieurs - Produit : élément de sortie du métier

IV.3.5.4 Processus de rapport

Tableau 4.6 : Pattern *Process report*

Process report
Intention : convertir les données de traitement en tableau de bord.
Indications d'utilisation. Service workflow, service management
Structure 
Participants principaux - Axe d'analyse : donnée clé pour regrouper les données opérationnelles - Extraction des données : traitement fonctionnel et technique pour compacter les données opérationnelles par rapport aux axes d'analyse - Cube : données structurées selon un arbre défini par les axes d'analyse - Tableau de bord : Graphe, tableau ou autre illustration qui représente les données structurées - Base de cas : données liées aux traitements antérieurs

IV.4 Cartographie des processus

La cartographie processus offre une vue globale du fonctionnement d'un organisme. Elle permet de visualiser ses processus, leurs interactions et distingue les processus de réalisation, les processus support et les processus de management.

La cartographie processus est un outil qui :

- permet une meilleure compréhension du fonctionnement par le personnel;
- facilite le pilotage global de l'organisme;
- facilite l'intégration des nouveaux collaborateurs;
- met en évidence la finalité des activités et l'implication nécessaire de tous.

Avec cet outil, l'organisation peut déduire les axes des analyses de son système.

Les avantages de la cartographie de processus sont de permettre de communiquer de manière identique à un grand nombre d'acteurs impliqués dans une activité complexe et de donner du sens et de la clarté immédiate sur les tâches à réaliser.

IV.4.1 Caractéristiques de la cartographie des processus

Etablir la cartographie des processus est une étape préalable indispensable non seulement pour faciliter les opérations de rationalisation mais aussi pour mieux cibler la démarche de progrès.

Pour élaborer cette carte, il y a deux points critiques :

- les mots doivent être compris par tous. Il faut donc faire valider par les acteurs impliqués dans le processus.
- la cartographie ne doit pas être trop surchargée ou illisible.

Les critères de succès de la cartographie peuvent être synthétisés par comme suit :

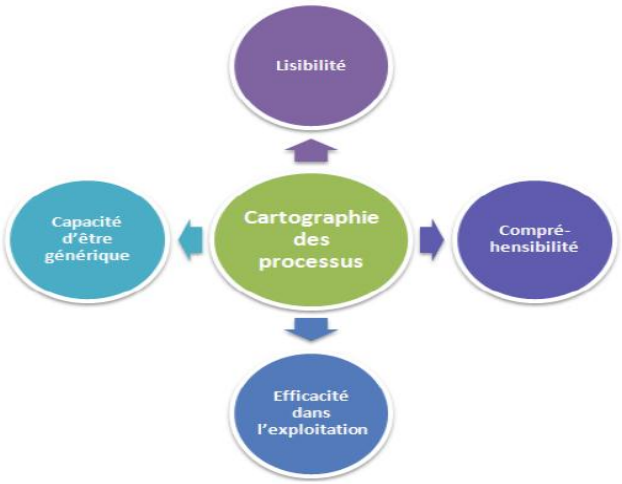


Figure 4.9 : Propriétés d'une cartographie de processus

IV.4.2 Plan orienté métier et produit

La carte de processus implique l'ensemble de métiers de l'organisation et sa structure (imbrication). La lecture de cette carte permet d'identifier la hiérarchie de processus et leur interaction jusqu'au détail de l'opération (activité d'un processus). Au bout de chaque hiérarchie, on doit trouver un produit/service ou un sous-produit.

La navigation peut se faire en :

- drill up : de l'organisation au produit (ou service)
- drill down : de produit vers l'organisation

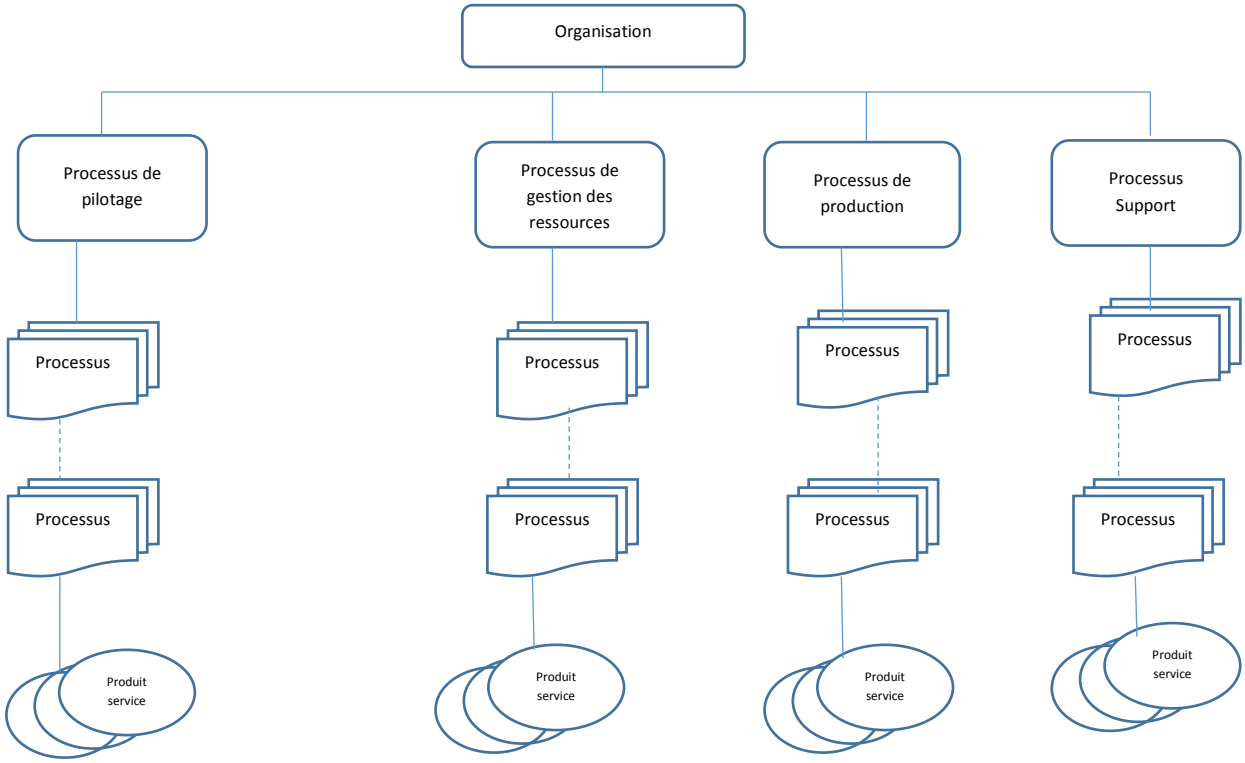


Figure 4.10 : Arbre des processus

La relation parent/fils (exemple Figure 4.10) pour les processus est définie par l'appel par service (car un service peut être une implémentation d'un autre processus) ou par la fonctionnalité sous processus.

IV.5 Pattern BPM

Le terme BPM est apparu pour la première fois au milieu des années 1990 dans le contexte de la gestion des stratégies d'entreprise en se basant sur les processus métier.

Définition 4.3 :

BPM est un support des processus métier en utilisant des méthodes, des techniques et des outils logiciels afin de concevoir, modéliser, commander et analyser les processus opérationnels impliquant des humains, des organisations, des applications et toutes autres sources d'information.

Le terme Gestion des Processus Métier (BPM) inclut l'analyse, la définition et la redéfinition des processus de l'entreprise, l'allocation des ressources, la planification, la gestion des processus, la mesure de la qualité et de l'efficacité à l'aide d'indicateurs et l'optimisation des processus.
En plus, l'optimisation des processus nécessite la collecte et l'analyse de mesures en temps réels (monitoring) et de mesures stratégiques (gestion des performances) et leur corrélation, en tant que base pour, l'amélioration et l'innovation.

Le paragraphe IV.3 définit le modèle conceptuel de pattern de management par processus tandis que le BPM fournit le modèle physique de ce traitement relatif aux modèles de l'outil de gestion de système d'information.

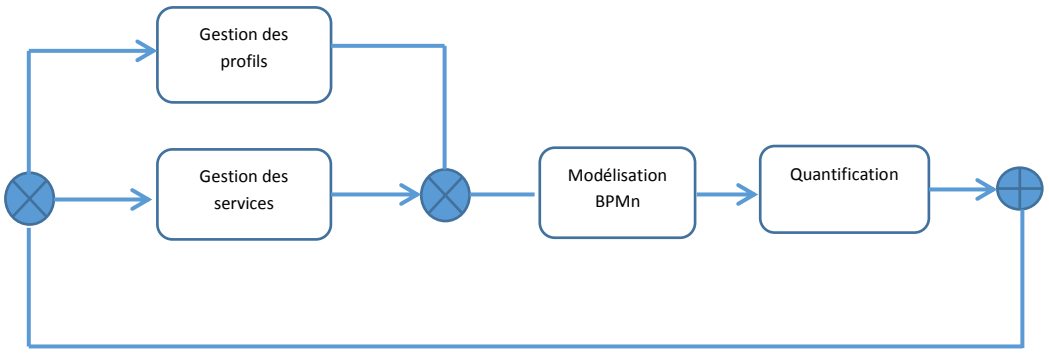


Figure 4.11 : Pattern BPM

Selon le modèle du processus métier et le processus de la figure 4.11, le pattern BPM comporte cinq patterns :

- modélisation : création de la structure et l'interaction des activités
- gestion de profil : l'interface utilisée pour présenter une ressource est le profil
- moteur workflow : gère les instances du processus
- gestion des services : gère l'allocation de traitement pour chaque activité
- quantificateur : service qui permet de quantifier la valeur d'une entité

IV.5.1 Modélisation

Le pattern de conception de processus utilise la norme BPMN. Il fournit une notation graphique pour exprimer les processus métier en *Business Process Diagram*.

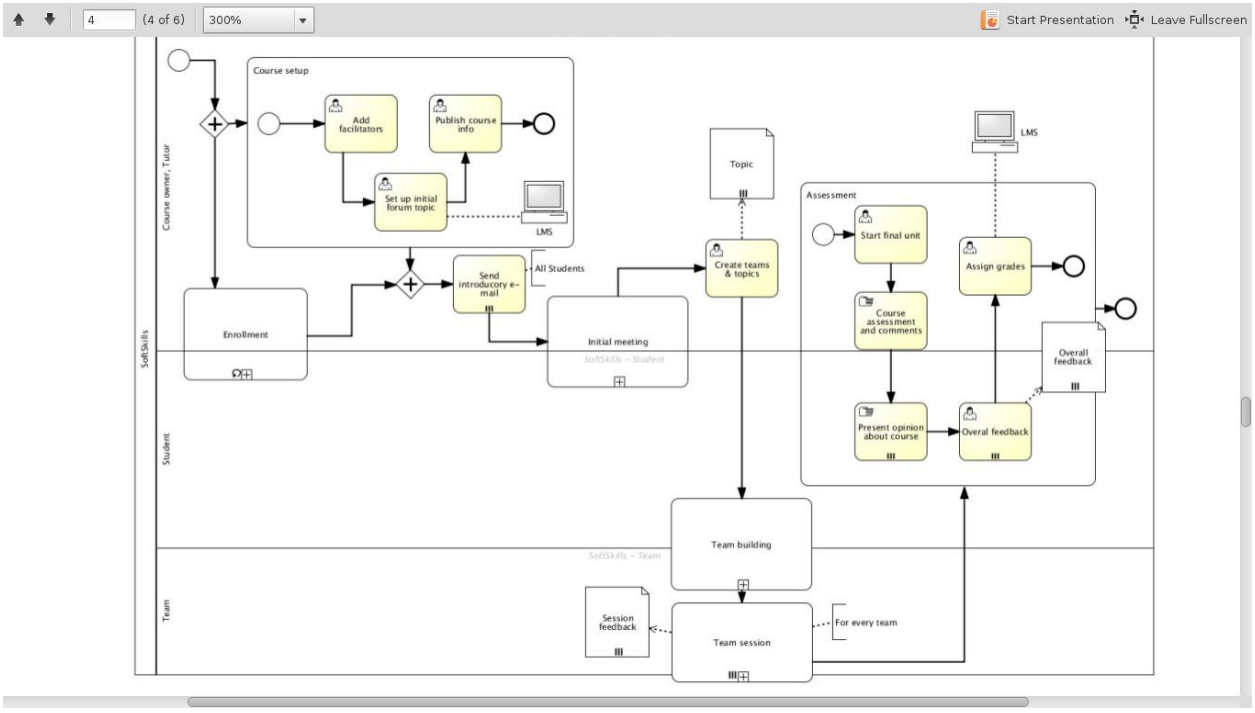


Figure 4.12 : Exemple de diagramme BPMN

L'objectif de la norme et l'outil BPMn est de fournir :

- une notation graphique complète permettant de représenter un processus métier (exemple de graphe sur la figure 4.12) en découplant les informations métiers des informations techniques ce qui fournit un cadre de travail commun aux utilisateurs métiers et techniques
- un *mapping* complet vers les langages d'exécution. Ainsi, une fois les processus modélisés par les utilisateurs métiers, et les informations techniques renseignées pour rendre le processus exécutable

BPMn utilise un schéma d'échange standard basé sur XML. Il utilise trois catégories d'éléments :

- les tâches,
- les évènements
- les connecteurs (branchement).

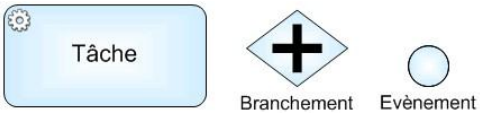
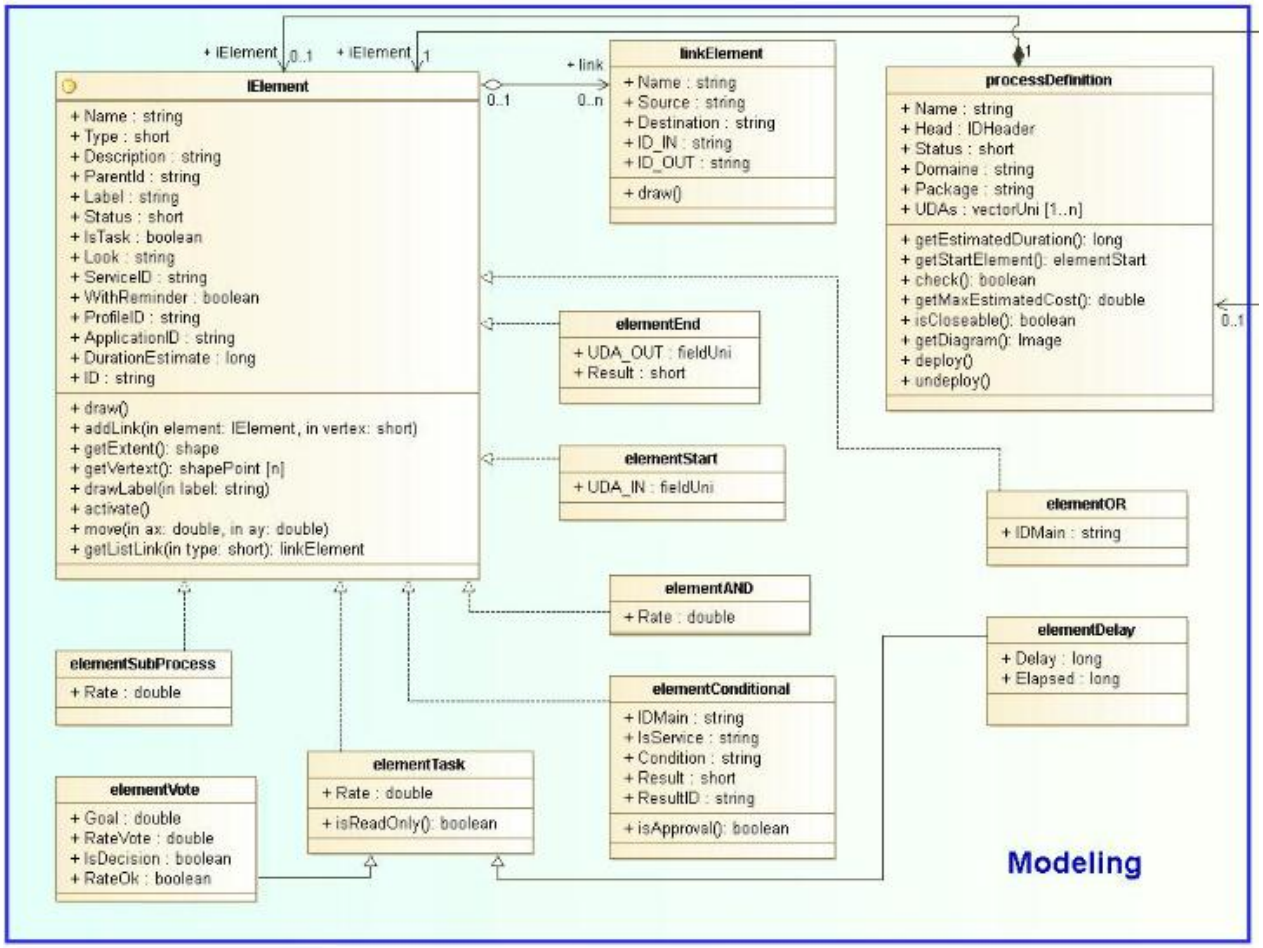


Figure 4.13 : Objet de base de diagramme BPMN

Tableau 4.7 : Pattern Process Modeling

Process modeling
Intention : convertir un processus sous forme graphique.
Indications d'utilisation. Service workflow
Structure 
Participants principaux - processDefinition : définit les données d'identification d'un processus - IElement : présente un élément du processus (activité) - linkElement : présente l'élément qui relie les activités - startElement : définit le nœud qui écoute l'évènement de l'instanciation - endElement : définit le nœud qui écoute l'évènement de clôture - elementSubProcess : présente un point d'entrée pour activer un sous processus

IV.5.2 Instanciation

Un moteur de workflow est un dispositif logiciel permettant d'exécuter des instances de modèle de processus, il permet de dérouler les tâches définies dans les processus. L'exécution est programmée ou déclenchée par des évènements.

Plusieurs possibilités sont offertes par les outils d'exécution de processus (BPMS) :

- possibilité de modifier un processus lorsqu'il est en cours d'exécution, ce qui permet à un utilisateur d'effectuer rapidement des changements dans un processus lorsqu'il se produit un problème, sans avoir à arrêter le processus et le recommencer au début après modification ;
- possibilité de distribuer la charge de travail en tenant compte de disponibilités (horaires, vacances) des employés
- possibilité de gérer les versions différentes des processus métier et donc la possibilité de pouvoir revenir en arrière sur l'évolution qu'a suivi un processus donné (pour des raisons légales ou lors d'optimisation)

Pour exécuter réellement une tâche, le moteur workflow fait appel à un annuaire de services qui à son tour gère les données opérationnelles du système.

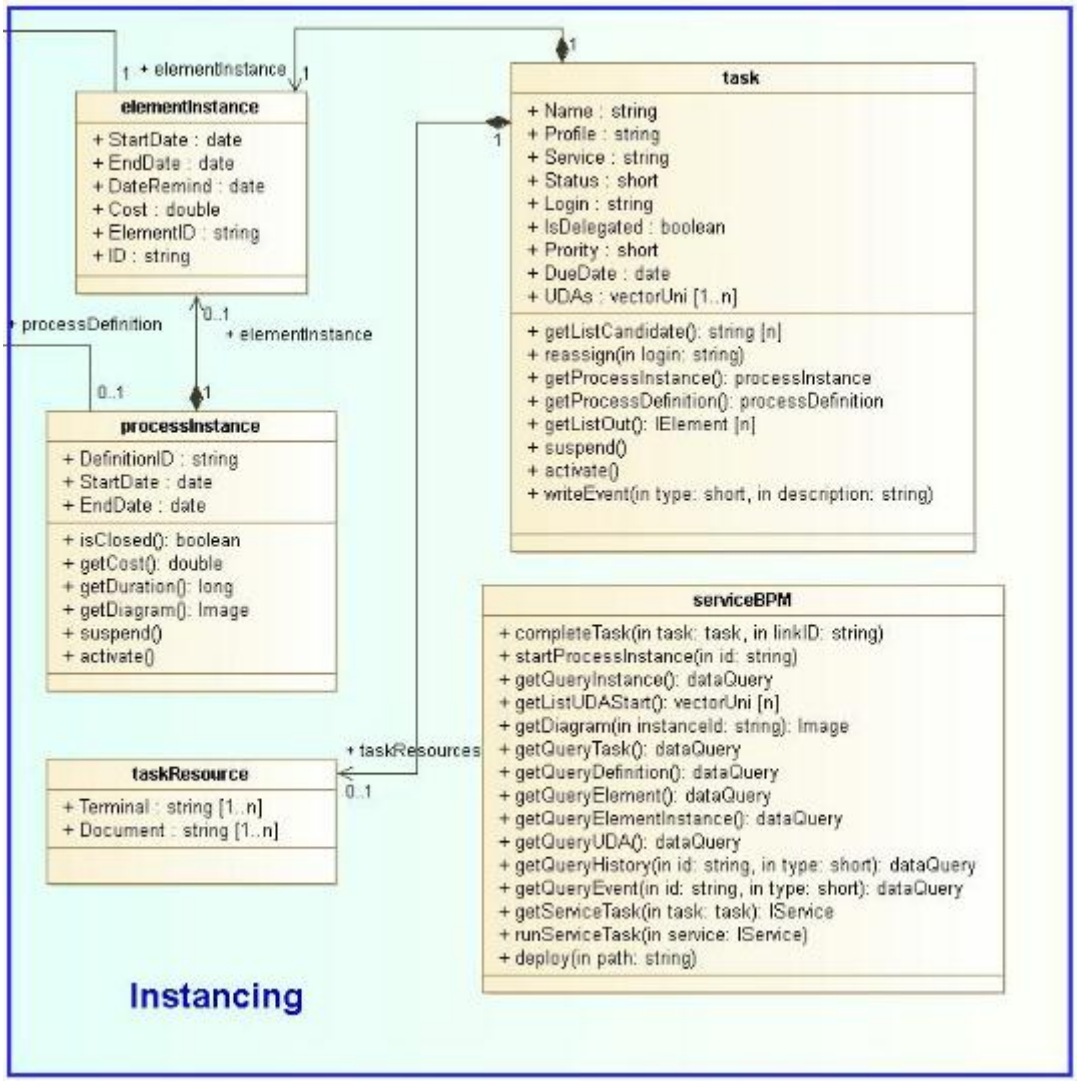
IV.5.3 Suivi et administration des instances de processus

Avec la norme BPMn, une tâche ne doit pas être re-exécutée et il est interdit de changer le cours de traitement pour une instance démarrée. Mais dans le cas pratique, au cours de l'évolution du processus, on doit tout accepter pour réduire l'impact négatif du défaut du processus.

La gestion et le traitement des instances de processus nécessitent des outils performants pour gérer :

- réallocation des ressources
- la cohérence du traitement avec le métier
- les historiques de traitement
- la traçabilité d'un produit spécifique

Tableau 4.8 : Pattern *Process instantiation*

Process instantiation
Intention : convertir les activités dans un processus en séquence des tâches.
Indications d'utilisation. Service workflow
Structure 
Participants principaux - processInstance : définit les données d'identification d'une instance - elementInstance : présente les informations relatives à la réalisation d'une tâche - task : présente une tâche liée à un taskResource - taskResource: définit une instance de ressource - serviceBPM : service workflow

IV.5.4 Service d'intégration

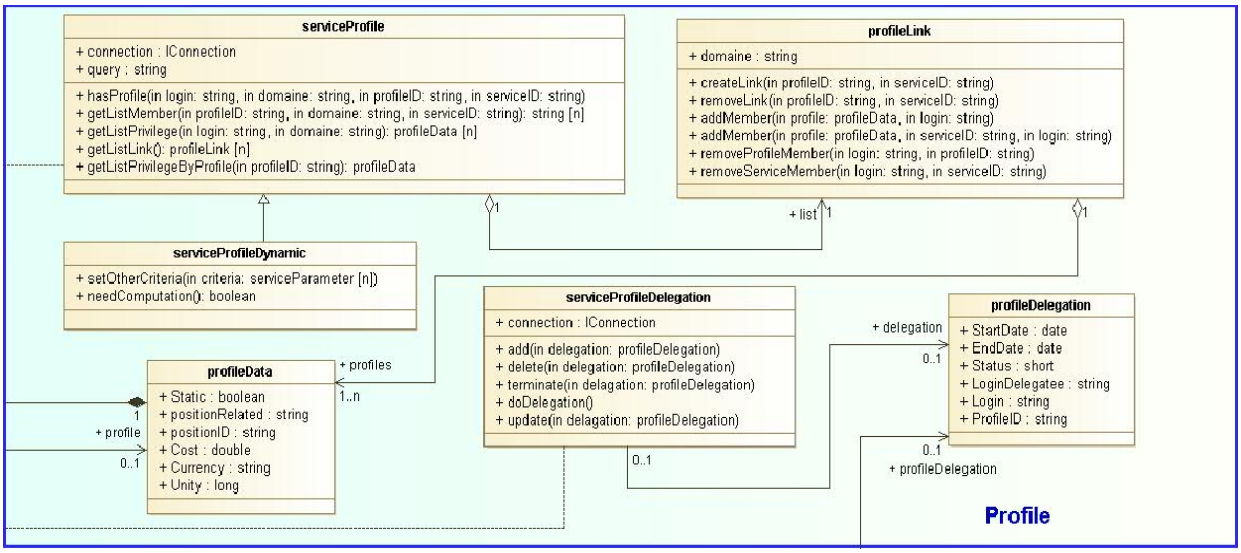
Pour l'architecture SOA et Event bus, la gestion des événements et l'appel des traitements par service constituent l'artère du système. Pour s'intégrer avec le reste du système, le cœur du BPM a besoin de trois services :

- gestion de profil
- gestion de service métier
- gestion de cartographie de processus

La liste des profils et services métiers sont indispensables pour définir les propriétés de chaque activité dans le processus tandis que la cartographie de processus permet de lier chaque processus entre eux pour le besoin de management (système qualité et pilotage).

IV.5.4.1 Service de gestion de profil

Tableau 4.9 : Pattern *Profile management*

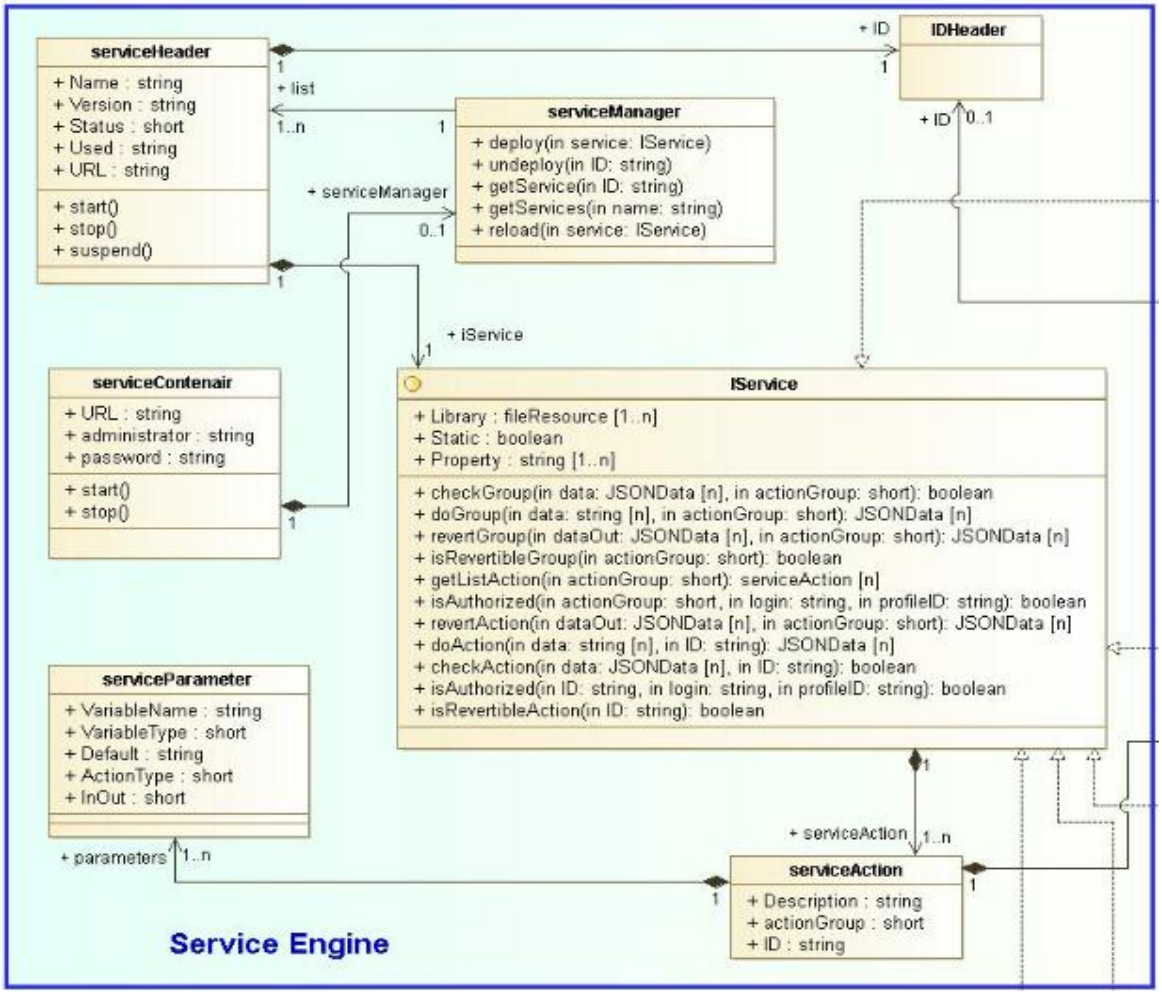
Profil Management
Intention.: présente les données et les interfaces utilisées pour gérer des ressources d'identification des acteurs.
Indications d'utilisation. Service BPM, Service d'authentification sur un IHM
Structure 

Participants principaux
- profileData : identification d'un profil
  o rôle
  o position
  o coût
- profileDelegation : profil temporaire
- serviceProfile : Service qui valide un profil
- serviceAuthenticator : Service qui valide une identification

Le profil présente un point d'accès du service BPM aux ressources du système : il permet de sélectionner une ressource adéquate au traitement et la gestion des accès aux listes des tâches générées par le BPMS.

IV.5.3.2 Fournisseur de service métier

Tableau 4.10 : Pattern *Service engine*

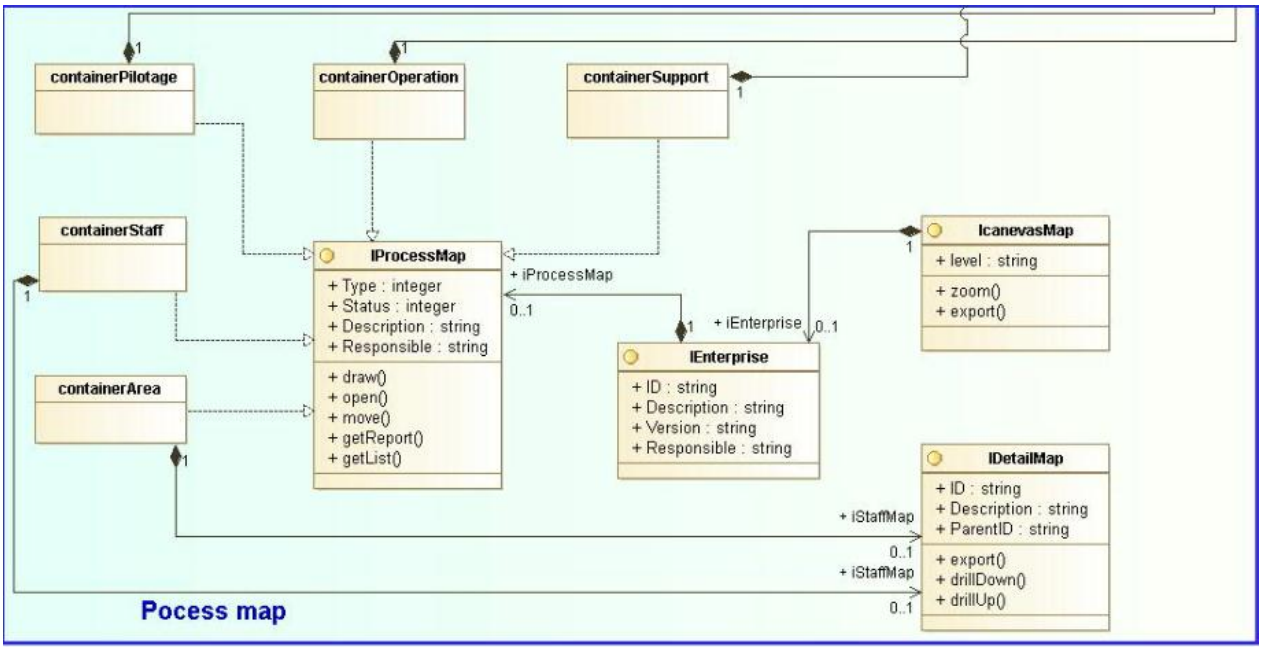
Service engine
Intention : présente les données et les interfaces utilisées pour gérer une liste des services
Indications d'utilisation. Service BPM, application
Structure 
Participants principaux - serviceHeader : identification d'un service - serviceContainer : identification d'un fournisseur de service - serviceManager : service qui gère l'intégration d'un service dans un container - IService : définit l'interface principale de service - serviceAction : liste les actions valables dans un service .

Les services sont des composants volatils, la correction du système implique toujours la mise à jour d'une liste de services. Pour les services non utilisés, le système doit les garder dans le container avec de statut désactivé pour pouvoir garder les mesures liées à ces composants.

IV.5.3.3 Service de pilotage de processus

Comme le service, la version des processus change aussi selon le contexte. Le service de pilotage de processus permet la traçabilité de la cartographie des processus et de gérer ses composants actifs.

Tableau 4.11: Pattern Process map

Process Map
Intention : présente les données et les interfaces utilisées pour gérer une liste des processus et des composants entrepris.
Indications d'utilisation. Service BPM, service Business Intelligence
Structure 
Participants principaux - IProcessMap : définit un composant graphique qui constitue la carte - IEnterprise : identification de l'organisation - containerStaff : définit une entité de l'organisation - containerArea : définit une zone de l'organisation - IDetailMap : liste des sous éléments d'une zone ou d'une organisation .

IV.5.4 Service de quantification

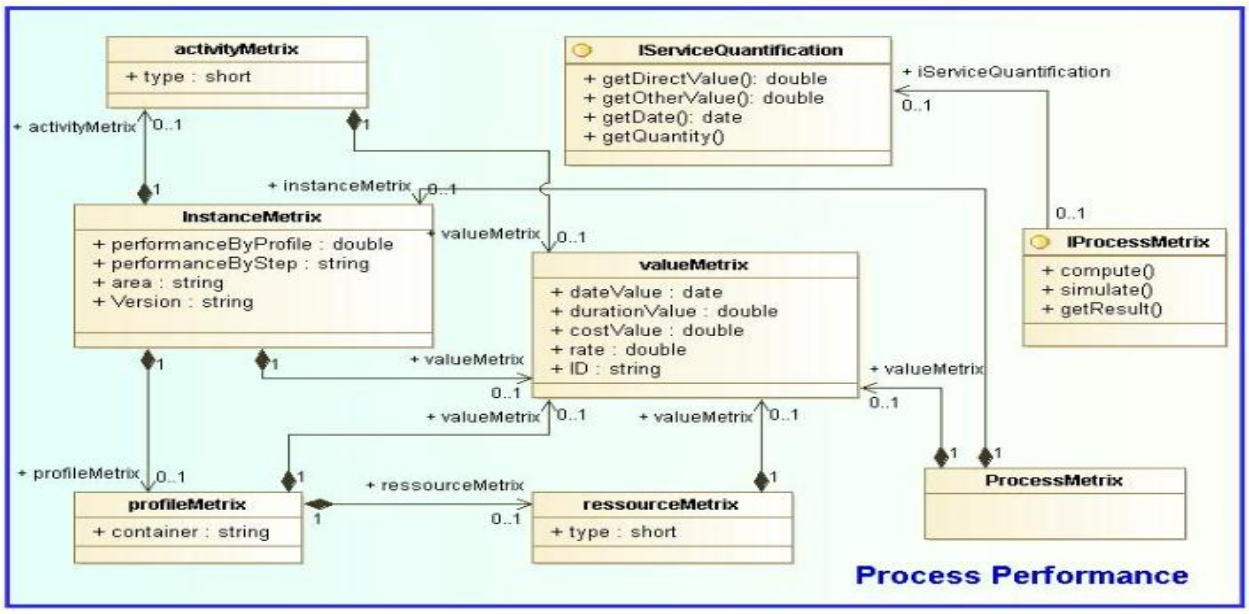
Le lien entre ressource et la performance est une combinaison complexe. La valeur (quantité, qualité et cout) d'une ressource varie selon plusieurs paramètres :

- temps
- zone
- marché
- climat
- etc.

La quantification d'une ressource n'est pas toujours exacte, le système doit créer un service dynamique capable de fournir une valeur à un moment donné selon les paramètres existants.

Chaque ressource utilisée dans la définition de processus devrait implémenter un service capable de fournir au système une valeur lors de la mesure.

Tableau 4.12: Pattern Process performance

Process Performance
Intention : présente les données et les interfaces utilisées pour générer des mesures de performances.
Indications d'utilisation. Service BPM, service Business Intelligence
Structure 
Participants principaux - IServiceQuantification : Interface qui assure la quantification des ressources - IProcessMetrix : service qui assure la mesure - valueMetrix : définit les données de mesure de base .

IV.6 Dualité métier - processus

Dans le SI, on définit un métier comme étant les traitements principaux effectués par le système pour produire ; ceci nous conduit aux processus métiers. D'après cette définition, les autres processus (pilotage, support, gestion des ressources) ne sont pas considérés comme principaux et ne reflètent pas l'activité de l'organisation, ce qui est faux :

- en décomposant l'organisation, un processus non métier de l'entité globale devient métier de sous système
- la compétitivité d'une organisation sont liés généralement à ses sous processus (car la plupart de mode de production est normé et utilise les mêmes technologies avec la mondialisation)
- le processus de pilotage est le garent de l'existence de l'organisation (la tête) : ce processus représente alors le métier de commande et de sauvegarde

Selon les propriétés des activités principales et activités support, un métier implique toujours un processus et vice versa. Cette complémentarité est relative aux organisations du système, l'espace et le temps (Figure 4.14) :

- pour définir un métier, on trace un processus ⇔ pour maitriser un processus, on analyse un métier
- pour améliorer un métier, on mesure un processus ⇔ pour analyser un processus, on définit un métier

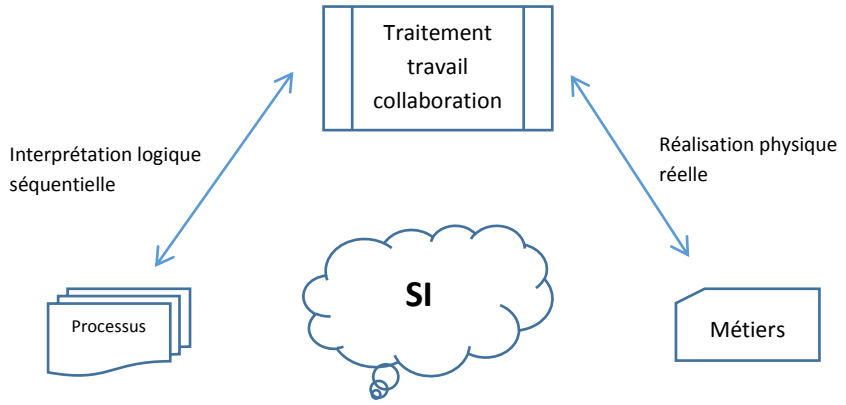


Figure 4.14 : Concept processus/métier dans un SI

Dans le domaine du SI, un processus représente le modèle logique et discret du métier et ses instances imitent la réalisation physique des tâches en utilisant l'outil BPM et Workflow.

IV.7 Conclusion

Le trio processus, services et workflow défini dans un pattern BPM représente un socle flexible pour organiser un système d'information évolutif. Le responsable métier peut auditer et mis à jour autant que possible le tableau de bord pour mieux contrôler le système.

Avec le concept « discrétisation » du métier et le principe d'amélioration continue défini dans la norme qualité, la gestion de processus métier offre des solutions réutilisables telles que le moteur workflow, la modélisation BPMn ainsi que des processus de contrôle.

L'utilisation d'une telle solution dans une organisation présente un grand risque à cause de la lenteur de maitrise des processus : il se peut que le nombre de cycle d'amélioration suive une courbe exponentielle pour stabiliser le processus car on ne maitrise, ni les ressources, ni l'organisation ni les métiers.

Dans le dernier chapitre de cette étude, nous allons utiliser le système BPM couplé par une architecture SOA avec un concept de raisonnement à partir de cas afin qu'on puisse maitriser l'évolution du système. Pour mener à bien une étape de prise de décision, le système a besoin de disposer un outil pour gérer ces connaissances.

L'ensemble des expériences passées est représenté sous forme de base de Cas, où un Cas est un couple descripteur du problème et de sa solution. Ce type de raisonnement est la genèse de tous les concepts design pattern.

CHAPITRE 5 : Concept Quanta BPM

V.1 Introduction

Jusqu'à ce chapitre, on a considéré les opérateurs humains dans notre modélisation comme étant des simples ressources utilisées par le système pour réaliser son objectif. Cette limite a été imposée pour simplifier l'étude du système en analysant uniquement l'aspect technique et structurel.
Mais en réalité l'impact de facteur humain peut accélérer ou freiner l'évolution du système même si la production est totalement maitrisée et automatisée, le pilotage et la relation avec le monde extérieur est toujours en relation direct avec l'action humaine.
La plateforme Quanta BPM qui est basée sur le management par processus et l'architecture SOA offre une autre opportunité pour tenir en compte l'expérience du système ainsi que l'expérience humaine dans la démarche de la mise en place et l'amélioration continue du SI.

V.2 Quantification du système

Définition 5.1 :

Quantum représente une quantité minimale d'une grandeur physique pouvant séparer deux valeurs de cette grandeur.

Définition 5.2 :

La théorie des quanta stipule que certaines grandeurs physiques ne varient pas de façon continue mais passent par des valeurs discontinues (discrètes) correspondant chacune à un nombre entier de quanta.

Le concept quanta dans cette étude concerne la logique et le principe de conception dans un outil de gestion de système d'information. Il se définit avec les hypothèses suivantes :

- le domaine et les composants de l'outil (SI) sont élastiques : accepte des changements
- les grandeurs principales qui constituent l'outil est quantifiable
- chaque changement et traitement doivent suivre des processus (quantification des tâches)
- un même traitement peut avoir plusieurs définitions dans un temps donné (versionnement de processus)
- les données sont cohérentes et structurées mais sa valeur (interprétation) dépend de la nature (connaissance) de l'observateur et l'instant de mesure

Tableau 5.1: Résumé des quantums du système d'information

Catégorie/ Classe	**N3 : Donnée**	**N2 : Information**	**N1: Connaissance**
Organisation	Système d'information	Indicateurs	Pattern, Cas
Organigramme	Hiérarchie	Responsabilité	Objectif
Métier	Traitement	Activités	Processus
Ressource	TIC, matériel, logiciel etc.	Réutilisation	Adaptation
Acteur	Ressource humaine	Acteur qui a un intérêt	Membre de cockpit
Outil SI	Message/Donnée/ Composant	Evénement/TDB/ Framework	BPM/SOA

V.2.1 Domaine du système d'information

La figure 1.6 spécifie que dans le système d'information, on a trois éléments qui agissent sur l'outil de gestion de SI :

- acteurs : autre système, ressource humaine
- ressources : humaine, matériel, TIC
- processus : contrôle, pilotage, métier, support, ERP

L'homme peut implémenter plusieurs interfaces dans le domaine de système d'information :

- employé : ressource
- décideur : pilotage
- client : système extérieur

Ce rôle prépondérant de l'homme dans l'environnement est toujours le facteur clé de la réussite du système. L'outil de gestion du système d'information doit accompagner et diriger l'acteur humain dans ses diverses activités pour atteindre l'objectif ; c'est la raison d'être du système.

V.2.1.1 Fondement d'une organisation

L'organisation est le résultat d'une interaction des acteurs qui fixent des objectifs et les réalisent ensemble. Une organisation passe par plusieurs phases avant de rentrer dans la phase d'exécution :

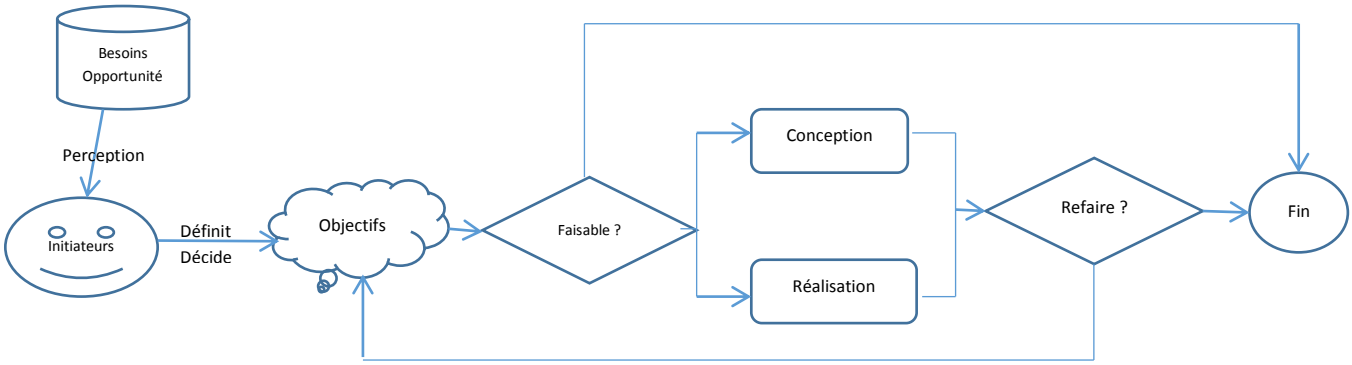


Figure 5.1 : Cycle de vie d'une organisation

Le concept organisation est purement naturel :

- si un groupe d'individu a le même besoin ou identifie la même opportunité, ils se lancent dans la définition des objectifs et de projet
- s'ils ont les moyens nécessaires et si le projet est faisable, il passe dans la création de l'organisation réelle pour concevoir et réaliser les traitements (métiers) afin de produire
- à chaque fois qu'ils peuvent prendre de décision, ils peuvent redéfinir les objectifs ou mettre à fin l'association

Les initiateurs sont les premiers éléments des organes centraux de l'organisation, ils s'ajoutent petit à petit des autres acteurs : la direction, les responsables, les cadres, les mains d'œuvres :

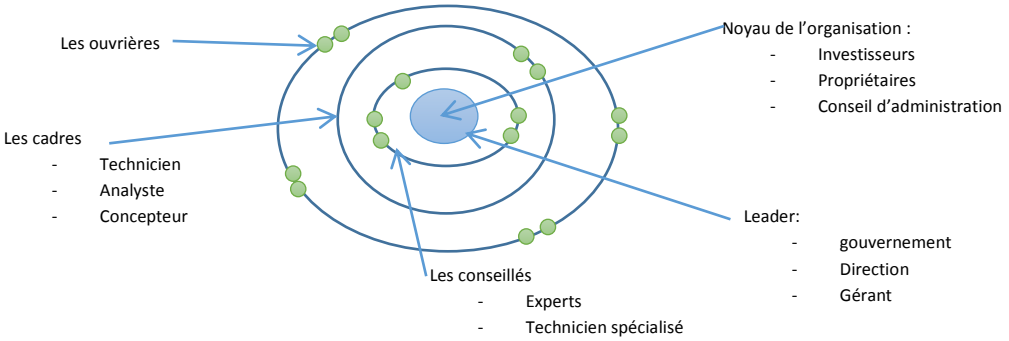


Figure 5.2 : Concentration des acteurs sur l'organisation

La figure 5.2 démontre que chaque acteur gravite autour du noyau selon l'orbite de son groupe d'appartenance.
La force gravitationnelle qui relie les acteurs avec le noyau est « l'intérêt » : plus cette force est forte, plus l'acteur est proche du noyau : l'intérêt des initiateurs de l'organisation se confonde avec l'intérêt de l'organisation d'où l'indivisibilité du noyau.

V.2.1.2 Intérêts des acteurs

Dès la naissance du système, l'intérêt est la force qui surgissent sur les initiateurs (Figure 5.2) pour qu'ils puissent s'unissent afin d'atteindre leurs objectifs. Tant qu'il y a des intérêts communs, le système continue à survivre et même s'élargit.

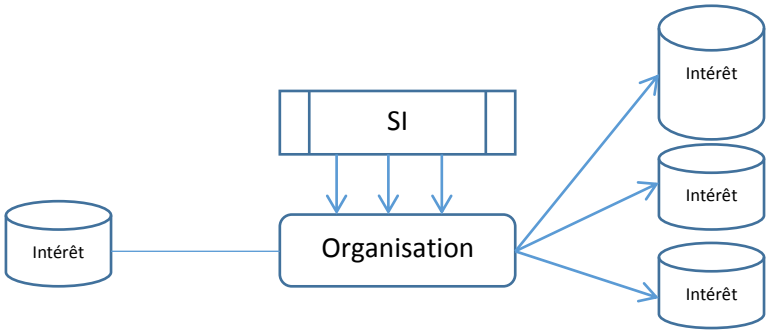


Figure 5.3 : Multiplication des intérêts

Une organisation peut être assimilée à un champ de culture d'intérêt (Figure 5.3), si le champ est bien traité par l'outil de système d'information et elle possède les ressources suffisantes, on peut espérer de bonnes récoltes.

V.2.1.3 Interaction entre systèmes

Au cours de son cycle de vie, le système interagit avec d'autre système de même manière que les pôles magnétiques :

- attraction des pôles opposés
- répulsion des pôles identiques

Un acteur peut être assimilé aussi à un système indépendant, il peut rester travailler dans une organisation (attraction) si l'intérêt offert par l'organisation lui est encore meilleur par rapport aux opportunités offertes par d'autre système.

V.2.1.4 Quantification des acteurs

Selon la Figure 5.2, la quantification des acteurs suit l'organigramme et la structure du système. Un acteur peut passer d'un niveau à l'autre selon les forces données par l'acteur ou la force libérée par le noyau du système.
L'unité de mesure dans cette quantification est donc le niveau entre deux hiérarchies

V.2.2 Outil Quanta BPM (Q-BPM)

D'après la définition de l'organisation défini dans le paragraphe V.2.1.1, un moteur BPM bien structuré et évolutif ne suffit pas pour maitriser le SI. Q-BPM définit une autre dimension de traitement pour réagir selon les échanges et les interactions des acteurs avec le système :

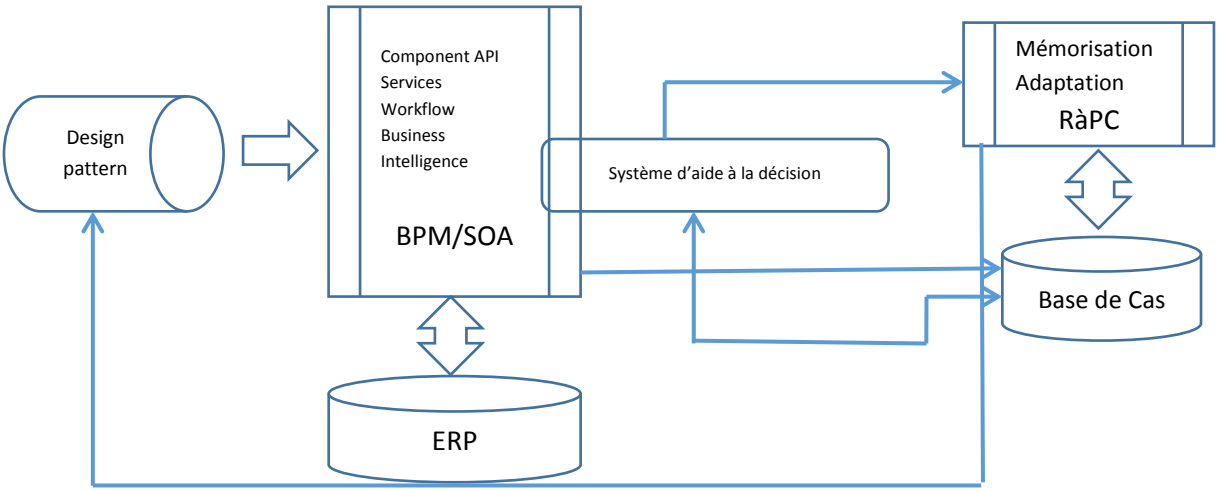


Figure 5.4 : Architecture de Q-BPM

En tenant compte de la Figure 5.3 et le Paragraphe I.2.4.3 (Processus décisionnel), l'outil Q-BPM est composé essentiellement de deux plateformes :

- corps : crée des valeurs, des patterns et des connaissances (BPM – SOA)
- tête : mémorise et gère les connaissances (RàPC)

Le système à raisonnement par Cas permet d'acquérir des différentes connaissances des acteurs (interne et externe du système) et des systèmes afin de les réutiliser dans la prise de décision et la conception des nouveaux patterns.

Avec un composant intelligent, le système peut résister aux chocs et aux interférences du milieu extérieur. La qualité des décisions, des modèles et des patterns s'améliore en utilisant les expériences du passé et cette boucle d'apprentissage ne fait qu'améliorer la performance du système.

V.2.3 Quantification des connaissances

Définition 5.3 :

Pour un système d'information, une connaissance est une structure de données liée à un concept et/ou une résolution (contexte) qui est testée et validée et qui donne au système des compétences sur le genre de situation.
La modélisation de la connaissance revient donc à la modélisation de son origine, ses actions intermédiaires et ses résultats.

Définition 5.4 :

Un Cas est une façon de représenter une situation ou un état avec la définition du problème (ou défi) à résoudre et la solution correspondante.
Avec l'expérience et l'habitude, un système intelligent restructure ses informations et les mémorisent dans une base de données pour qu'il puisse le réutiliser si le Cas se reproduise.

V.2.3.1 Système RàPC

Le Raisonnement à Partir de cas (RàPC) est un paradigme de raisonnement qui s'appuie sur la remémoration de problèmes passés résolus, appelés les cas sources, pour résoudre de nouveaux problèmes, appelés les problèmes cibles.

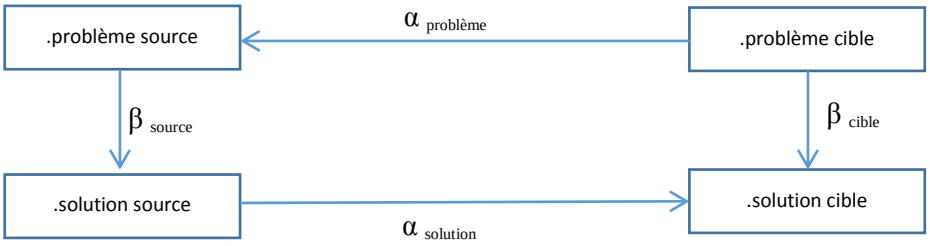


Figure 5.5 : Carré d'analogie

Le carré d'analogie défini par la figure 5.5 montre qu'un Cas source est sélectionné à partir des valeurs de descripteurs du problème cible et la mesure de similarité alpha. Les relations de dépendance Béta entre les valeurs de descripteurs de problème ciblent et la valeur de descripteurs solution permettent de mettre en évidence les descripteurs solution qui doivent être adaptés (car les dépendants de descripteurs problèmes source sont différents des descripteurs problèmes cibles).

Si une valeur de descripteur source dépend d'une valeur de descripteur problème, une modification de la valeur du descripteur problème entraînera une modification "analogue" à la dépendance du descripteur solution correspondant. Cette connaissance est nécessaire pour l'adaptation.

En fonction de ces dépendances et des écarts alpha constatés à corriger, l'adaptation permet de proposer une solution cible candidate qui pourra être vérifiée par la vérification de sa conformité aux dépendances particulières qui pourraient exister entre problème et solution cible.

a. Cycle de vie du système RàPC

Le RàPC a été présenté comme une solution au goulet d'étranglement de l'étape d'acquisition des connaissances grâce à l'utilisation de connaissances expérimentales, les Cas, qui sont plus faciles à recueillir.

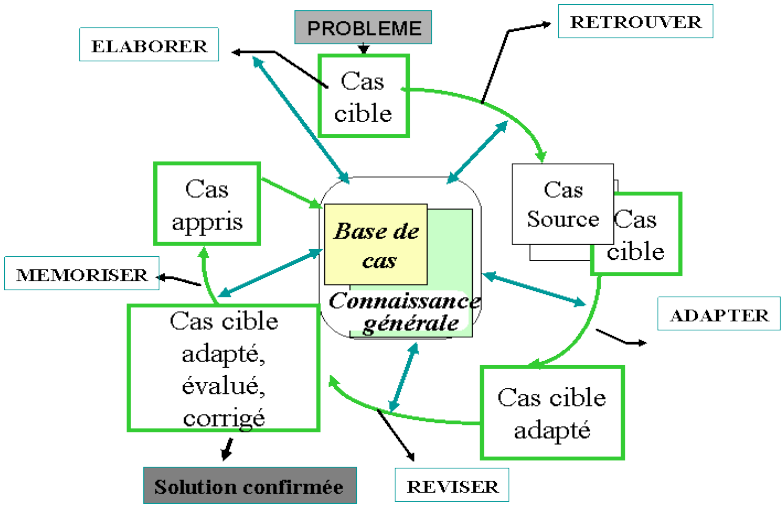


Figure 5.6 : Cycle d'étapes du Raisonnement à Partir de Cas

Le RàPC peut être modélisé (selon la Figure 5.6) par un cycle constitué de cinq étapes :

- élaboration,
- remémoration,
- adaptation,
- révision
- mémorisation

gravitant autour d'une base de connaissances du domaine d'application. Chacune des étapes du cycle mobilise ces connaissances pour supporter la recherche de la solution du problème cible.

b. Elaboration - Révision

L'élaboration d'un nouveau Cas consiste à définir la description du problème (l'état ou l'ensemble des valeurs d'un objet en un instant donné) pour permettre la recherche d'un Cas dont la solution sera la plus proche adaptable. L'adaptabilité d'un Cas se mesure à "l'effort" d'adaptation qui sera nécessaire pour que la solution passée (du Cas source) puisse servir de base à la solution courante (du Cas cible).

La méthode générale consiste à compléter ou filtrer la description d'un problème en se fondant sur la connaissance du domaine pour inférer tout ce qu'il est possible à partir d'une description éventuellement incomplète, et pondérer (mettre des poids) les descripteurs en fonction des dépendances identifiées entre les descripteurs problème cible et les descripteurs de la solution recherchée.

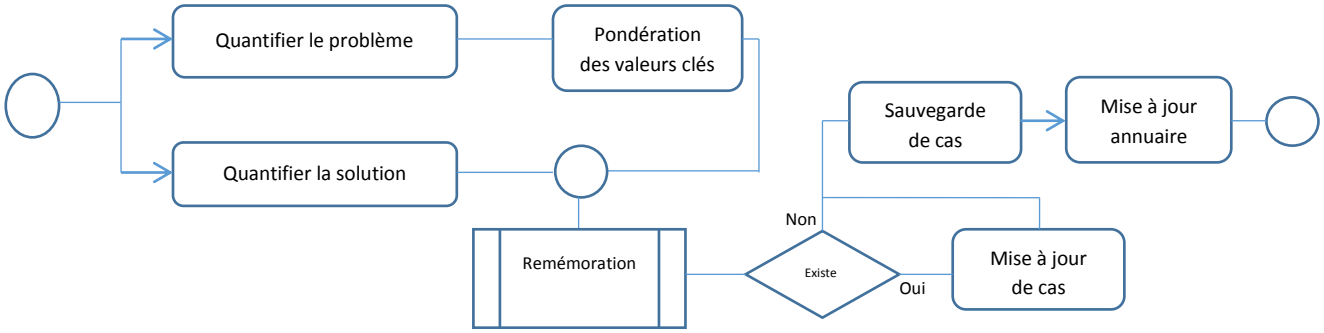


Figure 5.7 : Processus d'élaboration de Cas

En utilisant le processus de la figure 5.7, l'élaboration d'un nouveau Cas devrait être automatique, à chaque évènement, une nouvelle solution.
Après l'indexation, le processus fait appel à un sous processus de remémoration pour vérifier si une autre version du Cas existe déjà ou non avant de sauvegarder le nouveau Cas.

c. Remémoration

L'étape de recherche de Cas sources similaires est le plus important dans le cycle du système RàPC. Le Cas source qui sera choisi sera le Cas ayant la description de problème la plus proche possible de la description du problème cible dans la classe de solution la plus représentée.
En pratique, il est nécessaire de définir une mesure de similarité qui tiendra compte de l'influence d'une variation d'une valeur de descripteur problème sur une variation de valeur d'un descripteur solution.

Cette mesure qui fait partie intégrante de la connaissance est la base de la qualité de Cas retrouvé par ce processus de remémoration et donc la qualité du système tout entière.

Comme chaque élément du Cas est déjà quantifié, la mesure consiste à varier la valeur de la pondération selon la description du nouveau problème et calculer la différence (distance) :

$$d = \frac{\sum p_i \times d_i}{\sum p_i} \qquad (5.1)$$

Où :

di : distance élémentaire
pi : poids qui portent la connaissance sur l'importance de l'influence du descripteur i sur la solution.

Le calcul de la différence s'applique à l'ensemble des Cas de même catégorie (processus défini dans la figure 5.8) et pour trouver le Cas similaire, on utilise l'algorithme qui cherche le Cas qui a la distance minimale dans la base de Cas.

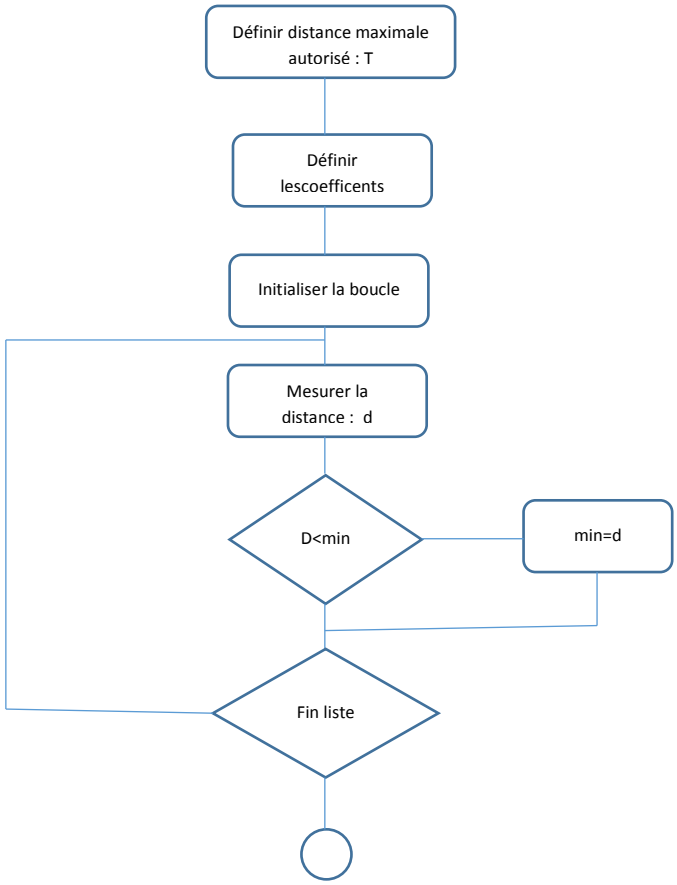


Figure 5.8 : Processus de remémoration

La performance de ce processus (Figure 5.8) sera améliorée en présélectionnant les Cas selon les informations dans son entête (meta data) et en sauvegardant aussi certaines mesures.

d. Adaptation

L'adaptation termine l'inférence analogique en calculant quelle pourrait être la solution au problème du Cas cible inspirée de la solution du cas source le plus similaire.

Les règles d'adaptation de la solution s'expriment en fonction des écarts relevés entre les descriptions des problèmes cible et source.

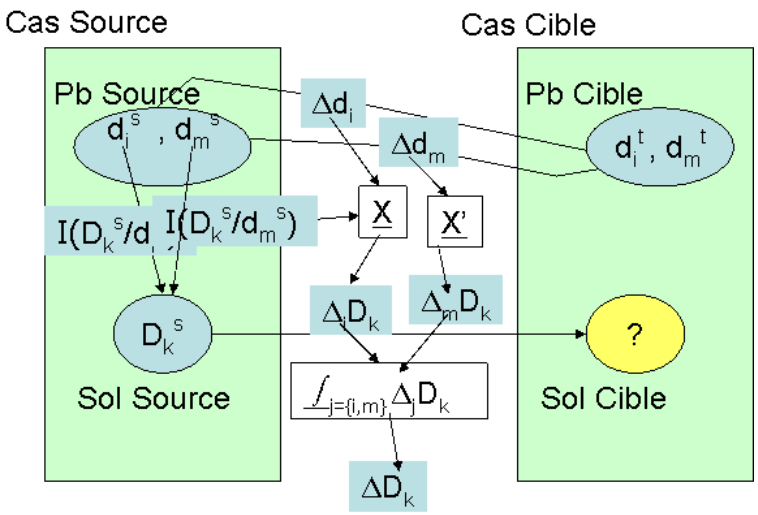


Figure 5.9 : Opérateurs d'adaptation

Le calcul d'un descripteur solution cible à partir de descripteurs solution est selon l'influence des écarts sur les descripteurs problème correspond aux formules suivantes (Figure 5.9) :

$$D_k^t = D_k^s \pm \Delta D_k$$
$$D_k^t = D_k^s \pm \int_{i=\{j,m\}} (I(D_k^s/d_i^s) \underline{X} \Delta d_i) \qquad (5.2)$$

Où :

Δd_i: écart entre les valeurs de descripteurs problème selon la fonction d'appariement

$I(D_k^s/d_i^s)$: rapport entre la valeur d'un descripteur problème avec un descripteur solution

$\underline{X}$: opérateur de mise en œuvre de l'influence en fonction de l'écart observé sur le problème

$\int_{i=\{j,m\}}$: opérateur de sommation des effets des différentes influences appliquées aux écarts de problème constaté pour une solution donnée

$\pm$: opérateur d'ajout des effets des écarts problème sur le descripteur solution source pour fournir une valeur candidate pour le descripteur solution cible

L'ensemble des descripteurs sont déjà quantifiés (Paragraphe 5.2), le calcul des nouveaux descripteurs de la solution cible devient une projection matricielle selon les écarts trouvés :

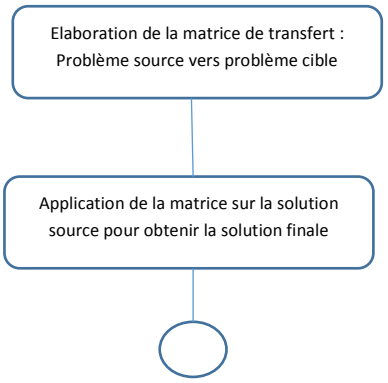


Figure 5.10 : Processus numérique d'adaptation

V.2.3.2 Représentation des connaissances

Pour construire les processus par lesquels une organisation devient apprenante, axée sur le savoir, il faut des méthodes de modélisation des connaissances. L'objectif est d'identifier et de structurer les connaissances en une représentation schématique pour les rendre visibles, manipulables, compréhensibles et communicables.

D'après la Figure 1.4 (Evolution de la compréhension), la connaissance (faits, concepts, procédures, principes, habiletés) est d'abord une information qui est interprétée par une activité mentale avec un certain degré de compréhension.

Les schémas jouent un rôle primordial dans la construction et la modélisation des connaissances :

- ils guident la perception,
- ils servent à la mémorisation de séquence de traitement, rendent la compréhension possible par la confrontation des schémas existants avec des évènements nouveaux,
- ils servent de soutien à la résolution de problèmes et à l'accomplissement de tâches complexes.

Le concept BPM est un cas particulier de la représentation graphique des connaissances du métier en processus.

Il existe plusieurs représentations graphiques de la connaissance :

- arbres sémantiques et cartes conceptuelles.
- réseaux sémantiques et graphes entités-relations.
- algorithmes et ordinogrammes.
- diagramme causal et arbres de décision.
- arbres de déduction.
- modèles orientés objets

Le choix du type de modèles à construire influe nettement sur la façon dont les problèmes sont abordés et les solutions élaborées.

Le problème de la représentation des connaissances est un problème qui est né avec l'intelligence artificielle et qui se pose toujours. Dans le domaine de développement d'application, les représentations des connaissances par objets (RCO) offre un avantage sur l'intégration des modèles dans la structure du système.

a. Principe de représentation par Objet

Ce système de représentation est fondé sur la théorie des schèmes qui possèdent trois catégories :

- schèmes déclaratifs ou conceptuels
- schèmes procéduraux,
- schèmes conditionnels stratégiques

On distingue donc deux types de principes :

- les principes d'opération, qui indiquent quand appliquer certaines procédures,
- et les principes relationnels, qui associent deux ou plusieurs concepts, notamment les relations de cause à effet

Le concept objet permet de définir ou encapsuler les propriétés de la connaissance (conceptuellement), les actions que l'on peut manipuler et ses relations avec les autres connaissances dans un schéma ou modèle.

b. Types de connaissances

La première classification des connaissances est basée sur sa propriété d'existentielle :

- abstraite
- fait

Le type « fait » est l'instanciation de type « abstraite » en complétant toutes les propriétés de la connaissance avec des valeurs concrètes.

La deuxième classification est basée sur la nature des connaissances « abstraites » :

- concept : décrive la nature des objets d'un domaine (le quoi)
- procédure : décrive des ensembles d'opérations permettant d'agir sur les objets (le comment)
- principe : est une énoncé permettant de décrire les propriétés des objets, d'établir des liens de cause à effet entre les objets (le pourquoi) ou de déterminer dans quelles conditions appliquer une procédure (le quand); les principes prennent le plus souvent la forme "si telle condition. Alors telle condition ou telle action".

La troisième classification définit les trois instances de trois types de connaissance « abstraite » :

- exemples : sont obtenus en spécifiant les valeurs de chacun des attributs d'un « concept », ce qui donne un ensemble de faits décrivant un objet bien précis
- traces : sont obtenues en spécifiant les variables de chacune des actions qui composent une procédure, ce qui donne un ensemble d'actions particulières bien précises, une trace d'exécution
- énoncés : sont obtenus en spécifiant les variables d'un principe, ce qui donne un lien de cause à effet entre les propriétés d'objets particuliers ou entre des propriétés d'un objet particulier et une action précise à effectuer

c. Relation entre connaissances

Il existe 7 moyens d'implémenter des relations entre deux connaissances :

- instanciation : relie une connaissance abstraite à un fait
- composition : relie une connaissance à une de ses composantes ou de ses parties constitutives
- spécialisation : met en relation deux connaissances abstraites de même type, dont l'une est « une sorte de », un cas particulier de l'autre. Autrement dit, la seconde est plus générale ou plus abstraite que la première.
- précédence : relie deux procédures ou deux principes dont le premier doit être terminé ou évalué avant que le second commence
- intrant/produit : relie un concept et une procédure. Du concept vers la procédure : le concept est un intrant de la procédure. De la procédure vers le concept : la procédure a pour produit le concept
- régulation : relie un principe vers une autre connaissance abstraite qui peut-être un concept, une procédure ou un autre principe. Dans le premier cas, le principe définit le concept par des contraintes à satisfaire (contraintes d'intégrité) D'un principe vers une procédure ou un autre principe, le lien de régulation signifie que le principe contrôle de l'extérieur (régit) l'exécution d'une procédure ou la sélection d'autres principes
- application : relie un fait à une autre connaissance ; ce fait est déterminé par une connaissance générique et il s'applique à des connaissances spécifiques d'un autre domaine d'application

d. Gestion de connaissances dans le SI

L'ingénierie des connaissances est un système complexe ainsi que son exploitation dans le domaine de l'intelligence artificielle. Mais dans cette étude, on se concentre uniquement dans la nature du système d'information et l'utilisation de ses connaissances dans le processus de prise de décision.

Le SI possède quatre groupes de connaissances à gérer si on se réfère de la Figure 1.2 :

- processus (concept, principe)
- service (procédure)
- décision (fait)
- ressources (fait, concept)

Chaque étage du module du SI lance des événements au système RàPC pour élaborer de nouveau cas à chaque fois qu'on approuve (perception de l'acteur) un de ces éléments. Le système de pilotage peut définir des nouveaux objectifs qui entrainent des nouveaux problèmes de conception au niveau chaque étage et cela implique une recherche de solution et à implémenter.
La figure suivante montre le cycle de génération de Cas via les différents modules du système d'information :

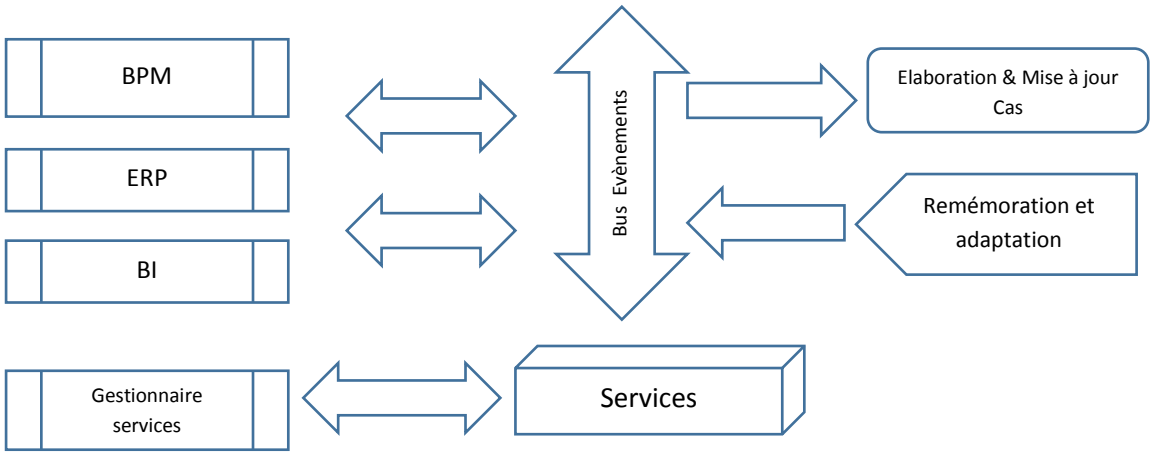


Figure 5.11 : Echange de cas et des événements dans le SI

V.2.3.3 Connaissances collectives du SI

Dans le système d'information, la connaissance collective représente sa faculté à s'adapter, ses capacités et ses moyens pour fonctionner à un moment donné. Il englobe les Cas enregistrable sur l'ensemble des données et des informations qui circule dans le système.
Q-BPM permet déjà la gestion des données et de ses contraintes selon le type de conteneur et des ressources disponibles. L'élaboration de Cas revient donc à restructurer ses données selon les besoins du système décisionnelle et RàPC.

a. *Axe d'analyse du SI*

Le management par processus utilise 3 axes (Paragraphe IV.3) :

- organisation des acteurs
- production pilotée par des processus
- pilotage par les expériences

La modélisation des connaissances du système est aussi organisée selon ces axes. Mais comme le système évolue dans le temps et dans l'espace, ces deux autres dimensions (temps et espace) sont aussi ajoutées à la modélisation (Figure 5.12).

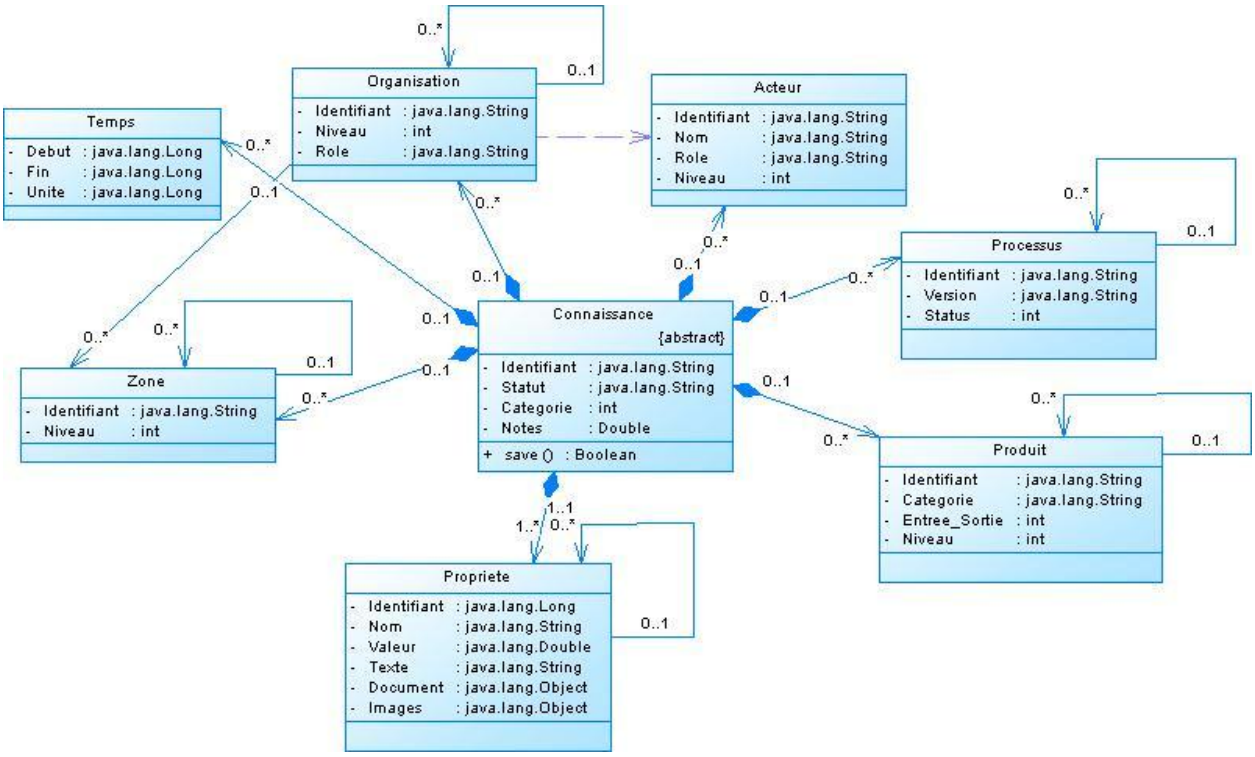


Figure 5.12 : Les dimensions des connaissances dans le SI

i. Axe temps

La définition d'une connaissance peut nécessiter la déclaration de la plage de temps concerné par le Cas. Ces informations sont de critères d'indexation et de validation d'un Cas lors de la phase de remémoration.
Les informations sur la dérivée du temps par rapport aux autres dimensions sont classées dans la partie de données, propriétés de la connaissance.

ii. Axe zone, organisation, acteur

La zone, l'organisation et les acteurs représentent des contraintes spatiales pour la connaissance. Ils sont complémentaires et interdépendants entre eux pour définir le responsable, la cible ou l'acteur de la connaissance. Le statut de ces dimensions est limité dans le temps et la localisation de connaissance par ces axes force l'utilisation de l'axe temps pour s'assurer de sa validité.

L'attribut niveau définit l'importance ou gravité de l'information par rapport aux métiers du système.

iii. Axe processus

Avec cette information, le Cas peut faire référence à des processus existant dans le système. Les mesures et les indicateurs livrés par le processus font parties de la liste des propriétés mais cet axe définit seulement quels sont les processus concernés.

iv. Axe produit

Du point de vue fonctionnel, le système utilise le produit d'autre système pour en crée d'autre produit. Cet axe permet de relier le Cas à un produit d'entrée ou sortie du système.
La catégorie et le niveau permet de classifier les produits selon ses impacts dans le bon déroulement du système.

v. Axe propriétés

Cet axe est le seul axe qui permet d'évaluer et de mesurer un Cas, les autres dimensions servent généralement à indexer et à définir les connaissances.

Il contient spécialement de valeurs des indicateurs, et les objets à remémorer : technique, principe, situation, etc....

b. Nature des connaissances du SI

Autre ces dimensions, la connaissance est aussi caractérisée par sa catégorie. Chaque catégorie spécifie la raison d'être du Cas :

- savoir-faire
- problème
- décision
- solution

À partir d'un événement quelconque, le système détecte un problème (qui peut être lié à une ancienne solution ou un savoir-faire existant) qui nécessite l'adaptation d'une solution. Après des études, on élabore une solution qui implique la génération des nouveaux savoir-faire :

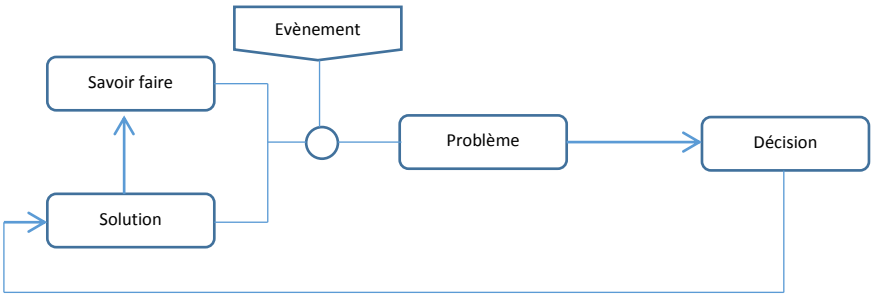


Figure 5.13 : Cycle de vie des connaissances

i. Savoir-faire

C'est un ensemble de Cas qui désigne des capacités techniques pour réaliser un métier. Il est composé des connaissances abstraites et réelles qui ont de durée de vie et d'importance élevée par rapport aux autre catégories. Il est constitué essentiellement des procédures, procédés, processus et outils stratégiques gérés par le noyau du système.

ii. Problème

Il spécifie un ensemble de Cas qui mémorise une situation réelle (le système tout entier ou ensemble de sous-système) et les Cas cible qui devraient être mis en place. Une fois définies les données, le système calcule l'écart entre la réalité et la cible et élabore une connaissance de catégorie problème.

En général, on ne peut pas définir un Cas problème si on ne maitrise pas les informations sur les états du système concerné par l'évènement.

iii. Décision

Pour Q-BPM, une décision est une liste de recommandation pour pouvoir ajuster des paramètres et pour corriger une partie ou tout l'écart défini pendant l'analyse du système.
Cette catégorie définit l'objectif, c'est-à-dire l'ensemble des états à atteindre après la mise en place de la solution.

iv. Solution

Il définit une mise à jour d'une partie de savoir-faire du système ou intègre un tout nouveau pour corriger l'état du système vers l'état défini par la décision.
L'efficacité d'une solution se mesure avec les deux propriétés suivantes :

- la durée de la mise en place
- les nouveaux écarts après déploiement

V.3 Modélisation de l'outil Quanta PBM

Q-BPM est une plateforme d'application et de procédure ; pour comprendre son fonctionnement et ses interactions, on va étudier son modèle via les modèles UML suivants :

- cas d'utilisation : interaction entre acteurs et le système
- diagramme de composant : les interactions entre chaque module
- diagramme de séquence : la séquence de traitement dans le temps

V.3.1 Cas d'utilisation

Dans le système, il y a des catégories d'utilisateurs qui ont des rôles spécifiques pour faire fonctionner le système dans l'organisation :

Tableau 5.2 : Liste des profils utilisateur

Profil	**Description**
administrateur Q-BPM	utilisateur « root » ; il peut tout faire dans l'outil
administrateur métier	profil propre à Q-BPM, il a pour rôle de gérer les processus
administrateur des données	profil propre à Q-BPM, il a pour rôle de gérer les ressources
administrateur système	profil propre à Q-BPM, il a pour rôle de gérer les structures du système cible
administrateur de Cas	profil propre à Q-BPM, il a pour rôle de gérer les base de Cas et les tableaux de bords
Utilisateur	profil propre à Q-BPM qui peut naviguer dans le système pour lecture seule
Administrateur de base de données	Profil interne du système qui gère les bases de données et qui travaille en collaboration avec l'administrateur des données du Q-BPM
Administrateur réseau	Profil interne du système qui gère les ressources TIC et qui travaille en collaboration avec l'administrateur système et des données du Q-BPM
Opérateur	Profil définit dans l'organisation du système qui réalisent les tâches décrit dans les processus

La figure 5.14 montre le diagramme de cas d'utilisation du module BPM qui spécifie les rôles principaux de chaque profil :

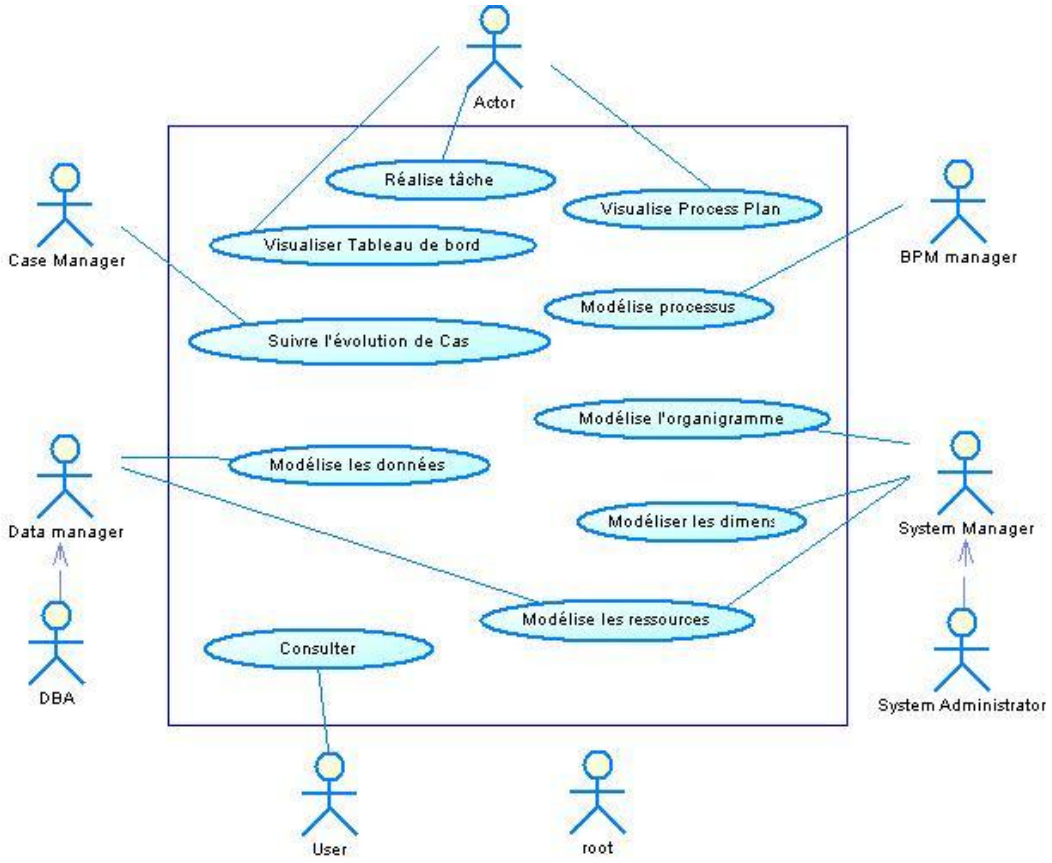


Figure 5.14 : Cas d'utilisation de Q-BPM

V.3.2 Diagramme de composant

Chaque composant de l'outil Q-BPM est une application multi-tiers, il se communique entre elles à travers d'un gestionnaire d'évènement et elles partagent les mêmes sources de données.

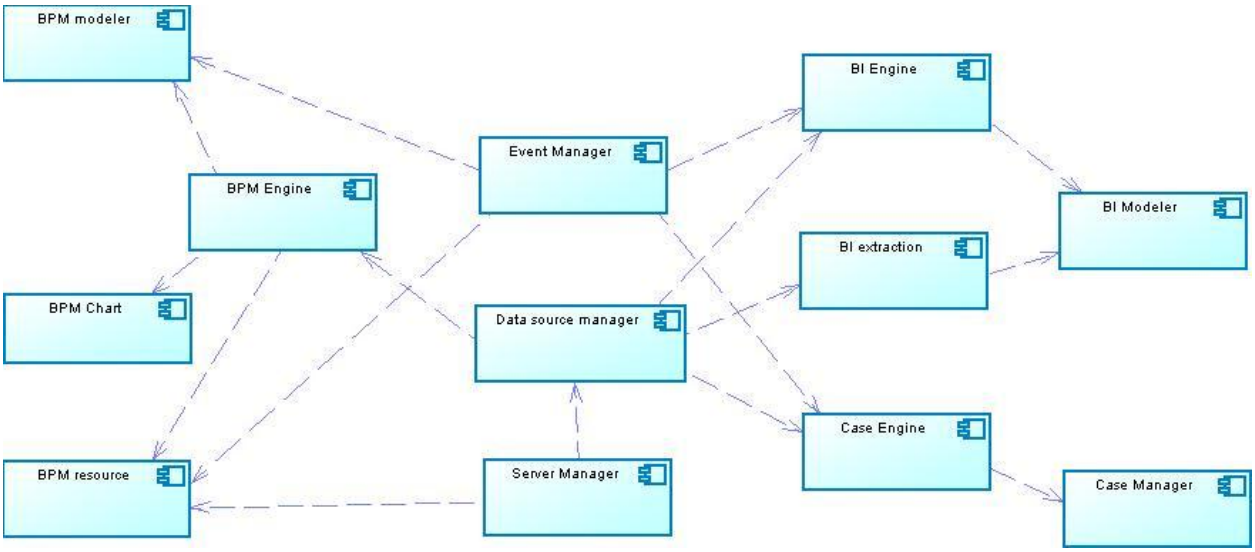


Figure 5.15 : Diagramme de composants

Il y a trois niveaux dans la conception des composants :

- niveau central : on gère les sources des données, les serveurs et les modes de gestion des évènements
- niveau intermédiaire : ces sont les « Engine », il constitue les règles de métier de base et les processus statiques pour réaliser le traitement supérieur
- niveau supérieur : contient les interfaces utilisateurs pour manipuler les données

V.3.3 Diagramme de séquence

Pendant un cycle de traitement d'information, les tâches sont divisées en quatre axes :

- modélisation
- instanciation
- extraction
- orientation

V.3.3.1 Modélisation

Dans la procédure de modélisation, on utilise trois types d'objet :

- processus
- ressource
- data source

Les acteurs concernés sont l'administrateur BPM et administrateur système ; et le traitement se résume par :

- la modélisation sur l'interface utilisateur
- l'enregistrement des données.
- contrôle
- affectation des profils
- attribution des types de ressources

La séquence du traitement manipule trois type d'objet (Figure 5.16) :

- Processus
- Ressources (profiles et service)
- Data source (données)

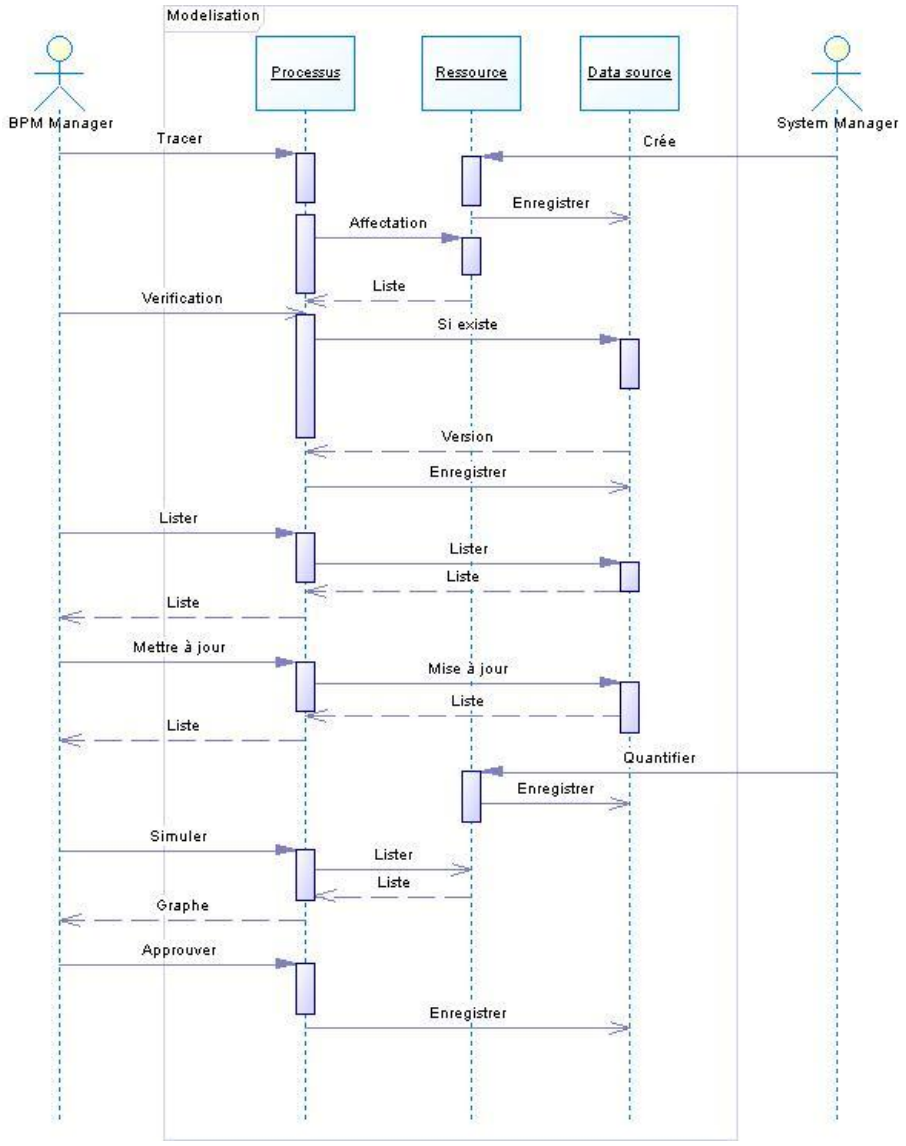


Figure 5.16 : Diagramme de séquence de modélisation

V.3.3.2 Instanciation

L'instanciation consiste à faire tourner un processus et de créer des tâches. L'attribution de tâche est définie par le profil attribué au nœud du processus. Une instance est une succession de tâches dont les patterns à suivre est le processus. Ce traitement nécessite une large compréhension du processus par l'ensemble des utilisateurs.

Les acteurs concernés par cette procédure sont l'administrateur BPM et les acteurs de l'organisation. Ces acteurs manipulent les objets de type processus, tâches, services et ressources tout au long de l'instanciation :

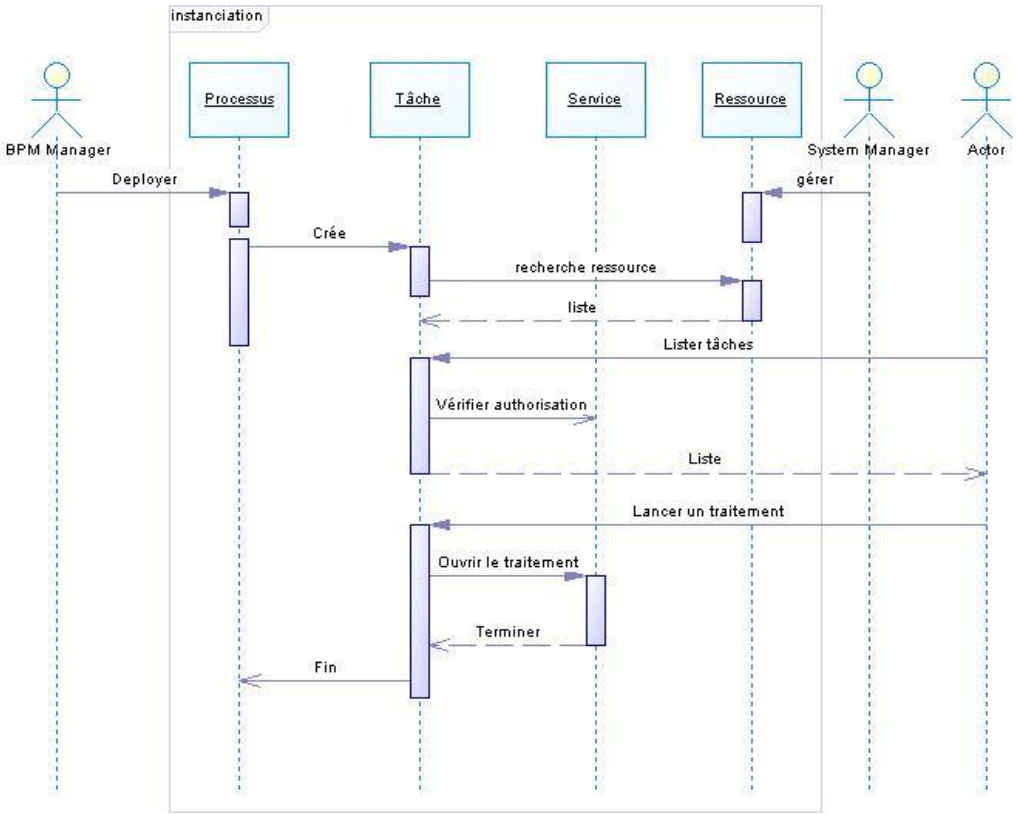


Figure 5.17 : Diagramme de séquence d'instanciation

V.3.3.3 Extraction

Cette procédure consiste à calculer les données selon les axes de dimension de l'organisation et de fournir des données au système Business Intelligence.

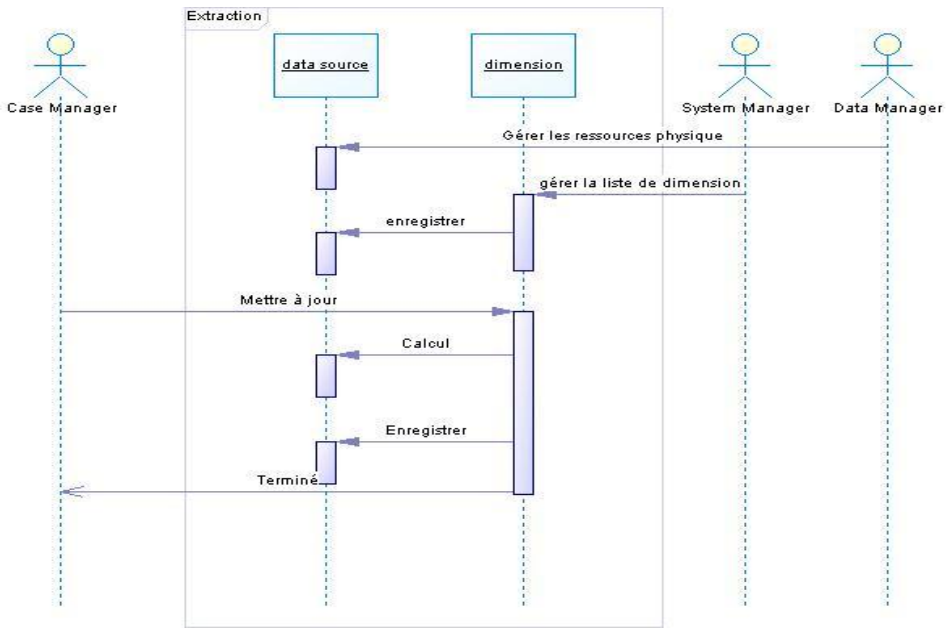


Figure 5.18 : Diagramme de séquence d'extraction

V.3.3.4 Orientation

Avec cette procédure, le système puisse collecter les Cas, élaborer les tableaux de bord et de concevoir les décisions dans le système :

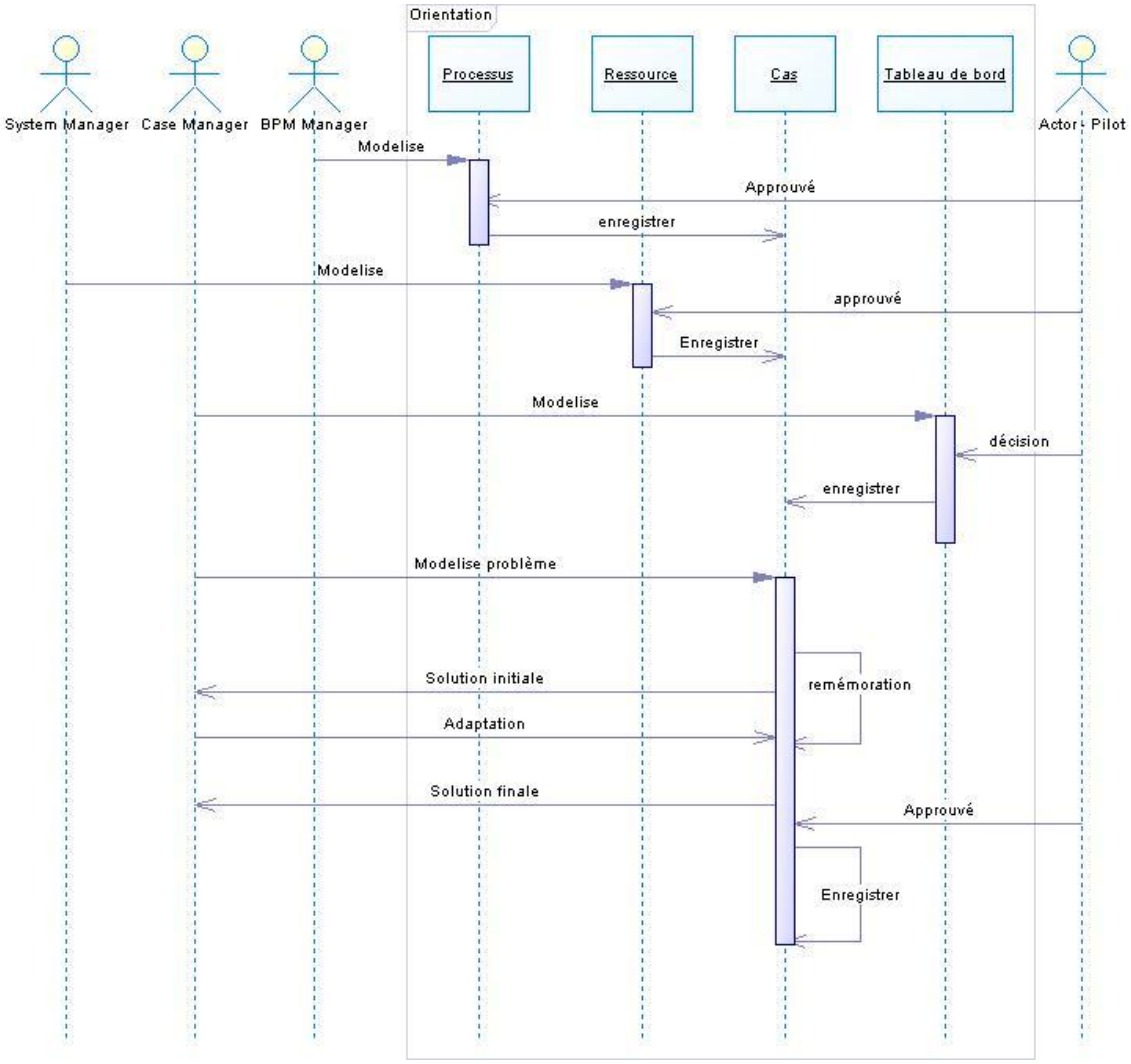


Figure 5.19 : Diagramme de séquence d'orientation

L'objet de type tableau de bord est une représentation des données KPI dynamique. Dans le cas pratique, un système de pilotage utilise plusieurs tableaux de bord issus des sous-systèmes ou axe d'analyse de l'organisation (exemple pilotage cockpit).

L'acteur qui a le profil de pilotage peuvent à la fois valider les informations dans le tableau de bord et définir les décisions et la génération des Cas de pilotage.

V.4 Noyau de l'outil Q-PBM

Le cœur ou noyau de l'outil Q-BPM est une application complète et distribuée dont l'architecture est définie par la Figure 5.4. Ce noyau comporte des applications dont la vocation est de modéliser ou de contenir d'autre module et des processus figés qui sont prouvés par des patterns :

Tableau 5.3 : Les éléments du noyau du Q-BPM

Type	**Module**	**Groupe**
API	- Framework ressource - Framework BPM - Framework BI	Interaction
Application	- modélisation des processus - quantification des ressources - gestion des flux : workflow	BPM
	- modélisation des organisations et des profils acteurs - cartographier les processus - gestion des dimensions d'analyse	BPM-Système
	- annuaire de services - gestion des bus (JBI) - Web service d'interfaçage - REST d'interfaçage	SOA
	- gestion d'extraction des données opérationnelles - affichage de tableau de bord - modélisation de décision	BI
	- gestion des Cas - adaptation de Cas	RàPC
Processus	- intégration de processus	BPM
	- changement d'organisation	BPM-Système
	- intégration de service	SOA
	- remémoration de Cas - adaptation de Cas	RàPC

V.3.1 Pourquoi un noyau statique

Dans les langages orientés objet, on est contraint de définir des éléments natifs et des méthodes natives pour pouvoir construire des objets. Par analogie, on a besoin de noyau statique pour recevoir les spécificités de l'organisation :

Les avantages de cette solution sont :

- le noyau statique est le fruit des patterns donc l'organisation hérite l'expérience des autres systèmes
- il permet de modéliser l'état initial d'une organisation
- il accélère le déploiement et la compréhension du SI

V.3.2 Module de modélisation de processus

Ce module est accessible uniquement par des administrateurs du système Q-BPM. Il permet de :

- tracer de nouveau graphe BPMn
- dupliquer de graphe existant
- paramétrer chaque nœud (service, formulaire, profil, évènement, variable)
- sauvegarder le processus en tenant compte de sa version
- rechercher et charger des processus existants

La figure suivante montre l'aspect de l'interface utilisateur pendant la phase de modélisation.

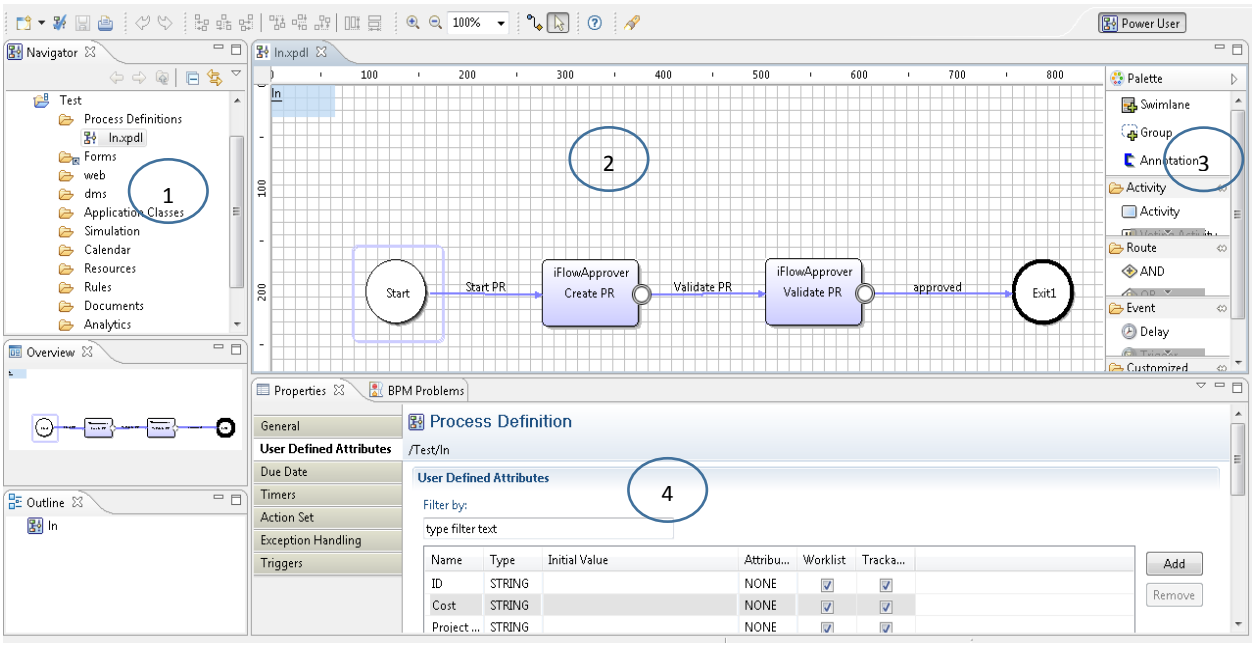


Figure 5.20 : Formulaire de modélisation de processus

1 : arborescence des processus
2 : graphe
3 : boite à outil pour construire le graphe
4 : onglet de paramétrage des nœuds

V.3.3 Module de quantification des Ressources

Ce module utilise la liste des ressources disponible dans le système et lui attribue un coût et une durée de réalisation de tâche élémentaire (si nécessaire) :

Le traitement consiste à :

- lister les ressources par catégorie : profil, matériel, logiciel, service
- modifier les paramètres
- analyser les valeurs des ressources attribuées à un processus (ou instance)

L'IHM correspond à cette phase permet d'identifier toutes les propriétés de chaque ressource (Figure 5.21).

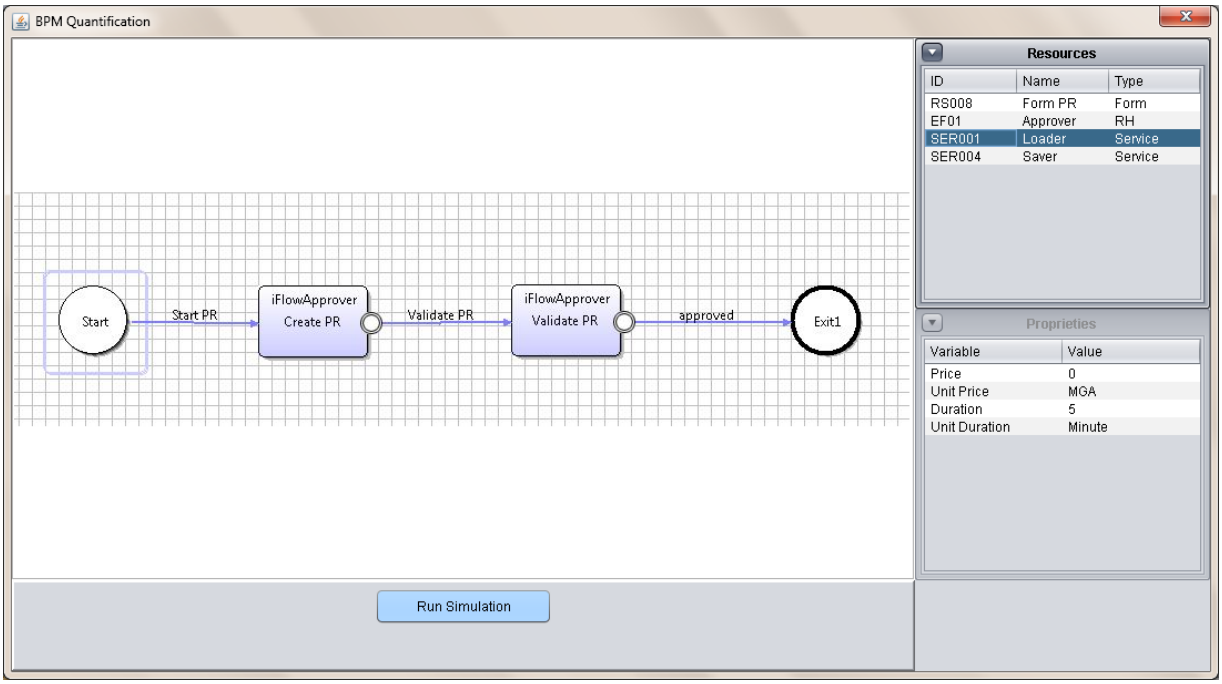


Figure 5.21 : Formulaire de quantification des ressources

V.3.4 Module de gestion de l'organisation

Pour saisir et paramétrer la structure de l'organisation dans le Q-BPM, on utilise ce module. Il permet de :

- présenter sous forme d'arbre la structure du système
- attribuer un nœud à des zones et de description
- lister les acteurs pour chaque nœud
- gérer les acteurs
- gérer les profiles

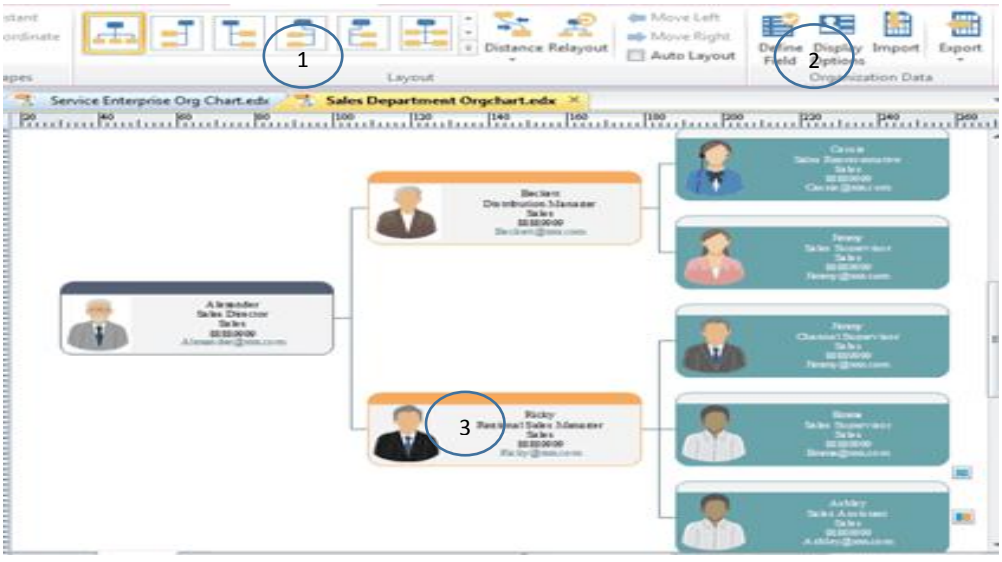


Figure 5.22 : Formulaire de gestion de l'organisation

1 : Outil graphique
2 : propriétés de chaque nœud
3 : graphe

En utilisant le même principe que le processus, l'organigramme de l'organisation est structuré en XML et présenté sous forme de graphe sur l'écran.

V.3.5 Module cartographie

Ce module permet de relier la structure de l'organisation avec la liste des processus actifs du système (Figure 5.23). Il permet de :

- élaborer la relation entre une entité de l'organisation avec leurs processus métier
- identifier les entités impactées par un processus
- tracer la production d'un article

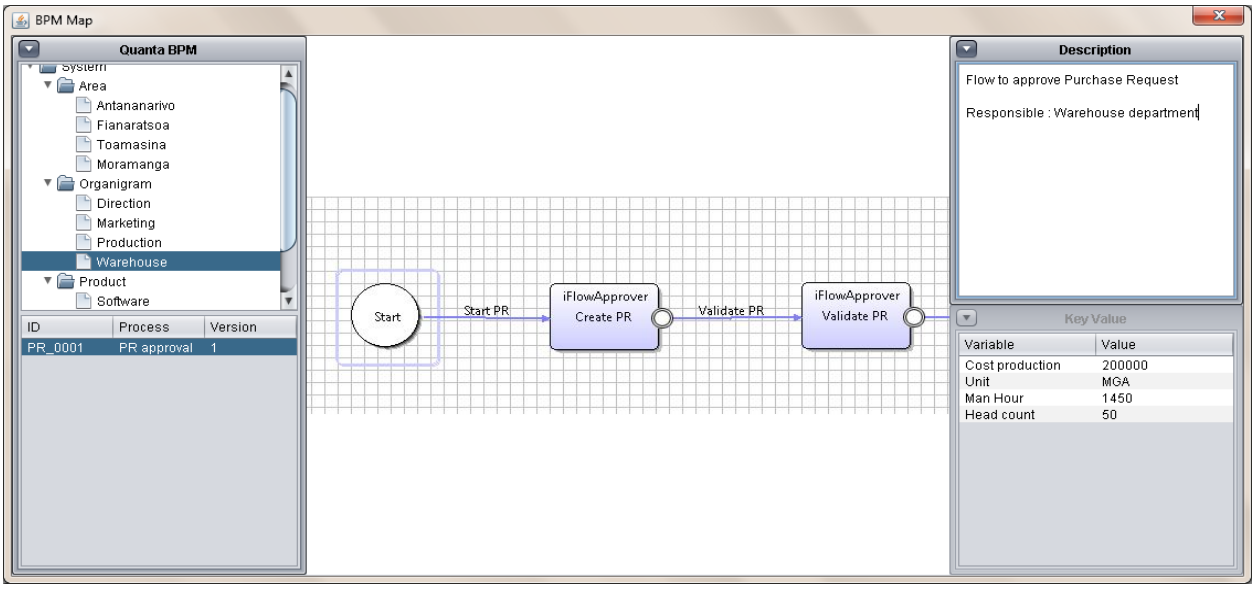


Figure 5.23 : Formulaire de cartographie de processus

V.3.6 Module de gestion de dimension d'analyse

Les données dans chaque module sont structurées selon le besoin du traitement, avec une telle structure, la fouille de données devient complexe sans définir au préalable de règle d'extraction.

Une dimension est une structure d'information qui stocke les données clés pour rechercher des données dans l'ensemble des données opérationnelles (Paragraphe I.2.4.2)

Ce module permet de saisir l'arborescence de la dimension (parent/fils et *mapping* des données) et les requêtes qui doivent être utilisées pour extraire les données pour chaque version des traitements.

Le formulaire de saisie de dimension (Figure 5.24) permet la visualisation hiérarchique de chaque élément de base de l'axe d'analyse du système.

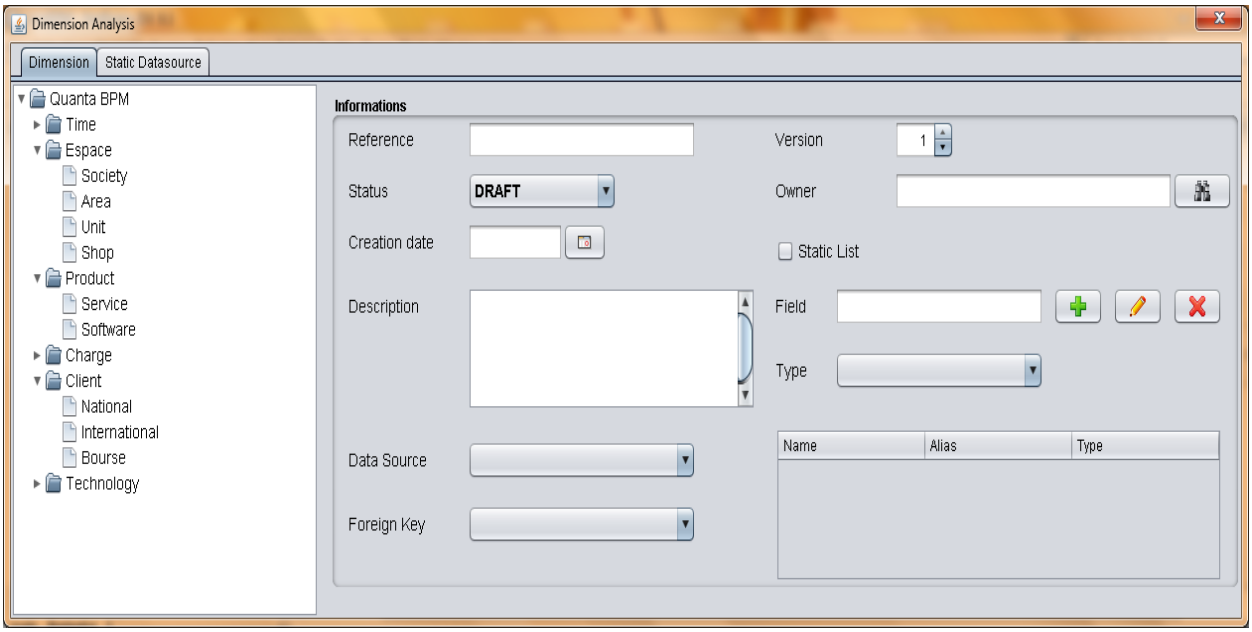


Figure 5.24 : Formulaire de gestion de dimension

V.3.7 Module de gestion de flux

Ce module est le plus gros générateur de données dans le système. Il gère l'automatisation et la distribution des tâches selon le modèle de traitement définit dans les processus.

Dans un système Workflow, la distribution des traitements est automatisée mais pour gérer les exceptions et pour suivre l'évolution de certaine tâche, le module offre les fonctions suivantes :

- recherche des instances
- changement de cours de traitement
- réassignation

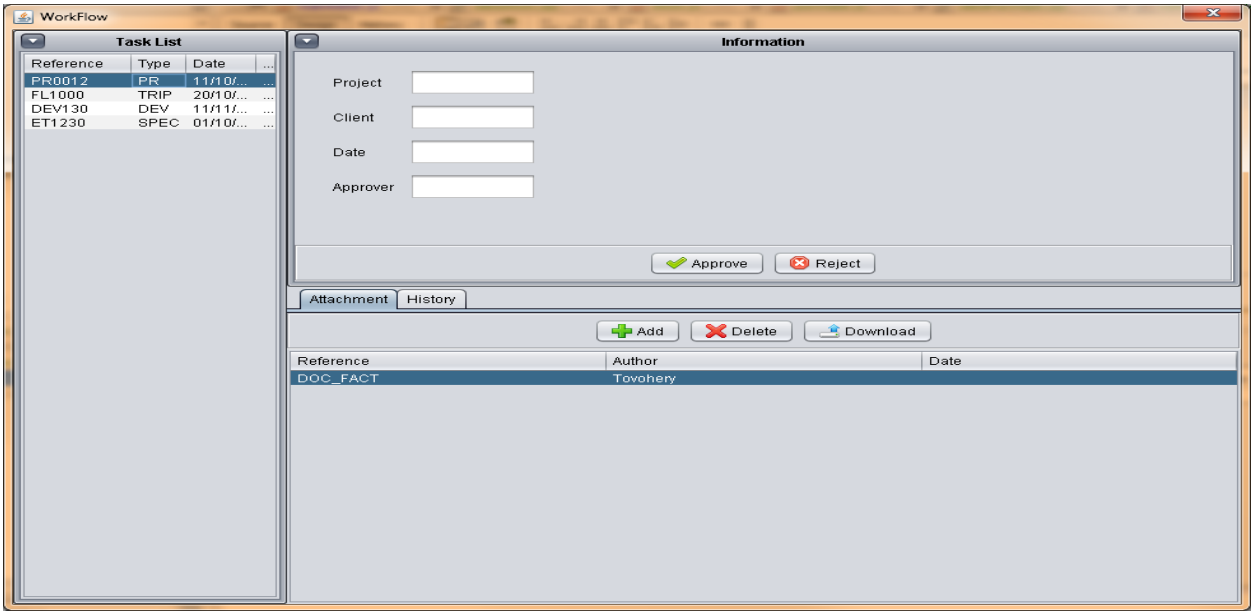


Figure 5.25 : Formulaire de gestion des instances de processus

V.5 Conclusion

La structure des entreprises et des organisations actuelles place encore l'homme au centre de tous les traitements et surtout dans la prise de décision. Par souci d'indépendance et de réduction de risque, on suggère d'ajouter une autre couche dans la structure de l'outil de gestion de système d'information afin de mémoriser les connaissances du système et valoriser les expériences vécues.

Les modélisations qu'on a réalisées sur l'ensemble de composants qui constituent le SI convergent sur l'utilisation de processus comme base de traitement ce qui nous amène à appeler l'outil Quanta BPM. Le processus est quantifié en activité, le métier en service, la connaissance en Cas, et la communication en événement c'est-à-dire que chaque élément possède une entité minima quantum indivisible.

Avec le support des ressources informatiques et de télécommunications, le travail à la chaine et l'ensemble de concept d'organisation de travail converge vers le management par processus ; c'est-à-dire le pattern sur les composants TIC a prouvé l'efficacité des patterns BPM.

La réalisation de Q-BPM ne s'arrête pas à la réalisation des modules qui constituent son noyau, il continue lors de la phase d'installation ainsi que l'amélioration du système. Dans le dernier chapitre de cette étude, on va spécifier les procédures de mise place de cet outil dans une organisation.

CHAPITRE 6 : Méthodologie de mise en place de Q-BPM

VI.1 Introduction

Mettre en place un système dans un autre système existant est une question d'adaptation, d'investissement et de technique. L'adaptation concerne plus les acteurs tandis que l'énergie nécessaire pour mettre en place les changements concerne la volonté du noyau du système.

Pour la partie technique, l'outil (Q-BPM) à mettre en place doit donner des assurances aux acteurs et au système tout entier que :

- il ne sera pas une charge de plus pour les gens du métier
- il sera d'un retour d'investissement (rentabilité et efficacité)
- il sera maniable et évolutif
- il sera sécurisé

VI.2 Méthodologie

La mise en place d'un SI en général ne se soucie pas de l'outil que l'on va mettre en place, il se concentre sur l'état actuel du système et les objectifs fixés par la direction pour proposer des solutions ainsi que l'outil à mettre en place. Pour le Q-BPM, c'est l'outil qui guide le système à s'introduire dans le concept d'amélioration continue appuyé par l'expérience.

VI.2.1 Démarche standard

Selon la boucle définie par la figure 6.1, la démarche est constituée de trois périodes :

- la conception,
- la réalisation
- la maintenance

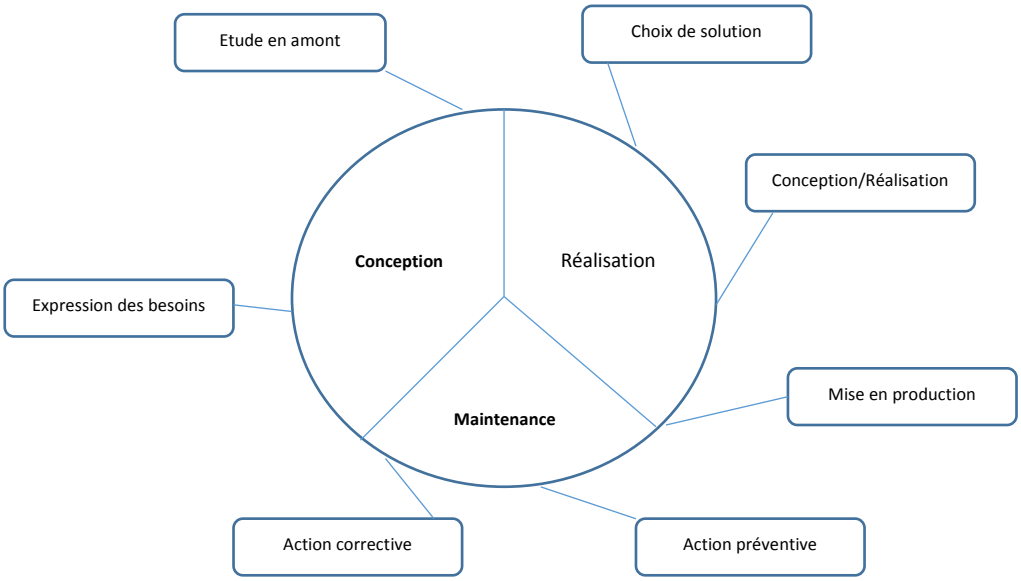


Figure 6.1 : Démarche de mise en place d'un SI

La période de conception se décompose en deux étapes : l'étude préalable et l'étude détaillée. Quant à la période de réalisation, elle comporte trois étapes : l'étude technique, la production du logiciel et la mise en service. Enfin la période de maintenance dans laquelle on distingue une maintenance corrective et évolutive.

VI.2.1.1 Etude préalable

Il se décompose en deux phases principales :

- élaboration de cahier de charges.
- proposition de réponse

a. Elaboration de cahier de charge

Cette phase assure un cadrage du projet, en précisant ce qui doit être faite.
L'objet du cahier de charges est de définir les points suivants :

- présentation générale du projet (contexte, objet, périmètre, démarche, planning organisation).
- prestations attendues (fournitures de progiciels et/ou de matériels, maîtrise d'œuvre du projet, définition des livrables).
- spécifications fonctionnelles (définition macroscopique du futur système d'information, présentation des acteurs internes et externes, définition des grands concepts, des règles de gestion, des fonctions attendues).
- spécifications techniques (volumétrie, contraintes d'architectures techniques, droits d'accès et confidentialité).

b. Proposition de réponse

Une réponse au cahier de charge est un dossier qui comporte :

Une présentation générale du futur système d'information en indiquant les principales novations par rapport au système actuel.

- conséquences sur l'organisation de l'entreprise.
- moyens humains et matériels à mettre en œuvre.
- scénarios de mise en œuvre.
- chiffres clés (bilan, coûts, avantages).

VI.2.1.2 Etude détaillée

Elle est basée sur la décomposition du système pour faire sortir la liste des acteurs, la liste fonctionnelle, les ressources et les séquences de traitement.

En pratique on utilise la méthode QQOQCP pour extraire ces informations. Il s''agit de poser des questions de façon systématique afin d'avoir le maximum d'informations :

- Quoi?

On pose cette question pour une meilleure description de l''activité ou de la tâche ou du problème.

- Qui?

Cette question permet une meilleure description des exécutants, acteurs ou personnes concernées.

- Où?

Cette question concerne la description des lieux ou des zones concernées par l'action.

- Quand?

On pose cette question pour bien définir les temps, la fréquence ou la répétition de la tâche.

- Comment?

Elle permet d'obtenir la manière ou la méthode pour réaliser l'action.

- Pourquoi?

Elle est utile pour identifier la cause d'un fait ou une valeur quelconque.

VI.2.1.3 Choix de solution

En comparant l'objectif fixé et les détails de l'existant, on choisit la solution à mettre en place :

- achat de logiciel et adaptation
- conception d'un nouveau logiciel

a. Adaptation d'un logiciel existant

Ce choix est guidé par la rapidité de la mise en place et la sécurité de la maintenance : il y a toujours de contrat d'assistance ou maintenance après l'achat pour pallier les gaps :

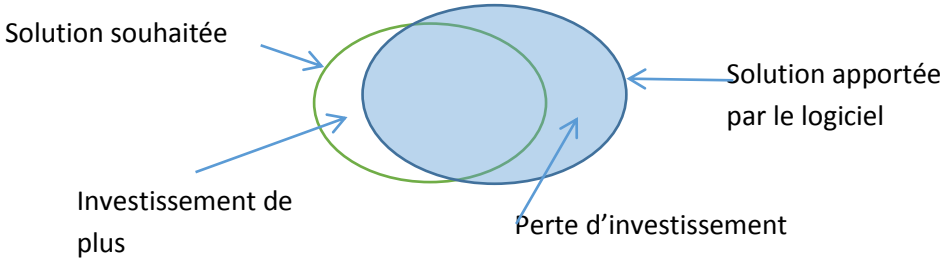


Figure 6.2 : Problème d'adaptation de logiciel existant

b. Conception de nouvel outil

Pour avoir plus d'indépendance et pour assurer les détails de la solution, on choisit de concevoir un tout nouveau logiciel pour le SI.

Ce choix présente des inconvénients :

- lent et perturbant
- les interférences entre le concepteur et les acteurs peuvent nuire la qualité de l'outil
- problème de maintenance après le départ des consultants

VI.2.1.4 Conception et réalisation

Cette étape consiste à réaliser concrètement dans un environnement prédéfini, l'ensemble des spécifications. A l'issue de cette étape, l'ensemble du système devra être conforme aux exigences fonctionnelles et techniques.

Techniquement, la conception aboutira à la documentation technique de l'outil et au maquettage (en cas de nouveau logiciel) :

- document de spécification des données
- document de spécification de traitement

VI.2.1.5 Déploiement

L'objectif principal de cette étape est de rendre opérationnel le système d'information. A l'issue de cette étape, les parties prenantes dans le projet peuvent se prononcer sur la recette définitive, et dissoudre la plupart des structures spécifiques ayant permis à la concrétisation du projet.

La mise en service peut se découper en quatre phases principales :

- la planification de la mise en service.
- connaissance des caractéristiques du matériel.
- spécifications détaillées des procédures transitoires.
- documentation utilisateur.

VI.2.1.6 Maintenance

Le système installé peut avoir deux types de problème :

- une ou plusieurs fonctionnalités qui ne donnent pas de résultat attendue : bug
- une ou plusieurs fonctionnalités sont manquantes dû aux analyses ou conception : amélioration

Pour pouvoir corriger le problème, la démarche remis en cause la phase de conception et la phase de déploiement.

VI.2.2 Démarche avec Q-BPM

Dans le cas pratique, la solution standard ne donne pas vraiment une garantie que le SI fonctionnera correctement après le déploiement :

- pas de phase d'apprentissage
- pas d'outil de référence pendant l'analyse
- on recommence toujours à zéro pour une nouvelle organisation
- l'état initial est dans un document donc pas de référence de départ pour mesurer l'efficacité du système

Pour améliorer cette démarche, la méthodologie Q-BPM de mise en place du système d'information possède deux volets :

- méthode de conduite du changement et de projet
- méthode technique d'implémentation du système

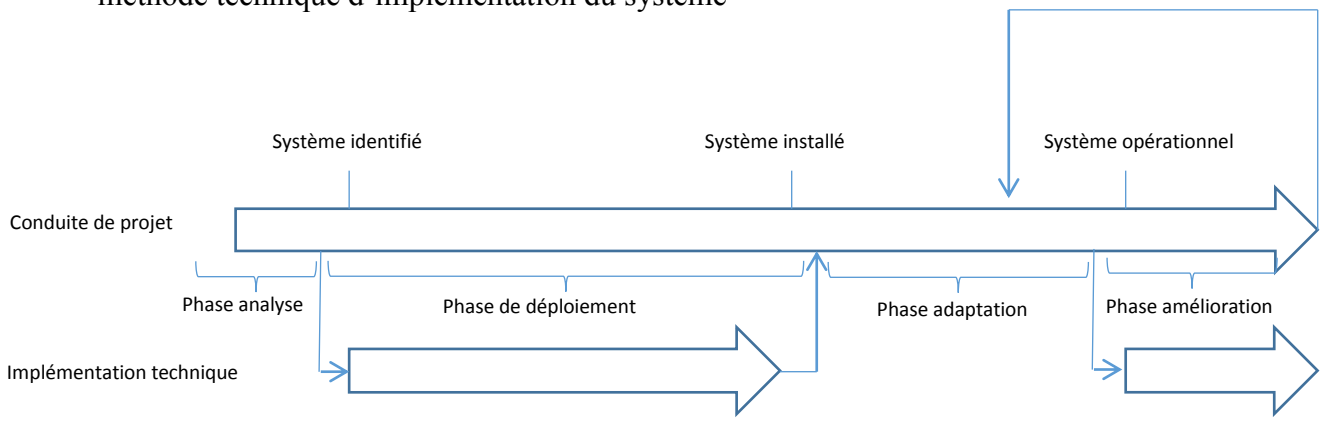


Figure 6.3 : Etat du système pendant la mise en place de l'outil

A partir du planning défini par la figure ci-dessous, la réalisation de la mise en place du système comporte quatre phases :

- analyse
- conception, réalisation et déploiement
- adaptation et suivi
- amélioration continue (maintenance)

VI.3 Conduite de changement

Un projet de changement est un processus psychosociologique complexe. Plus il y a de collaborateurs, plus le projet est complexe. Les intervenants rencontrent tous des problèmes. Le problème le plus fréquent est la résistance au changement. Cette résistance est causée souvent par le bouleversement des habitudes que cause le changement.
Pour mener à bien cette mission, il faut appliquer plusieurs techniques de management :

- communiquer les objectifs : vision claire, complète et positive du changement désiré.
- impliquer l'ensemble des acteurs
- impliquer d'avantage le noyau du système dans la procédure de changement
- encourager les critiques constructives : gestion des incidents et des requêtes

Cette démarche a pour objectif de transformer les ressources humaines à des acteurs susceptibles d'utiliser et de faire marcher le nouveau système c'est-à-dire impliqué ces ressources dans le changement tout en expliquant leur intérêt dans le nouveau système.

VI.3.1 Phase d'analyse

L'objectif de cette phase dans la gestion des changements est de définir les savoir-faire du système et ses relations avec ses acteurs (Figure 6.4). L'analyse du système devrait inclure donc les points suivants :

- enquête sur l'organisation et les différents modes opératoires
- enquête sur l'ensemble des acteurs
- enquête sur le mode de travail
- enquête sur l'efficacité des communications existantes

Ces données seront incluses dans la liste des indicateurs de performances initiales du système pendant la mise en place de tableau de bord de départ.

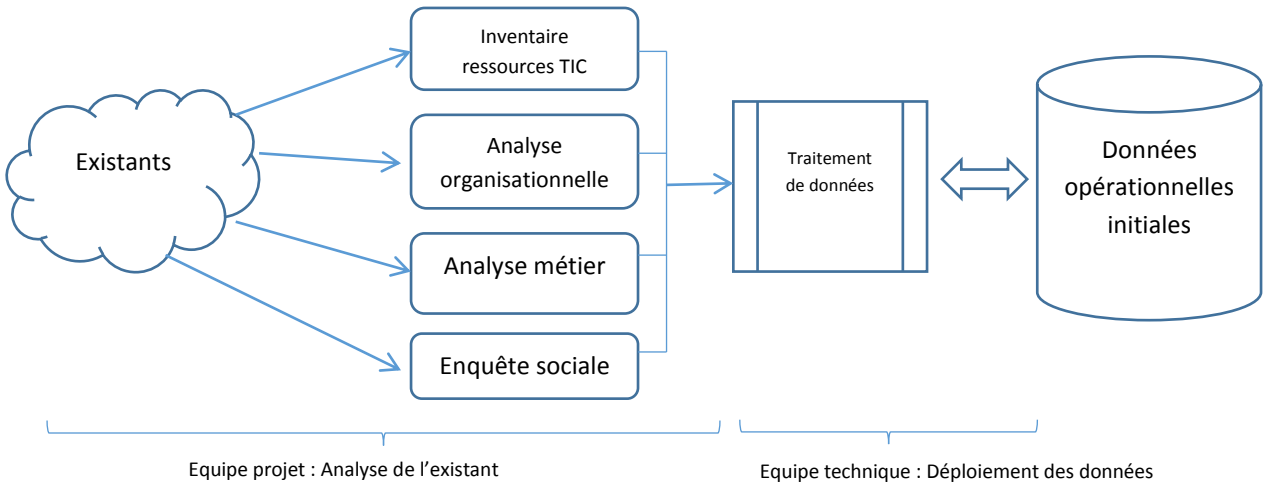


Figure 6.4 : Elaboration des données opérationnelles initiales

Le tableau de bord initial est généré à partir des données collectées manuellement pendant la phase d'analyse. Il est l'outil principal pour le gestionnaire du projet et le noyau du système (équipe projet) pour prendre des décisions et mener à bien le changement.
Dans cette phase, le système est installé dans un environnement restreint pour réduire l'impact de son présence et pour que l'équipe technique puisse travailler en dehors du quotidien de l'organisation.

Les infrastructures nécessaires sont :

- serveur de base de données
- serveur d'application pour BPM, BI, JBI et services de saisie
- serveur de fichiers

Les enquêtes et la collecte des données sont basées en général sur des formulaires prédéfinis et de document de spécification technique. Pour les questionnaires relatifs aux ressources humaines, on peut utiliser les trois types de formulaire suivant :

- individu à l'acteur
- activité en ressource
- communication en ressource

VI.3.1.1 Formulaire « de l'individu à l'acteur »

Les critères et dispositifs à étudier pour rendre les collaborateurs pleinement acteurs de leur organisation :

- sentiment d'appartenance à l'organisation
- culture d'entreprise dans l'organisation
- stéréotypes professionnels.
- autorité (style de management) et le comportement des membres de l'organisation.
- les éléments sur lequel se fonde l'autorité d'un chef d'entreprise (exemple).
- motivation qui pousse à choisir l'organisation : valeurs, la culture commune, mode de travail
- les comportements des individus au sein des groupes.
- l'individu et sa personnalité, ses émotions,
- communication interpersonnelle dans l'organisation,
- les groupes : groupe de référence, groupe d'appartenance…
- confiance : vie privée
- répartition de valeur de l'organisation
- reconnaissance

VI.3.1.2 Formulaire « conversion de l'activité humaine en ressource »

Les conditions de travail qui favorisent l'implication et l'efficacité dans leurs activités doivent être analysées :

- la mixité dans le lieu de travail.
- les indicateurs qui mesurent l'activité humaine.
- l'activité humaine : une ressource ou une charge.
- rémunération et les autres couts liés à l'activité humaine.
- actions pour améliorer les conditions de travail.
- actions pour améliorer les performances dans l'entreprise.
- méthode pour garder les meilleurs
- méthode de recrutement
- organisation de temps de travail
- part des salaires dans les coûts
- motivation et l'efficacité du salarié.

VI.3.1.3 Formulaire « conversion de la communication en ressource »

Les outils de communication mis en place par l'organisation pour partager de l'information doit être l'objet d'un audit :

- pratiques mises en œuvre dans le domaine du partage de l'information.
- partage de l'information conduit à la collaboration.
- enjeux de la maitrise de l'information collective pour l'organisation.
- les rôles, les droits et les responsabilités des acteurs d'un réseau.
- les apports de l'intelligence collective pour l'organisation.
- intérêt des réseaux sociaux pour les organisations.
- comment l'organisation permet-elle aux salariés de partager l'information ?
- l'organisation utilise-t-elle les réseaux sociaux ?
- les forums de discussions professionnels sont-ils nécessaires dans l'organisation ?
- les nouvelles technologies sont-elles favorisé les relations avec les clients

VI.3.2 Phase de déploiement

A la fin de la phase d'analyse, on a déjà une version de Q-BPM installé et déployé qui peut assurer les tâches suivantes :

- saisir des données via les différentes enquêtes de la phase d'analyse pour établir le tableau de bord numérique.
- saisir les processus initiaux
- implémenter les services de base
- tracer les organisations
- élaborer les Cas initiaux
- préparer les solutions de mise en marche

Pendant la phase de déploiement, on installe la version finale de l'outil sur le réseau de l'organisation. L'équipe projet travaille avec l'équipe technique pour intégrer l'existant dans le nouveau système et de tester l'environnement avant la mise en place de la solution initiale.

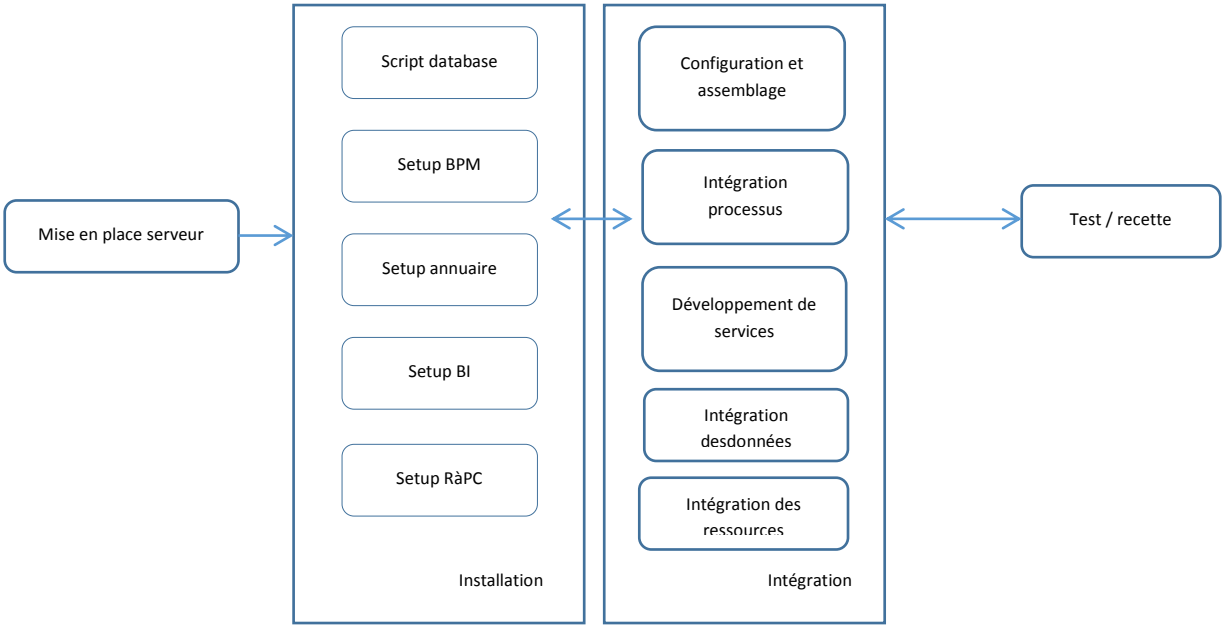


Figure 6.5 : Déploiement du Q-BPM

Le déploiement comporte quatre étapes (Figure 6.5) :

- préparation : installation et configuration des serveurs, configuration de réseau et préparation des listes des ressources à intégrer
- installation : déploiement de chaque module
- intégration : données, services, processus
- test : les acteurs puissent accéder sur le système pour faire de test. Après le test on doit réintégrer les données (ou restauré) pour avoir un système propre.

VI.3.3 Phase d'adaptation

C'est la phase clé de la réussite de la mise en place du nouveau SI :

- l'équipe projet donne de formation et communique avec les acteurs sur le nouveau système
- l'ensemble de l'équipe technique travaille sur l'intégration des nouvelles solutions (processus, organisation, procédure, service) pour adapter le Cas initiale à la réalité.

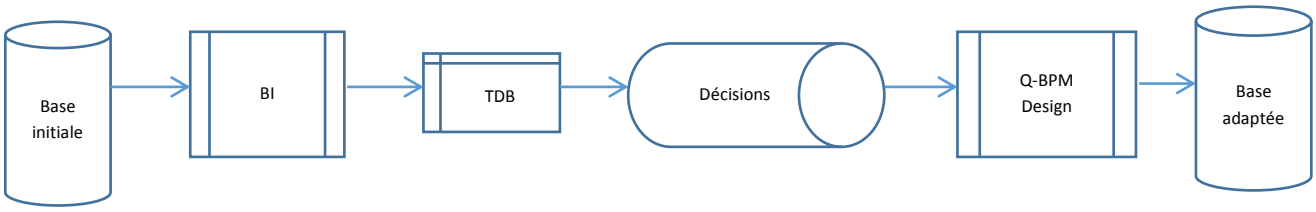


Figure 6.6 : Adaptation de la situation initiale

Selon l'objectif de départ de l'organisation, l'équipe projet définit des décisions en utilisant le Tableau de bord initial et puis l'équipe technique transforme les décisions prises en solutions tout en considérant les ressources disponibles.
La figure 6.6 permet d'identifier les différentes étapes de conversions de la structure des données du système par le système BI et BPM.

VI.3.4 Phase d'amélioration

C'est la phase de la responsabilisation des acteurs du système. L'équipe projet assiste l'organisation pour réaliser quelques cycles d'amélioration du système avant de clôturer le projet de mise en place.

Q-BPM et le système d'information, en général ne sont pas seulement une gestion de l'information mais surtout une culture : « pour évoluer, il faut utiliser les informations et les transformées en connaissance car la connaissance donne de la maitrise ».

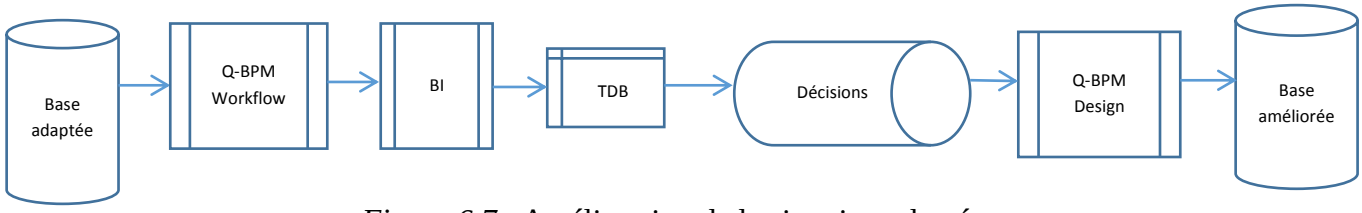


Figure 6.7 : Amélioration de la situation adaptée

On incite donc le système à critiquer les situations existantes pour relancer des nouveaux cycles d'amélioration et de faire vivre les connaissances collectives de l'organisation.
L'amélioration de la base adaptée passe par le système Workflow, BI, et BPM pour obtenir la base améliorée (Figure 6.7).

VI.4 Implémentation technique

Cette partie concerne uniquement l'utilisation, l'adaptation et l'amélioration de cœur du Q-BPM pour le besoin de l'organisation cible.

Pour la phase d'analyse, l'équipe technique n'intervienne que dans l'évaluation des ressources techniques et TIC disponible sur le système existant.

Pour la phase de déploiement, d'adaptation et d'amélioration, le système Q-BPM peut être manipulé avec les quatre options suivantes :

- installation
- *componentization*
- factorisation
- évolution

VI.4.1 Procédure d'installation

Selon la Figure 6.5, la procédure d'installation est purement technique. Dans cette partie, l'organisation peut choisir les éléments du noyau du Q-BPM à intégrer dans son système.

En fonction de sa taille, une organisation n'a pas nécessairement besoin de l'ensemble de l'outil car plus il y a de modules, plus on a besoin d'infrastructure.

Tableau 6.1 : Liste de version de Q-BPM

Version	**Modules**	**Matériels de base minimum**
BPM	- BPM engine - BPM modeler - Annuaire de Service	- 1 Serveur de base de données - 1 Serveur d'application - 1 Serveur JBI
BPM Intelligent	- BPM engine - BPM modeler - Annuaire de Service - BI *engine* - BI *dashboard*	- 1 Serveur de base de données - 2 Serveurs d'application - 1 Serveur JBI
Q-BPM	Tous	- 2 Serveurs de base de données - 3 Serveurs d'application - 1 Serveur JBI
Q-BPM +	Tous, plus les modules non standards approuvés	Non définis

VI.4.2 Procédure de Componentization

Q-BPM offre des interfaces pour intégrer des nouveaux composants dans le système selon le besoin du SI :

- composant d'interface utilisateur
- composant ressource
- composant service

Les composants « interface utilisateur » et ressource ne sont pas utilisables dans le système qu'à travers des services.

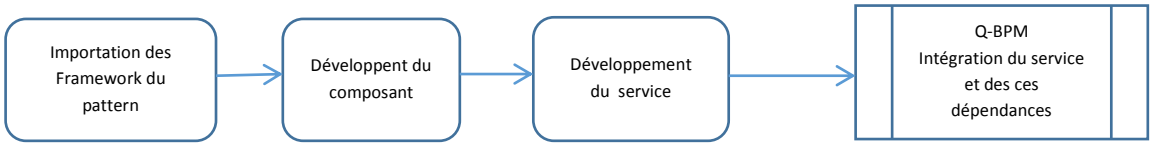


Figure 6.8 : Procédure de componentization

Les étapes définis dans la figure 6.8 permettent la transformation du métier en composant manipulable par les utilisateurs.

VI.4.2.1 Interface utilisateur

Cette catégorie d'interface permet de personnaliser les présentations des données afin que les acteurs puissent interagir avec l'outil.

Selon le pattern sur l'IHM (Paragraphe III.3.1), la modification d'un composant existant est seulement possible en agissant sur la liste des propriétés définies dans les instances de UIStyle.

Si on veut créer des nouveaux UILayout, UIPresentation, UIContainer ou dataModel, on doit créer d'autre composant et implémenter les interfaces du pattern.

VI.4.2.2 Ressource

Les matérielles et les technologies varient d'un système à l'autre et évoluent dans le temps ; ceci implique que les déclarations dans la classe statique du noyau du Q-BPM soient dépassées par l'événement.
Le système permet la réécriture de ces classes selon leur responsabilité et d'écraser la version ancienne. Cette opération nécessite un redémarrage complet du système tout entier :

- classe serveur : FTP, http, LDAP, Mail, *Database* etc...
- classe conversion : Système de fichier, données natives, office

VI.4.2.2 Service

Dans le cas pratique, le service est l'organe le plus volatil du système. À chaque fois qu'on change un élément dans le système : processus ou composant, la mise à jour de service est toujours recommandée.
Le développement et la conception des services et ses dépendances sont indépendants du cœur de l'outil Q-BPM mais l'intégration nécessite les spécifications suivantes :

Tableau 6.2 : Contraintes d'intégration

Elément	**Contrainte**
Langage de programmation	- version - librairie de base
Pattern	- même interface - nomenclature - version
Q-BPM	- vérification des instances en cours qui utilisent encore les dépendances - référence des Cas - référence de service de contrôle
Equipe	- documentation - serveur Agile et serveur des codes GIT

VI.4.3 Procédure de factorisation

L'erreur fatale qui peut détruire un SI est de mettre en parallèle deux ou plusieurs systèmes distincts qui ne se communiquent pas entre eux.

Alors que dans une organisation, il se peut que plusieurs systèmes soient obligatoires pour faire fonctionner correctement la production. Pour éviter ce problème de dispersion des données, l'équipe technique devrait trouver des moyens pour faire communiquer tous les systèmes qui sont susceptibles de traiter des informations avec le Q-BPM.

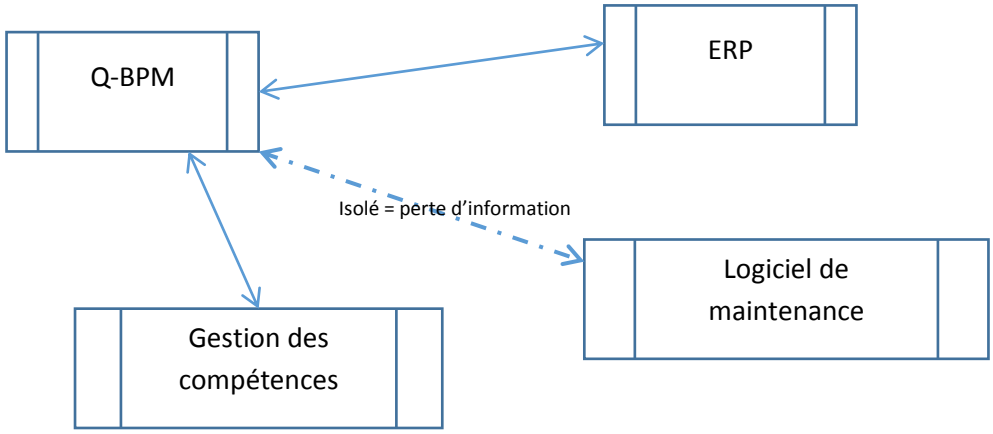


Figure 6.9 : Exemple de système isolé

Dans cet exemple (Figure 6.9), les données et les processus définis dans le logiciel de maintenance ne seront pas accessibles par Q-BPM. Les problèmes (coût, lenteur, etc...) ou les avantages (retour d'investissement, nombre des matériels réparés par unité, etc...) apportés par cet outil seront donc totalement inconnus par le décideur qui regarde le tableau de bord via Q-BPM.

Il y a trois moyens techniques pour créer des relations entre deux systèmes :

- utilisation d'interface commune
- partage des événements
- partage des bases de données

VI.4.3.1 Interface commune

Cette technique consiste à utiliser un élément tiers qui est utilisé par les deux systèmes :

- web service
- Framework interface
- Système REST :

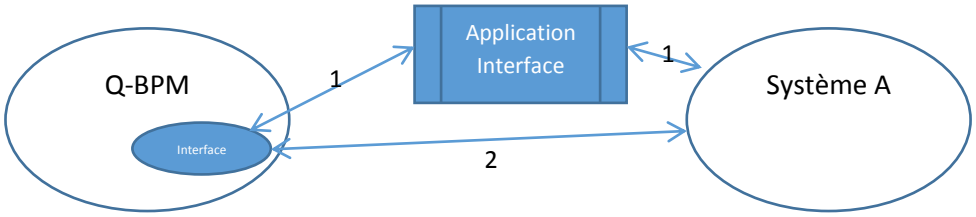


Figure 6.10 : Mode de communication par Interface

Relation 1 : Le « système A » permet une ouverture avec la mise en place d'une application Interface qui peut se connecter avec d'autre système.

Relation 2 : Le « système A » utilise des Framework et des services susceptibles d'être instanciés à l'extérieur.

VI.4.3.2 Récupération des évènements

Pour une application client-serveur, il est possible que le serveur envoie de signal broadcast vers ses clients à chaque fois qu'il y a de traitement réalisé. La technique de récupération des évènements consiste à écouter le système cible et d'interpréter les informations qu'on a reçues selon la documentation fournie par le constructeur.

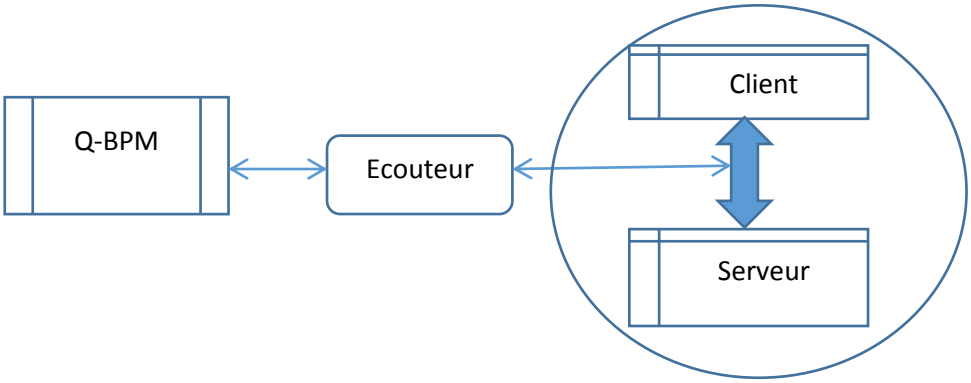


Figure 6.11 : Mise en écoute d'un système

L'application qui fait l'écoute utilise les moyens de communication du système pour extraire les données (Figure 6.11) et les modéliser pour être transférées dans le processus de gestion de ces données dans Q-BPM.

VI.4.3.3 Accès direct aux données

Cette technique n'est pas recommandée mais si le système cible ne permet pas un accès indirect, on peut puiser les informations directement de sa base de données.

Cette technique est possible si :

- on comprend le modèle de données
- on peut accéder à la base sans risque de perturbation du système (charge d'accès, tables verrouillée)

VI.4.4 Procédure de l'évolution

La capacité de transformer les connaissances tacites en concepts explicites permet à l'entreprise de créer une intelligence collective qui améliore durablement sa compétitivité.

Avec Q-BPM, on peut partager ces expériences par la création des modules métiers qui correspond aux expériences des autres organisations. Cette option peut accélérer la croissance des autres organisations qui l'utilisent en sachant qu'elles comprennent son origine.

La procédure d'évolution consiste à extraire d'un système une partie de ses savoir-faire pour créer un module.
Q-PBM permet l'import-export des processus et ses dépendances vers d'autre système Q-BPM.
Un module est un objet qui contient :

- l'entête de l'organisation d'origine, description problème/solution
- modèle du processus
- liste des profils
- script de données pour ses services
- setup des services

VI.4.4.1 Extraction de module

Le processus et ses composants sont des propriétés de l'organisation, et il a tout le droit de protéger ou partager ces informations.

L'administrateur « root » du système seul peut lancer la procédure d'extraction. Elle consiste à empaqueter les données dans un fichier zippé avec la même technique du langage Java sur les fichiers de type jar, war, ear :

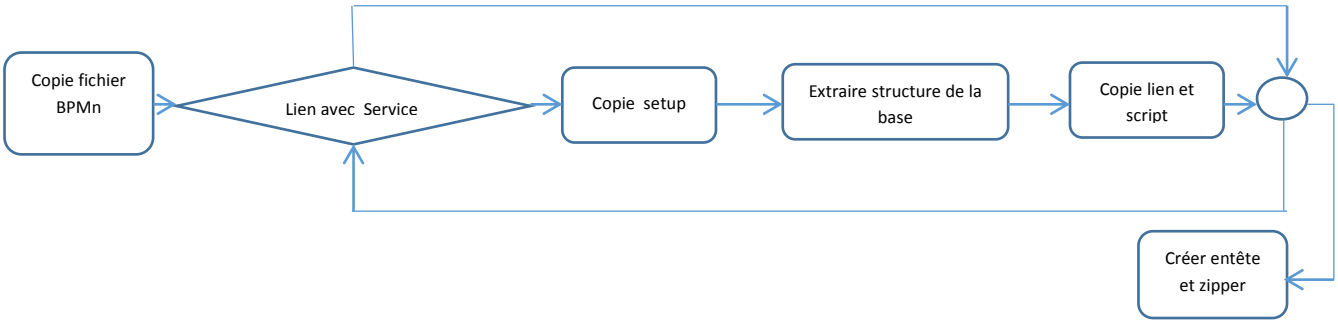


Figure 6.12 : Procédure d'extraction de processus

VI.4.4.2 Importation de module

C'est la procédure inverse de l'extraction. Il est aussi dangereux que la procédure d'extraction car on ne connait pas d'avance les codes qui s'exécutent dans les services intégrés.

Avant toute opération d'importation, le système vérifie si le module est compatible (base de données, version de Framework) avec l'environnement et qu'il est possible de créer les profils sur l'organigramme en cours.

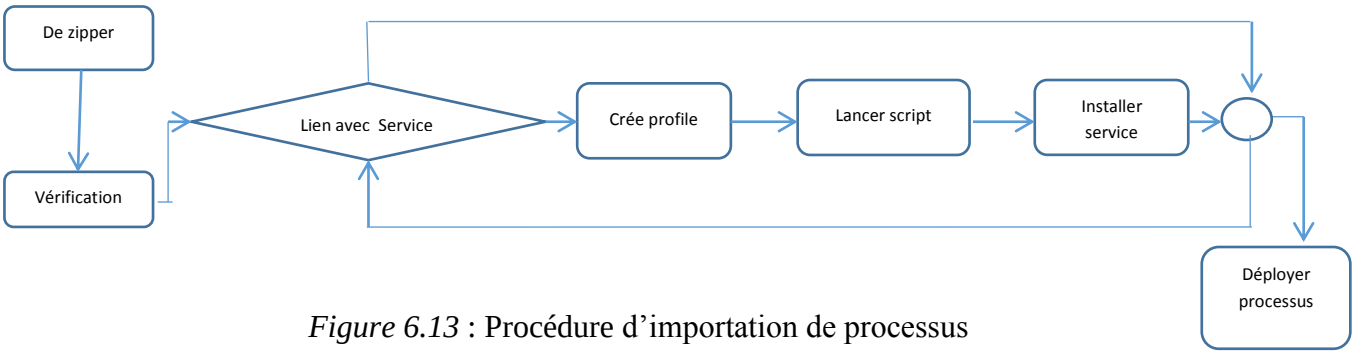


Figure 6.13 : Procédure d'importation de processus

Cette procédure nécessite une compétence de haut niveau sur la structure du système existant pour éviter des conflits et de blocage de traitement.

VI.5 Conclusion

Pendant la phase de conception, de déploiement et de traitement au sein du système d'information, l'acteur humain présente toujours le facteur clé dans la définition de ce qu'on va faire et comment on va le faire. Cette situation a des impacts positifs et négatifs sur le système qui nécessite d'être mis en cause dans le principe de fonctionnement de Q-BPM.

Notre solution a pour vocation d'accompagner l'homme tout au long du processus par des principes et de procédure afin qu'il puisse exécuter ses tâches en toute confiance.

L'outil Q-BPM offre un moteur workflow pour assister les acteurs sur la réalisation de son métier ; il donne un tableau de bord pour pouvoir prendre de bonne décision ; il enregistre les Cas antérieurs pour avoir des idées pendant la conception ; il assiste l'équipe projet sur l'élaboration des Cas de la situation initiale et il aide les systèmes différents à échanger des expériences.

Pour conclure, le concept Q-BPM est un instrument conçu à l'aide des expériences de l'homme dans un système d'information pour en construire un autre plus rentable.

Conclusion générale

Il y a un moment où il faut se retourner et de revoir le chemin qu'on a fait pour pouvoir construire le futur. Le *design pattern* représente une partie de notre expérience et de notre solution ; nous devrions l'exploiter pour construire des outils et des méthodologies pour avancer vers la maitrise des systèmes qui nous entourent.

La technologie sur la modélisation de système informatique est un domaine de l'abstrait qui se base sur des définitions et des représentations. Il est parfois difficile de prouver une théorie ou une spécification sans l'instanciation. Mais avec les patterns, on n'a pas besoin de démontrer une solution car on suppose déjà qu'ils sont prouvés.

La modélisation d'un système d'information avec des *design patterns* est donc une conception logique et qui devrait constituer la base de la nouvelle génération de génie logicielle.

La transformation de l'information par le processus métier et l'interprétation des données par le processus décisionnel constitue le moteur du système d'information ; avec ces deux blocs principaux, on peut en déduire que l'outil de gestion de système d'information est composé de :

- gestionnaire de processus,
- gestionnaire de ressources,
- gestionnaire des services,
- générateur de rapport,
- organe de communication des indicateurs

Ces outils de base tournent dans une architecture modulaire et distribuée qui est commandable par les acteurs du système à partir d'un seul noyau fixe que l'on a appelé ***Q-BPM***.

On a proposé cette étude dans le cadre de formalisation des modèles de conception et d'apporter notre vision du système d'information. Elle nous a permis d'identifier les différents composants techniques qui constituent le SI et leurs interactions. En plus on est arrivé à identifier les différentes classes de l'information dans le SI au cours de son traitement :

- information physique
- information codée
- information de masse
- information agrégée
- information clé
- incident
- risque
- décision
- objective

Ces types de réorganisation de l'information peuvent prendre plusieurs états et plusieurs interprétations selon le contexte et le niveau de traitement. Avec cette complexité de la notion de l'information, le SI reste encore un système complexe et ouvert, et on a besoin de continuer la recherche pour augmenter notre maitrise du système et notre savoir-faire sur l'information.

ANNEXES

ANNEXE 1 : Framework Ressource

Ce Framework fait partie des piliers du système Quanta. Elle fournit des services élémentaires pour créer ou accéder à une ressource.

A1.1 Architecture

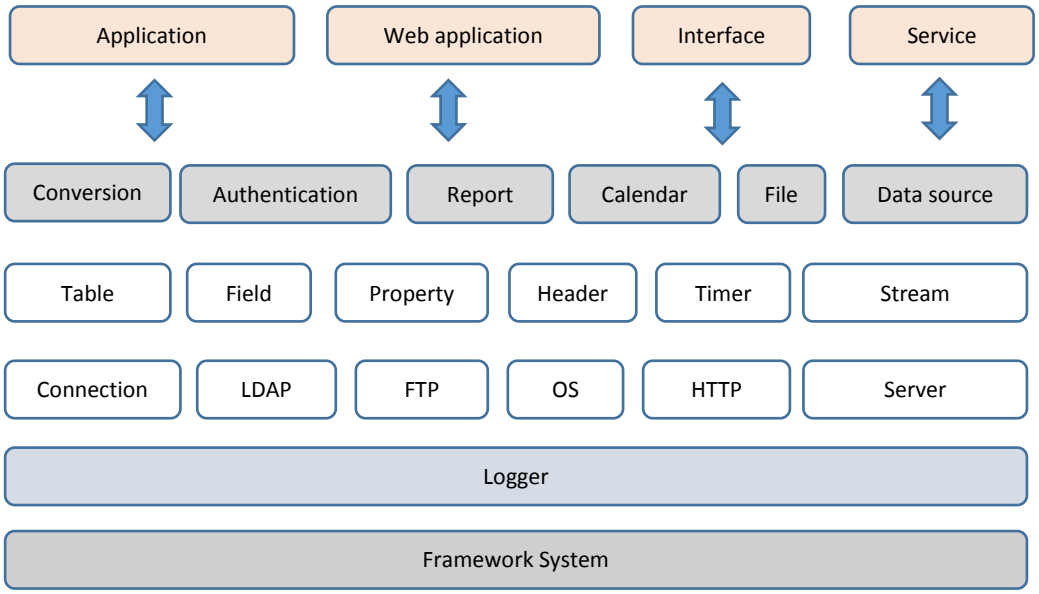


Figure 7.1 : Couches logicielles du Framework ressource

Le code relatif aux implémentations des classes dépend entièrement des Framework offerts par l'environnement de développement.
L'architecture est composée de quatre couches :

- couche log : pour tracer les traitements afin de faciliter la maintenance
- couche accès aux ressources
- couche modèle de données
- couche de traitement et d'interface qui est en relation direct avec d'autre système extérieur

A1.2 Package

Tableau 7.1 : Liste des packages

Package	Description
com.hitechs.commons.resources	Contient les modèles de données
com.hitechs.commons.resources.component	Contient des composants graphiques lourds
com.hitechs.commons.resources.component.calendar	Contient des composants calendrier
com.hitechs.commons.resources.component.grid	Contient des composants tableaux
com.hitechs.commons.resources.conversion	Contient des services de conversion de type
com.hitechs.commons.resources.conversion.loggers	Contient de classe d'enregistrement de Log
com.hitechs.commons.resources.database	Contient de modèle de données enregistrement
com.hitechs.commons.resources.database.gdb	Contient des composants de connexion aux bases de données
com.hitechs.commons.resources.file	Contient des modèles de données flux
com.hitechs.commons.resources.office	Contient des composants de gestion de fichier office
com.hitechs.commons.resources.office.open	Contient des composants de gestion de fichier open office
com.hitechs.commons.resources.property	Contient des modèles de données propriétés : liste des caractères d'un objet
com.hitechs.commons.resources.service	Contient des interfaces d'accès aux ressources
com.hitechs.commons.utils	Contient des composants de gestion de ressources système

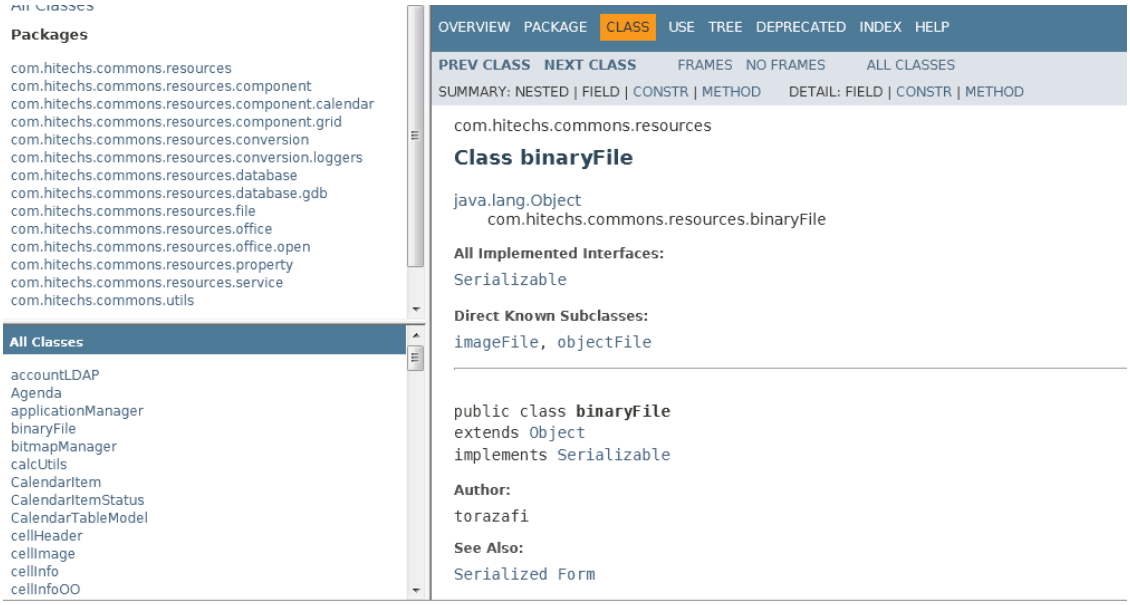


Figure 7.2 : Java doc du Framework

A1.2.1 Package resources

Tableau 7.2 : Liste des classes ressources

Class	Description
accountLDAP	Modèle de donnée LDAP
binaryFile	Modèle de donnée binaire
componentUni	Classe mère de composant
constanteUni	Contient des constantes
dataCron	Modèle de données Cron
dataForm	Modèle de données formulaire
dataMail	Modèle de donnée Mail
dataObject	Modèle de donnée liste
dataUni	Modèle de donnée enregistrement
field	Specify a field and his default value *
fieldUni	Modèle d'un champ
ftpServer	Modèle de données d'accès au serveur FTP
headerUni	Modèle de données d'entête
imageFile	Modèle de données Image
ldapServer	Modèle de données d'accès au serveur LDAP

objectFile	Modèle de flux d'objet
pageUni	Modèle de données pour gérer une page
pointUni	Modèle de données Point
proxyData	Modèle de données Proxy
rectangleUni	Modèle de données Rectangle
tableUni	Modèle de données Table
textFile	Modèle de données de flux texte
vectorUni	Modèle de données liste indexé

A1.2.2 Package component

Tableau 7.3 : Liste de classes composantes

Class	**Description**
imageViewer	Classe de traitement d'image
pageComponentUni	Classe de gestion de page
paneImage	Classe d'affichage d'image

A1.2.3 Package calendar

Tableau 7.4 : Liste des classes calendriers

Class	**Description**
Agenda	Modèle de données Agenda
CalendarItem	Modèle de données Item de calendrier
CalendarTableModel	Modèle de données liste des items
DayPanel	Classe d'affichage de Jour
DayView	Classe d'affichage de Jour avec format
MonthView	Classe d'affichage de mois
View	Classe d'affichage de calendrier
WeekView	Classe d'affichage de week

A1.2.4 Package grid

Tableau 7.5 : Liste des classes tableau

Class	Description
dateEditor	Modèle d'affichage de date
gridExport	Classe de transformation de données
gridUni	Modèle de donnée du tableau
metaColumn	Modèle de colonne
paneGrid	Classe d'affichage
paneGridPage	Classe d'affichage avec page
SpinnerEditor	Modèle d'affichage de chiffre

A1.2.5 Package conversion

Tableau 7.6 : Liste de classe de conversion

Class	Description
applicationManager	Gestionnaire du runtime Java
bitmapManager	Gestionnaire d'image
controlConnection	Gestionnaire de connexion oracle
dataComparator	Classe de comparaison
dateConversion	Gestionnaire de date
fileManager	Gestionnaire de fichier
ftpManager	Gestionnaire de connexion FTP
httpManager	Gestionnaire de connexion HTTP
jarManager	Classe de manipulation de fichier jar
ldapManager	Gestionnaire de connexion LDAP
mailManager	Gestionnaire de connexion au serveur Mail
nativeConversion	Classe de conversion des données natives
PDFManager	Classe de manipulation de fichier PDF
privilegedAction	Gestionnaire d'accès aux fichiers
requestParser	Parseur des objets Request

splashImage	Gestionnaire d'image de présentation
timerUni	Classe de donnée Timer
traductorDate	Classe de donnée pour traduire une date
traductorValue	Classe de donnée pour traduire objet
utilsData	Gestionnaire des données enregistrements
utilsForm	Gestionnaire de données formulaires

A1.2.6 Package loggers

Tableau 7.8 : Liste de classe d'enregistrement d'événement

Class	**Description**
ConnectionLogFactory	Constructeur d'entité Log
logConfigurationFactory	Classe de données de configuration
logJDBCConfigurationFactory	Classe de données de configuration pour une base de données
logWriter	Gestionnaire d'enregistrement de Log

A1.2.7 Package database

Tableau 7.9 : Liste de classe de gestion des données

Class	**Description**
comparatorUni	Classe de comparaison
connection	Classe de connexion
connectionManager	Gestionnaire de liste de connexion
connectionParameter	Classe de données de configuration d'accès
orderDataTable	Classe de données pour gérer un tri
queryBuilder	Classe de création de requête SQL
queryUtil	Classe de traitement de requête SQL
whereDataTable	Classe de données pour gérer le clause where d'une requête SQL

A1.2.8 Package gdb

Tableau 7.10 : Liste de classe de gestion des connections

Class	Description
connectionMySql	Implémentation d'une connexion Mysql
connectionOdbc	Implémentation d'une connexion ODBC
connectionOracle	Implémentation d'une connexion Oracle
connectionPostgres	Implémentation d'une connexion PostgreSql
connectionSqlServer	Implémentation d'une connexion Sql Server
metaMysql	Classe d'entête de connexion Msyql
metaOdbc	Classe d'entête de connexion ODBC
metaOracle	Classe d'entête de connexion Oracle
metaPostgres	Classe d'entête de connexion PostgreSql
metaSqlServer	Classe d'entête de connexion SQL server

A1.2.9 Package office

Tableau 7.11 : Liste de classe de traitement de fichier office

Class	Description
cellHeader	Classe de donnée de cellule
cellImage	Classe de donnée image
cellInfo	Classe de données sur le contenu de cellule
contentSheet	Classe de données d'une feuille
contentWord	Classe de donnée d'un document
pdfSheet	Convertisseur de feuille en PDF
pdfWord	Classe de conversion de document en PDF
rangeInfo	Classe de donnée pour un range de cellule
sheetManager	Gestionnaire de document Excel
writerHeadFooter	Gestionnaire d'entête d'un document
writerManager	Gestionnaire de document Word

A1.2.10 Package file

Tableau 7.12 : Liste de classe de données flux

Class	Description
fileInfo	Classe de données d'entête de fichier
fileResource	Classe de données fichier

A1.2.11 Package open

Tableau 7.13 : Liste de classe de traitement de fichier open office

Class	Description
calcUtils	Utilitaire sur le traitement de cellule
cellInfoOO	Classe de données cellule
connectManager	Gestionnaire de connexion avec l'application mère
constanteOo	Liste de constante
printListner	Ecouteur utilisé pour imprimer le fichier
sheets	Gestionnaire de document Sheet
tableInfo	Classe de données d'un tableau
tablesInfo	Classe de données pour une liste de tableaux
writer	Gestionnaire de document Writer

A1.2.12 Package property

Tableau 7.14 : Liste de classe de gestion des propriétés

Class	Description
groupUIProperty	Groupe de propriétés
UIProperty	Classe de donnée pour une propriété

A1.2.13 Package service

Tableau 7.15 : Classe de gestion de service

Class	Description
ResourceService	Gestionnaire de version des services ressources

A1.2.14 Package utils

Tableau 7.16 : Liste de classe réservée pour Java

Class	Description
javaRuntime	Classe utilitaire pour accèder aux ressources de Runtime Java en cours
keyDispatcher	Utilitaire Java pour partager les événements des touches claviers
listnerObjectEvent	Classe de gestion des événements

A1.3 Interface

Elle définit l'ensemble des traitements ouverts pour accéder et réutiliser l'ensemble des services offerts par le Framework.

A1.3.1 Package database

Tableau 7.17 : Liste d'interface dédié aux bases de données

Interface
IConnection

```
package com.hitechs.commons.resources.database;

import com.hitechs.commons.resources.dataObject;
import com.hitechs.commons.resources.fieldUni;
import com.hitechs.commons.resources.tableUni;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.Date;
import java.util.List;

public interface IConnection {

    String getName();

    void setName(String name);

    short getTypeConnection();
```

```
String getUrlDriver();
void setUrlDriver(String urlDriver);
String getDriverName();
void setDriverName(String driverName);
Connection getConnection();
void setConnection(Connection con);
connectionParameter getConnectionParameter();
void setConnectionParameter(connectionParameter parameter);
int getIndex();
void setIndex(int index);
void openConnection();
void openConnection(String url,String login,String pwd);
boolean isClosed();
void closeConnection();
void startTransaction();
void commitTransaction();
void rollBackTransaction();
Date getDate();
PreparedStatement getStatement(String sql)throws Exception;
void addDataAuto(dataObject data);
void execute(String sql,List<fieldUni> values);
Object executeScalar(String sql,List<fieldUni> values);
List<dataObject> executeQuery (tableUni table,String sql,List<fieldUni> values);
public dataObject executeQuerySingle (tableUni table,String sql,List<fieldUni> values);
void executeProcedure(String procedureName,List<fieldUni> values);
```

```
    Object executeProcedureScalar(String procedureName,List<fieldUni> values);

    List<dataObject> executeProcedureQuery(tableUni table,String
procedureName,List<fieldUni> values);

    List<dataObject> executeQueryListPages(tableUni table,String whereClause,String
orderClause,int indexBegin, int plage);

    void addData(dataObject data);

    void updateData(dataObject data);

    void deleteData(dataObject data);

    Object executeScalarExtra(tableUni table, String sql);

    List<dataObject> executeQueryListExtra(tableUni table, String WhereClause,
List<orderDataTable> orderClause);

    List<dataObject> executeQueryListPagesExtra(tableUni table, String WhereClause,
List<orderDataTable> orderClause, int indexBegin, int plage);

    String getValueSql(fieldUni field,Object obj);

    String getValueOfWhere(fieldUni field, Object value);

    IMetaData getMeta();
}
```

IMetaData

```
package com.hitechs.commons.resources.database;
import com.hitechs.commons.resources.fieldUni;
import com.hitechs.commons.resources.tableUni;
import java.util.List;

public interface IMetaData {

    tableUni getTable(IConnection con,String table);

    List<tableUni> getListTable(IConnection con);

    List<fieldUni> getListFields(IConnection con,String table);

    String createTable(tableUni table);

    String dropTable(tableUni table);

}
```

A1.3.2 Package service

Tableau 7.18 : Liste d'interface pour accéder aux services

Interface
IBitmapService

```
package com.hitechs.commons.resources.service;

import java.awt.Component;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.awt.image.RenderedImage;
import java.io.FileNotFoundException;
import java.io.IOException;
import javax.imageio.ImageWriteParam;
import javax.imageio.ImageWriter;

public interface IBitmapService {

    String getExtensionImage(short typeImage);

     Image resize(Image source, int width, int height) ;

    BufferedImage imageToBufferedImage(Image image);

    int getPageImage(String path) throws FileNotFoundException, IOException ;

    BufferedImage tiffToBufferedImage(String path) throws FileNotFoundException,
IOException ;

    BufferedImage tiffToBufferedImage(String path,int page) throws FileNotFoundException,
IOException ;

    Image rotate(Image source, double angle) ;

    BufferedImage convertRenderedImage(RenderedImage img,double zox,double zoy);

    Image getImageInComponent(Component component);

    BufferedImage getCaptureScreen(int x, int y, int w, int h);

    BufferedImage getCaptureScreen(javax.swing.JFrame frame);

    BufferedImage getCaptureScreen();

    void saveImage(BufferedImage buf,String path,String tipa);

    ImageWriteParam getHeaderParameter(ImageWriter writer);
```

```
    void saveTiffImage(BufferedImage buf,String path) throws IOException;;

    BufferedImage copyImage(BufferedImage src, int imageType) ;
}
```

IDateService

```
package com.hitechs.commons.resources.service;

import java.util.ArrayList;
import java.util.Date;

public interface IDateService {

    Date getCurrentDay();

    String getCurrentDay(String format);

    int getCurrentDayOfMonth();

    int getCurrentDayOfWeek();

    int getCurrentMonth();

    int getCurrentYear();

    int getCurrentWeek();

    void setCurrentDay();

    String getFormatUser();

    void setFormatUser(String formatUser) ;

    int getDayOfMonth(Date date);

    int getDayOfWeek(Date date);

    int getMonth(Date date);

    int getYear(Date date);

    int getWeek(Date date);

    int getWeekOfMonth(Date date);

    Date stringToDate(String date,String format);

    String dateToString(Date date,String format);
```

```
    boolean isDate(String date,String format);
    Date addDays(Date date,int number);
    Date addMonths(Date date,int number);
    Date addYears(Date date,int number);
    Date getDateByWeek(int year, int week);
    Date getDateByWeek(int year, int week,int dayOfWeek);
    Date getDayInWeek(Date m_date,int dayOfWeek);
    Date getDayZeroHour(Date m_date);
    Date getDayEndOfMonth(int year,int month);
    int getMaxDateMonth(int year,int month);
    long getDayCountIntervale(Date start,Date end);
    int getYearCountIntervale(Date start,Date end);
    int getCountDayWeek(int month,int year,int dayWeek) ;
    String getListDays(Date start,Date end,String format,String separator);
    String getListDaysSql(Date start,Date end,String format);
    ArrayList<Date> getListDays(Date start,Date end);
    Date createDate(int year,int month,int day);
    Date changerHours(Date date,int hour,int minute,int seconde, boolean matin);
    String getDayOfWeek(Date date,int lang);
    String getDayOfWeekShort(Date date,int lang);
    String getDayOfWeek(int daty,int lang);
    String getDayOfWeekShort(int day,int lang);
    String getMonthOfYear(Date date,int lang);
    String getMonthOfYear(int mois,int lang);
    boolean isBissextile(int year) ;
}
```

IFileService

```
package com.hitechs.commons.resources.service;

import com.hitechs.commons.resources.dataUni;
import com.hitechs.commons.resources.file.fileInfo;
import com.hitechs.commons.resources.tableUni;
import com.hitechs.commons.resources.vectorUni;
import java.io.File;
import java.util.ArrayList;
import java.util.List;

public interface IFileService {

    boolean isExist(String path);

    boolean isDirectory(String path);

    void createDirectory(String path);

    String getExtension(String path);

    String getNameWithExtension(String path);

    String getNameWithOutExtension(String path);

    void deleteFile(String path);

    void copyFile(String source, String dest);

    void copyUtils(String source, String dest);

     boolean moveFile(String source, String dest);

    ArrayList <String> readLines(String file);

    ArrayList <Object> readCsv(String file,String separator);

    ArrayList <String> readLineInResource(String path);

    ArrayList <Object> readCsvInResource(String chemin,String separateur);

    String readAllText(String path);

    String readAllTextInResource(String chemin);

    void writeText(String path,String text,boolean append,boolean retour);

    void writeTextLines(String path,ArrayList<String> values,boolean append,boolean retour);
```

```
    void writeCsv(ArrayList <Object> contenu,String file,String separator);

    ArrayList<String> getListDirectory(File path);

    fileInfo getFileInfo(File file);

    ArrayList<fileInfo> getListFileInDirectory(String path);

    boolean copieDirectory(String source, String dest);

    boolean deleteDirectory(String path,boolean withParent);

    List lçoadProperties(String path);

    void saveProperties(ArrayList<vectorUni> list,String path,String name);

    void loadProperties(String path,dataUni data);

    void saveProperties(dataUni data,tableUni table, String path,String name);
}
```

ILDAPService

```
package com.hitechs.commons.resources.service;

import com.hitechs.commons.resources.accountLDAP;
import com.hitechs.commons.resources.ldapServer;
import java.util.ArrayList;

public interface ILDAPService {

    boolean isAll();

    void setAll(boolean all);

    ldapServer getServer();

    void setServer(ldapServer server);

    void init();

    boolean authenticate(String login,String pwd);

    ArrayList<accountLDAP> getAllAccounts();

    ArrayList<accountLDAP> getAllAccounts(String dn);

    ArrayList<accountLDAP> getAccountsByEmail(String mail);

    accountLDAP getAccountByEmail(String mail);
```

```
    accountLDAP getAccountByLogin(String login);

    accountLDAP getAccountByLogin(String dn,String login);

    ArrayList<accountLDAP> getAccountByName(String name);

    ArrayList<accountLDAP> getAccountsByName(String dn,String name);

    ArrayList<accountLDAP> getAllAccountGroup(String group);

    ArrayList<accountLDAP> getAccountsGroup(String group,String name);

    boolean isAccountGroup(String group,String login);

    void addAccount(accountLDAP account);

    void updateAccount(accountLDAP account);

}
```

IMailService

```
package com.hitechs.commons.resources.service;

import com.hitechs.commons.resources.dataMail;
import com.hitechs.commons.resources.headerUni;
import java.util.ArrayList;

public interface IMailService {

    String getMailHost();

    void setMailHost(String mailHost);

    String getPort();

    void setPort(String port);

    boolean isVerbose();

    void setVerbose(boolean verbose);

    boolean isDebug();

    void setDebug(boolean debug);

    boolean isHtml();

    void setHtml(boolean html);
```

```
    String getContentType();

    void setContentType(String contentType);

    dataMail getNewEmail();

    void sendEmail(dataMail mail);

    ArrayList<dataMail>readMails(String address,headerUni criteria);

}
```

IPDFService

```
package com.hitechs.commons.resources.service;

import java.io.File;
import java.io.InputStream;

public interface IPDFService {

    String PDFToText(String path);

    String PDFToText(String path, int startPage,int endPage);

    void mergePDF(String fileName1,String fileName2,String fileName);

    void JpegToPDF(InputStream stream,String fileName);

    void tiffToPDF(File file,String fileName);

    void tiffToPDF(File file,String fileName,int page);

    void PDFToTiff(InputStream stream,String fileName);
}
```

ISheetService

```
package com.hitechs.commons.resources.service;

import com.hitechs.commons.resources.binaryFile;
import com.hitechs.commons.resources.component.grid.metaColumn;
import com.hitechs.commons.resources.office.cellHeader;
import com.hitechs.commons.resources.office.cellInfo;
import com.hitechs.commons.resources.office.rangeInfo;
import com.hitechs.commons.resources.property.UIProperty;
import java.io.InputStream;
import java.util.ArrayList;
public interface ISheetService {
```

```
    void openNewBook();

    void openBook(InputStream stream);

    void openBook(String path);

    void openModelBook(String pathModel,String realPath);

    void createSheet(String name,ArrayList<metaColumn> columns,int column,int
row,ArrayList<UIProperty> properties);

    void createSheet(String name,ArrayList<cellHeader> columns);

    void fillSimpleBorder(int page,int firstRow,int lastRow,int firstCol,int lastCol);

    String getCellValue(int page, int row, int col);

    int getColumnMax(int page);

    int getRowMax(int page);

    int getSheetCount();

    void setColumnWidth(int page,int column,int width);

    ArrayList<Integer> getColumnsWidth(int page);

    void fillSheetPerRow(int page,ArrayList<rangeInfo> liste,ArrayList<UIProperty> properties);

    void fillSheetPerCell(int page,ArrayList<cellInfo> liste,ArrayList<UIProperty> properties);

    void insertImage(binaryFile image,String extension,int page,int column,int row);

    void mergeCells(int page,rangeInfo range);

    void fillRange(int page,rangeInfo range);

    void setStyleRange(int page,rangeInfo range,ArrayList<UIProperty> properties);

    ArrayList<String> readListByRow(int page, int row);

    void saveToFile(String path);

    void saveToPdf(String path);

    void close();
}
```

ITraductorValue

```
package com.hitechs.commons.resources.service;

import com.hitechs.commons.resources.dataUni;

public interface ITraductorValue {

    void setMultipleIn(boolean multipleIn);

    boolean isMultipleIn();

    void addValue(Object code,Object value);

    Object getValue(Object code);

    Object getValue(dataUni data);
}
```

A1.3.3 Package utils

Tableau 7.19 : Liste d'interface aux événements

Interface	**Description**

IEventObject	package com.hitechs.commons.utils;
	import java.util.EventListener;
	
	public interface IEventObject extends EventListener{
	
	public void doEvent(Object obj);
	
	}

IFiredKeyEvent	package com.hitechs.commons.utils;
	
	public interface IFiredKeyEvent {
	
	public void firedKeyEvent(java.awt.event.KeyEvent evt);
	
	}

ANNEXE 2 : BPMN

BPM (Business Process Management) est une discipline qui consiste à considérer la gestion des processus comme un moyen d'améliorer la performance opérationnelle. Les processus métier sont représentés sous forme de modèles graphiques grâce à l'ensemble des conventions graphiques BPMN (BPMN Business Process Model and Notation).

A2.1 Version BPMN 2.0

BPMN 2.0 est un langage de modélisation qui remplace BPMN 1.x et BPEL pour exécuter les processus modélisés.

Il contribue à l'amélioration de la norme BPMN 1.x en apportant :

- un méta modèle normalisé et un format de sérialisation pour BPMN, qui permet aux utilisateurs d'échanger des modèles BPMN entre les outils de différents fournisseurs.
- une sémantique d'exécutions normalisées pour BPMN, qui va permettre aux fournisseurs logiciels d'implémenter des moteurs d'exécution interopérables pour les processus métier.
- un format d'échange graphique, permettant aux utilisateurs d'échanger les informations graphiques d'un diagramme de processus métiers
- une notation étendue pour les interactions inter-organisationnelles (également connues sous le nom de chorégraphies processus), qui permettra de créer de nouveaux cas d'utilisation pour les outils automatisés de soutien pour les processus qui impliquent plusieurs partenaires.
- un processus de transfert détaillé de BPMN pour WS-BPEL, montrant l'alignement de BPMN avec les outils et les normes existants
- certains éléments de modélisation supplémentaires pour des processus tels les événements et sous-processus non-interrompus.

A2.2 Fonctionnement du langage

BPMN est constitué d'un ensemble d'éléments graphiques. Ces éléments permettent le développement de schémas simples. Les éléments ont été choisis pour se distinguer les uns des autres. Par exemple, les activités sont des rectangles, et les décisions sont des diamants. Les différents éléments graphiques de la notation sont organisés selon un concept de layer (couche) extensible. Chaque couche est construite en se basant sur la couche précédente.

A2.3 Élément de notation

La spécification de BPMN 2.0 définit les symboles qui seront utilisés pour créer des modèles de processus. La spécification décrit la syntaxe et la sémantique de tous les symboles. Dans la pratique, les modélisateurs de processus sont libres de choisir des symboles pour modéliser.

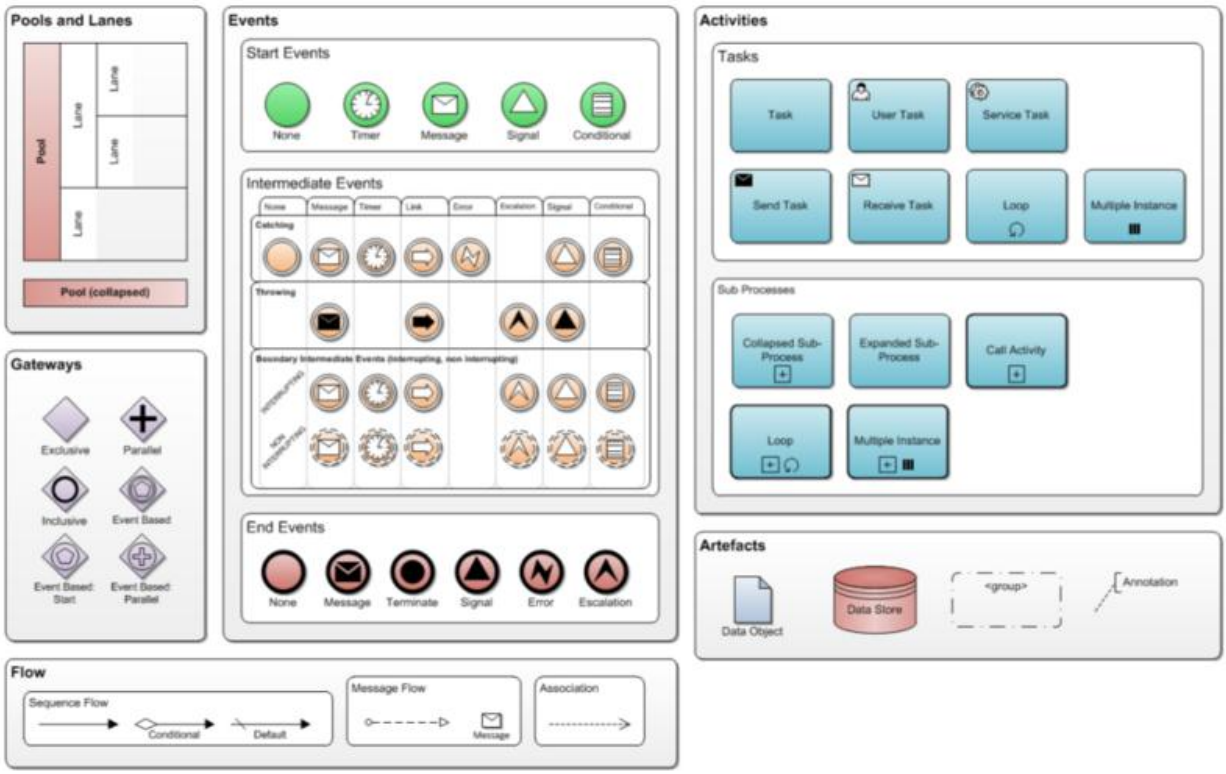


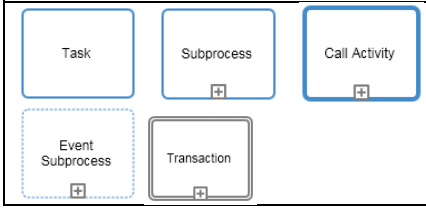
Figure 8.1 : Vue globale des icônes BPMn

La spécification BPMN 2.0 fournit trois sous-ensembles prédéfinis, appelé sous-classe de conformité (conformance sub-classes) :

- descriptif : éléments et attributs nécessaires à la description d'un processus de haut niveau.
- analytique
- exécutables

A2.3.1 Généralité

Tableau 8.1 : Groupe d'élément BPMn

Symbole	Elément
	Participant : regroupe par profile
	*Artifacts* : Informations descriptive
	*Gateways* : mode d'unification ou séparation
	Data : représente une source de données
	*Activities* : définit une tache ou ensemble de tâches

A2.3.2 Activités

Ce type d'objet permet d'identifier un processus, un sous-processus, une tâche. Une icône présente sur l'activité permet de comprendre visuellement à quel niveau nous sommes. Des icônes complémentaires permettent de distinguer les différents types de tâches (manuel, automatique, soumis à une règle métier, envoi, réception…)

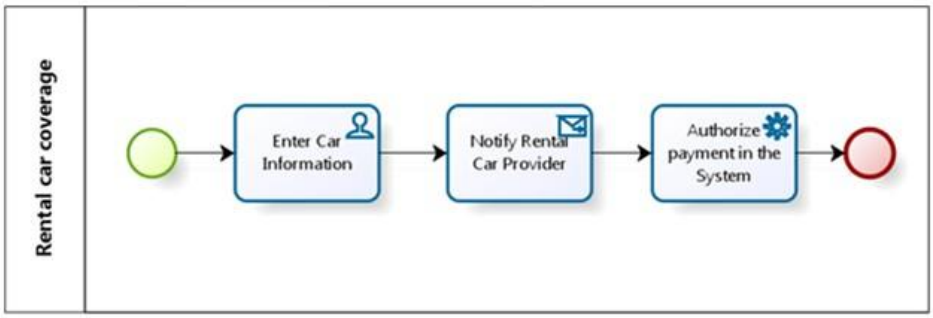


Figure 8.2 : Succession des activités dans un diagramme

A2.3.3 Connecteurs

Les connecteurs permettent d'interpréter les flux : ET, OU exclusif, OU inclusif… L'objet est représenté sous forme de diamant et sa signification est illustrée par l'icône à l'intérieur.

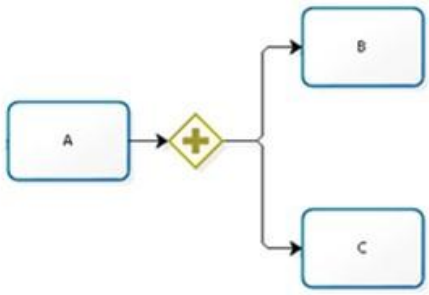


Figure 8.3 : Exemple d'utilisation d'un connecteur

A2.3.4 Evènements

Un événement est quelque chose qui survient. Il déclenche une action ou est le résultat d'une action. BPMN met à disposition 3 catégories d'événements, événement de début, de fin et intermédiaire, qui sont enrichis par une icône permettant d'illustrer le type d'événement (message, temporel, regroupant un sous-ensemble d'événements, …).

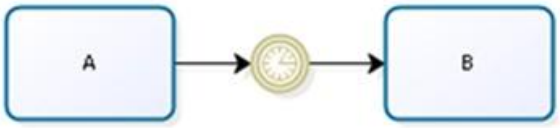


Figure 8.4 : Exemple d'utilisation d'un événement

Tableau 8.2 : Liste des événements BPMn

Elément	**Rôle**	**Start**	**Inter**	**End**
Event	Evénement quelconque.			
Message	Evénement impliquant la réception ou l'envoi d'un message.			
Error	Evénement impliquant le déclenchement d'une erreur.	N/A		
Cancel	Evénement déclenché lors de l'annulation d'une transaction.	N/A		
Compensation	Evénement déclenché lors de l'échec d'une transaction.	N/A		
Timer	Evénement lié à une date spécifique ou à un cycle temporel.			N/A
Rule	Evénement déclenché lorsqu'une règle devient vraie.			N/A
Link	Evénement permettant de connecter la fin d'un processus au début d'un autre.			

Multiple	Evénement résultant de la composition d'autres types d'événements.			
Terminate	Evénement utilisé pour indiquer que toute activité sur le processus doit être immédiatement suspendu.	N/A	N/A	

A2.3.5 Les Artifacts

Ce sont des objets additionnels utilisés pour mieux comprendre le schéma BPMN : comme les activités de même catégorie, les données traitées ou bien des commentaires ou annotations.

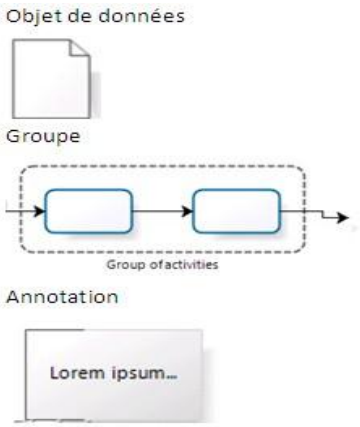


Figure 8.5 : Exemple d'utilisation d'artéfact

Tableau 8.3 : Liste des artéfacts BPMn

Elément	**Rôle**	**Icône de création**
DataObject	Artéfact produit et consommé par les activités du processus.	
Transaction	Une "Transaction" est une région du diagramme utilisée pour représenter l'existence d'une transaction englobant toutes les activités présentes.	
Group	Un "Group" est une région du diagramme utilisée pour rassembler des éléments du processus.	

A2.3.6 Les flux

Les flux permettent de relier et séquencer les éléments du processus entre eux.

Tableau 8.4 : Liste des connecteurs BPMn

Elément	Rôle	Icône de création
Flow	Flux utilisé pour séquencer deux activités du processus.	
ConditionalFlow	Flux conditionnel.	
DefaultFlow	Flux par défaut.	
MessageFlow	Flux utilisé pour faire transiter un message entre deux entités du processus.	

A2.4 Diagramme

BPMn dispose quatre configurations pour présenter un diagramme :

- privé ou public
- collaboration
- chorégraphie
- de conversation

A2.4.1 Diagramme Privé

Un processus privé est un type de modélisation qui permet de représenter un processus spécifique à une organisation en précisant que son contenu reste dans les limites du bassin et ne peut pas le traverser.
Le flux de messages peut traverser les limites du bassin pour montrer les interactions qui existent entre des processus privés qui sont séparés.

A2.4.2 Diagramme public

Le processus public est un processus privé plus des interactions avec à un ou plusieurs Participants en définissant les flux de messages, leur séquence, leur ordre etc.
Seules les activités de communication avec l'autre(s) participant sont présentées dans le processus public. Toutes les autres activités ou tâches internes du processus privé ne sont pas représentées.

A2.4.3 Collaboration

Le diagramme de collaboration permet de modéliser les processus et les interactions entre 2 entités au minimum.

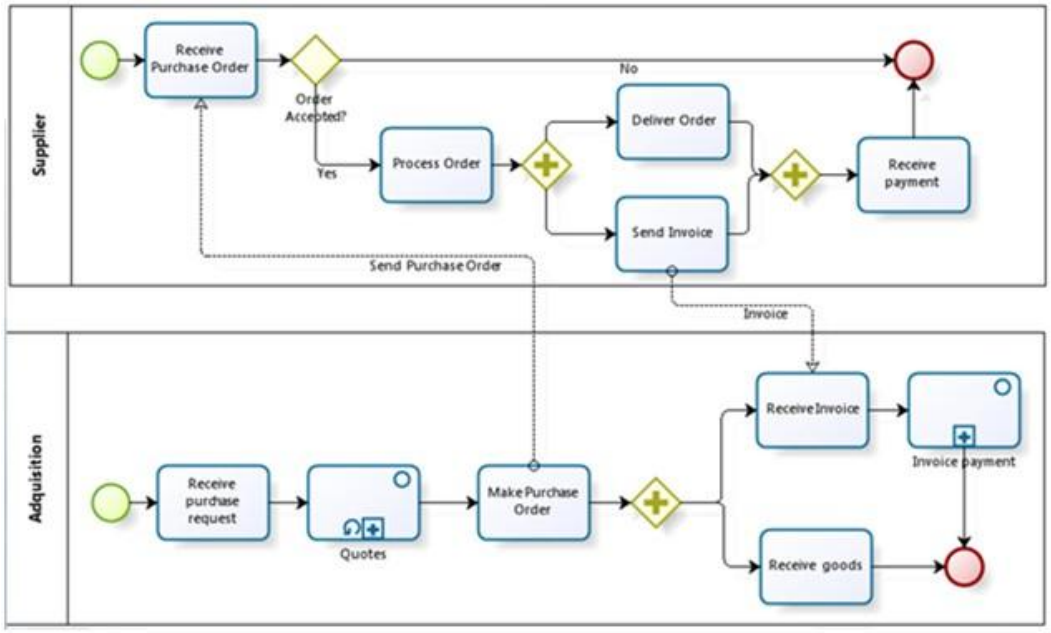


Figure 8.6 : Exemple diagramme de collaboration

Un diagramme de collaboration ne peut contenir qu'un seul bassin (département, service). Ce bassin peut contenir plus d'un couloir (qui correspondent à des rôles : manager, assistante, collaborateur). Pour chaque couloir des activités sont définies.
Pour illustrer les échanges entre 2 couloirs, on utilise des liaisons spécifiques appelées flux de message.

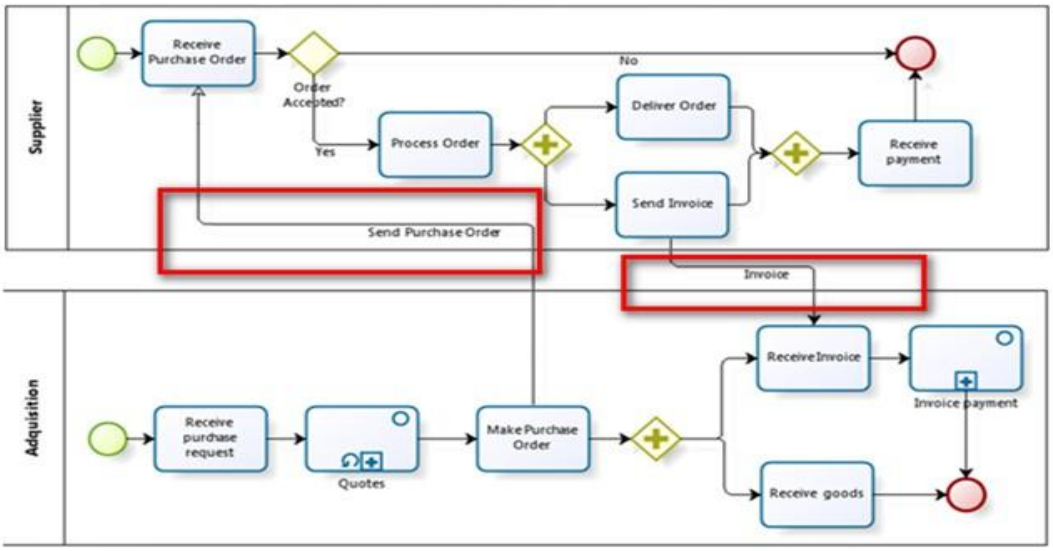


Figure 8.7 : Flux de message dans un diagramme de collaboration

A2.4.4 Diagrammes de chorégraphie

Un diagramme de chorégraphie est utilisé pour analyser la façon dont les participants échangent des informations afin de coordonner leurs interactions. On peut utiliser un diagramme de chorégraphie afin de développer et d'analyser en détails l'échange des messages associés à un nœud de conversation dans un diagramme de conversation.

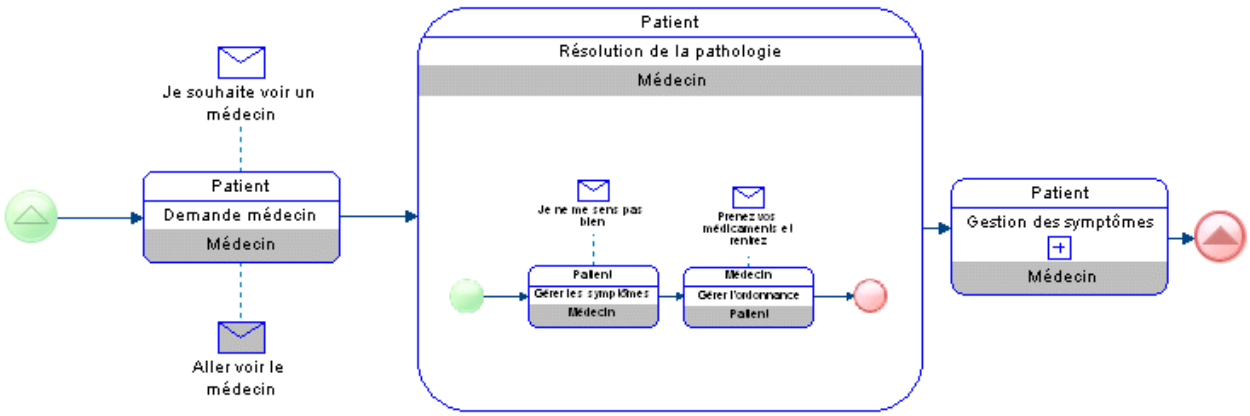


Figure 8.8 : Exemple d'un diagramme de chorégraphie

Cet exemple illustre l'échange de messages entre un patient et son médecin.

Une Chorégraphie est la modélisation d'un comportement attendu entre des Participants qui interagissent les uns avec les autres et qui veulent coordonner leurs activités ou leurs tâches à l'aide de Messages. Dans ce type de modélisation, la focalisation n'est pas sur l'Orchestration (processus public ou privé), c'est-à-dire sur la manière dont est accompli le travail selon le point de vue des participants, mais sur les échanges de Messages entre les Participants.

A2.4.5 Diagrammes de conversation

Le Diagramme de conversation est la description informelle d'un Diagramme de collaboration à haut niveau. Il représente des échanges de Messages (présentés sous la forme d'un regroupement de flux de messages), logiquement reliés entre les Participants et qui concernent un objet d'affaires d'intérêt, par exemple, un ordre, un envoi, une livraison ou une facture. Il représente un ensemble de Flux de Messages qui est regroupé ensemble. Une Conversation peut impliquer deux ou plusieurs participants.
Un diagramme de conversation représente les conversations comme des hexagones, entre les participants (les bassins). Si l'on clique sur l'hexagone, on peut avoir le détail des Flux de Message échangés entre les Participants.
Les bassins qui servent à la représentation d'un Diagramme de conversation, ne contiennent habituellement pas de processus et sont donc présentés vides.
Aucun Diagramme de chorégraphie ne peut être placé entre les bassins du Diagramme de conversation.

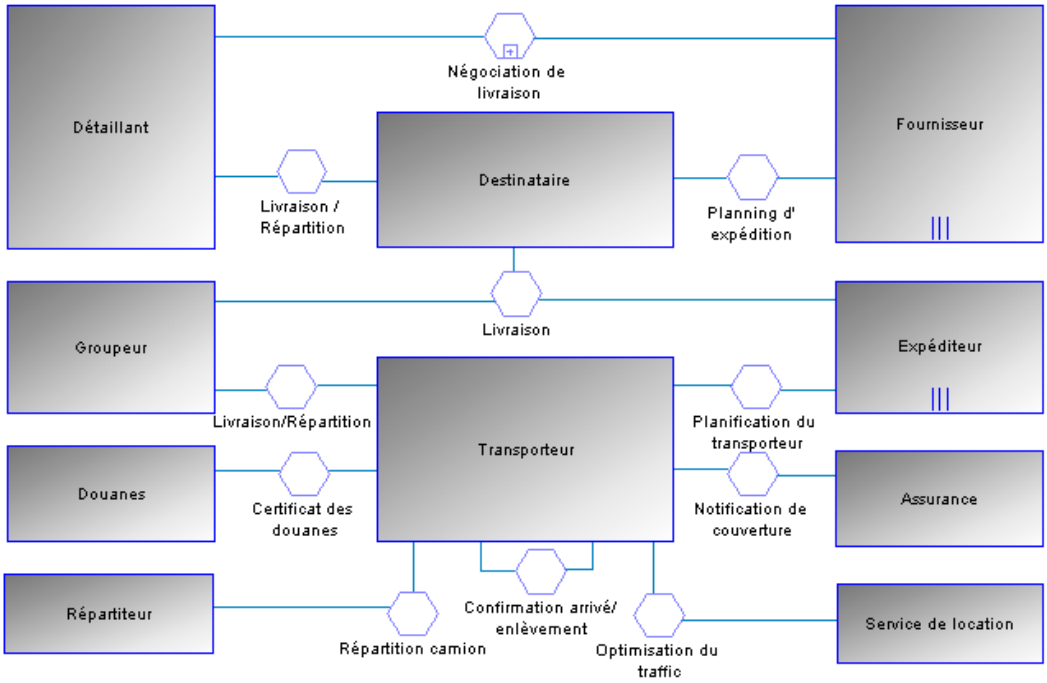


Figure 8.9 : Exemple d'un diagramme de conversation

Dans cet exemple, les diverses conversations associées aux livraisons d'un fournisseur à un détaillant sont analysées.

ANNEXE 3 : Architecture Bus d'événement

L'ESB (Enterprise Service Bus) est le composant central d'une architecture orientée services (SOA). Il autorise une approche de l'intégration hautement distribuée et indépendante de la plate-forme. Il s'appuie sur les standards pour soutenir la messagerie, les Services Web, le routage et la transformation de données.

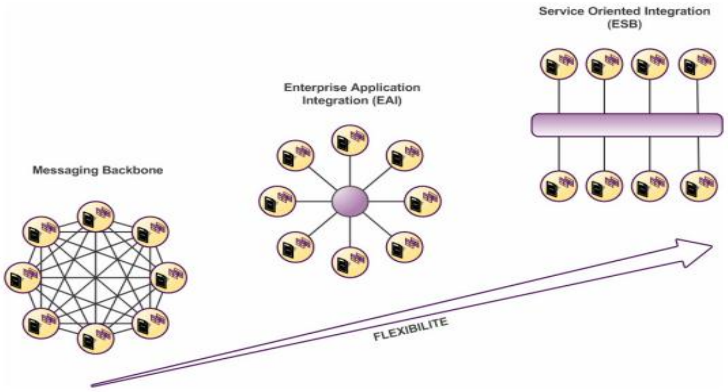


Figure 9.1 : Evolution de l'architecture de partage de message

A3.1 Architecture

Le bus de services se base sur des principes suivants :

- la découverte dynamique
- l'exposition et l'orchestration des services
- outil graphique permettant d'implémenter les processus et de les orchestrer
- la distribution forte
- la communication par message
- la communication entre services se fait par message représenté par un document auquel le service consommateur s'abonne.

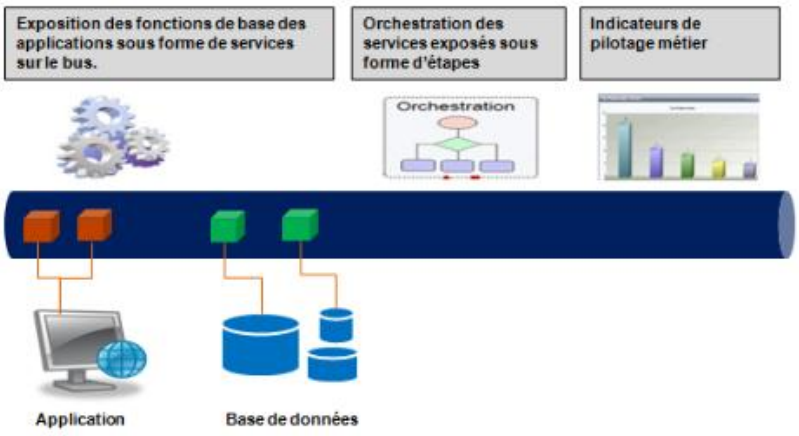


Figure 9.2 : Les éléments liés aux Bus

Son architecture est basée sur :

- échange asynchrone entre applications
- interopérabilité entre le bus et les applications grâce aux services
- transformation et enrichissement des données circulant sur le bus

ESB se concentrent sur les fonctions d'interconnexion et de médiation, et s'appuient pour cela sur un ensemble de standards parmi lesquels :

- Web Services pour gérer les communications synchrones.
- XML pour définir les formats des messages
- JMS2 pour adresser la communication asynchrone avec les MOM.
- JCA pour la connexion aux progiciels et systèmes exotiques (ERP, CRM, Mainframes, etc.)

A3.2 Fonctionnement

Le bus de service d'entreprise achemine les messages entre les applications, qui sont demandeurs ou fournisseurs de services. Le bus convertit les protocoles de transport ainsi que les formats des messages entre les demandeurs et les fournisseurs. Dans ce schéma, chaque application utilise un protocole différent (représenté par les différentes formes géométriques de leurs connecteurs) et utilise différents formats de message.

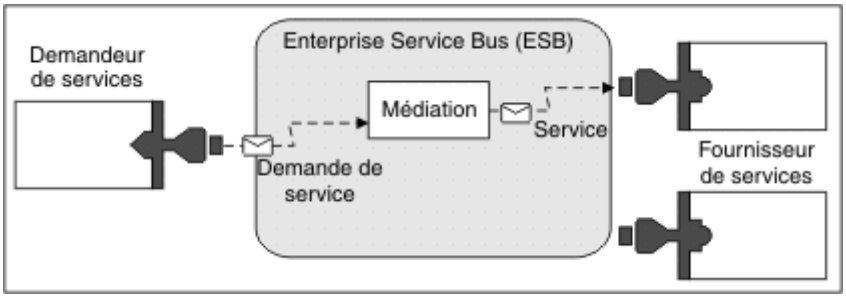


Figure 9.3 : Echange des informations au niveau du Bus

A l'aide des médiations, un bus de service d'entreprise exécute les actions suivantes entre le demandeur et le service :

- acheminement des messages entre les services : le bus de service d'entreprise offre une infrastructure de communication commune permettant de se connecter aux services et donc aux fonctions métier qu'ils représentent, sans que les programmeurs aient à écrire et gérer une logique de connectivité complexe.
- conversion des protocoles de transport entre le demandeur et le service : le bus de service d'entreprise est un moyen cohérent normalisé d'intégrer des fonctions métier qui utilisent des normes informatiques différentes. Cela permet de disposer de fonctions métier qui ne pourraient normalement pas communiquer, telles que la connexion d'applications dans

des silos départementaux ou la participation des applications de différentes sociétés aux interactions de service.
- conversion des formats de message entre le demandeur et le service : le bus de service d'entreprise permet aux fonctions métier de d'échanger des informations dans des formats différents, le bus garantissant que l'information distribuée à une fonction métier a le format correspondant à l'application.
- traitement des événements métier provenant de sources différentes : le bus de service d'entreprise prend en charge les interactions basées sur l'événement en plus des échanges de message pour traiter les demandes de service.

A3.3 Médiation et routage

La médiation est la principale valeur ajoutée d'un ESB. Dans ce domaine, il doit proposer une sémantique riche, susceptible de fiabiliser et de simplifier les échanges, tout en offrant une grande souplesse de mise en œuvre.
Les principales fonctionnalités que l'on peut attendre d'un ESB en matière de médiation et de routage :

- support de différentes sémantiques d'échange de messages (synchrone « requête-réponse », asynchrone point-à-point, asynchrone selon le modèle « publication-souscription »)
- gestion de règles de routage sur les messages.
- mécanismes de gestion de la priorité des messages.
- services de transformation et de conversion de messages (ex : XML <-> EDI, etc.).
- manipulation des messages (enrichissement, transformation, combinaison, découpage).
- validation des données entrantes ou sortantes.
- gestion de versions de services de façon transparente (systèmes évoluant à des rythmes différents).

A3.4 Apports du bus de services

Le concept service bus permet de mettre en place un système capable de :

- intégrer de concepts standards
- router de données

L'utilisation des documents XML et son mécanisme par le service bus permet de :

- d'avoir une architecture distribuée : Les fonctions du SI sont vues comme des briques réutilisables et faiblement couplées, donc pouvant être déployées indépendamment les uns des autres. Ces briques peuvent être orchestrées dans le cadre d'un processus technique ou métier.
- souplesse : Interface centralisée de gestion de configuration et de déploiement.

A3.5 JBI

JBI est une solution java pour implémenter une plateforme de bus d'événement. Il définit deux types de composants :

- les Service Engine (SE) qui sont des composants qui fournissent le logique métier et des services de transformation aux autres composants. Ils peuvent également être consommateur d'autres Service Engine.
- les Binding Component (BC) qui sont des composants qui assurent la connectivité pour les services extérieurs à l'environnement JBI. Cela peut impliquer les protocoles de communication ou des services fournis par le SI de l'entreprise.

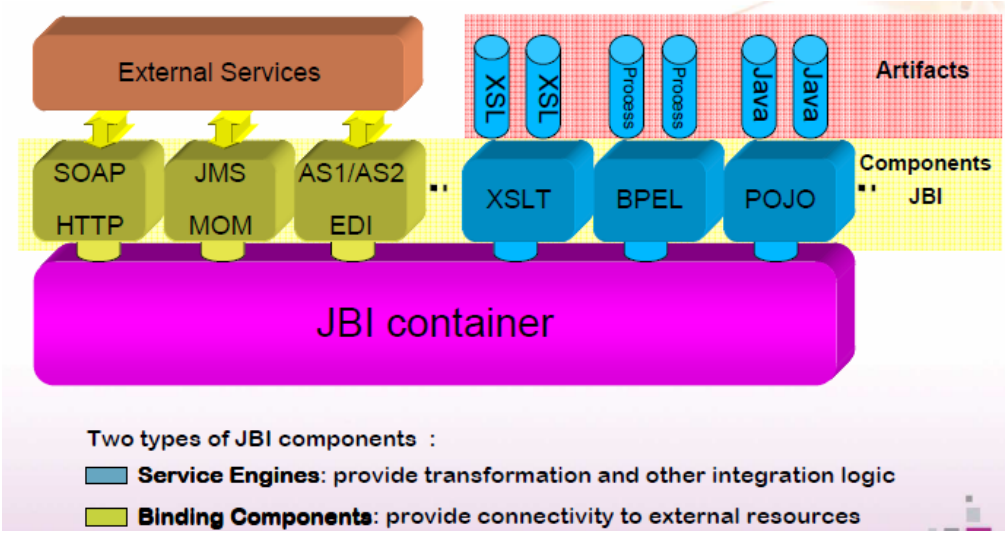


Figure 9.4 : Composants JBI

Exemple de SE :

- SE XSLT (eXtensible Stylesheet Language Transformations) qui permet d'appliquer une transformation d'un fichier/message XML à partir d'une feuille de style XSL,
- SE Validator qui permet de valider un message grâce à son XSD,
- SE CSV (Comma-Separated Value) qui permet de transformer un document au format CSV en document au format XML,
- SE EIP (Enterprise Integration Pattern) qui permet d'implémenter certains patrons d'intégration d'entreprise,
- SE JSR181 (Web Services Metadata for the java Platform) qui permet d'exposer un POJO (Plain Old Java Object) annoté comme un service JBI,
- SE POJO (Plain Old Java Object) qui permet de déployer des classes Java comme des services,
- SE BPEL (Business Process Execution Language) qui permet d'offrir un moteur d'exécution BPEL,
- SE SCA (Service Component Architecture) qui permet d'offrir une implémentation de SCA,

Exemple de BC :

- BC FileTransfert qui permet d'envoyer et de recevoir des fichiers,
- BC FTP (File Transfert Protocol) qui permet le transfert et le listage de fichiers sur un serveur ftp,
- BC JMS (Java Message Service) qui permet de s'interconnecter avec des Destinations (Queue/Topic) extérieures,
- BC SMTP qui permet d'émettre et de recevoir des fichiers d'un service de mail extérieur,
- BC SOAP (Simple Object Access Protocol) qui permet de s'interconnecter avec un service Web extérieur afin d'exposer des services JBI comme des services Web,
- BC XMPP (eXtensible Messaging and Presence Protocol) qui permet de s'interconnecter avec le protocole XMPP qui est un protocole standard de messagerie instantané (Google Talk, Jabber, …),

A3.5.1 Message

Un JBI Provider permet d'émettre et de recevoir des messages (ME) entre les divers services qu'il héberge au travers de composants. En fait, ces transmissions de messages suivent certains patterns qui sont définis au nombre de 4 (ce nombre n'est pas imposé et chaque JBI Provider est libre d'offrir plus) :

- In-Only

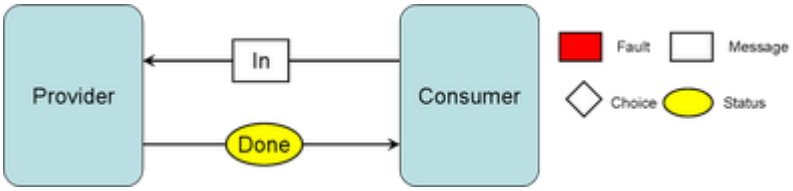


Figure 9.5 : Message « In only »

- In-Out

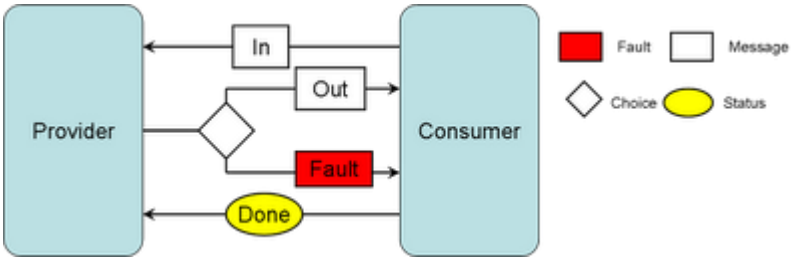


Figure 9.6 : Message « In/Out »

- In-Optional Out

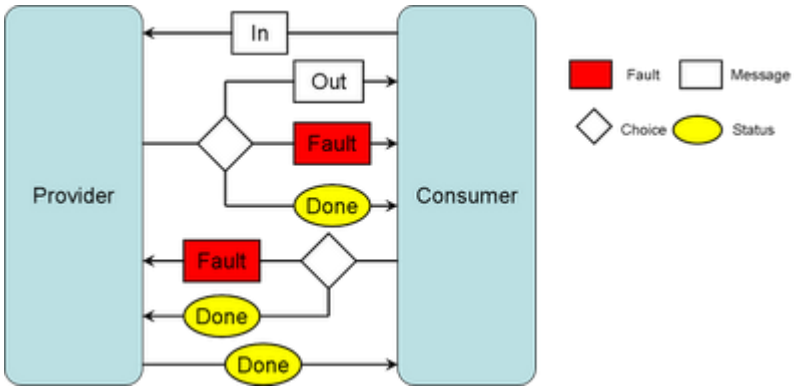


Figure 9.7 : Message « In optional Out »

- Robust In-Only

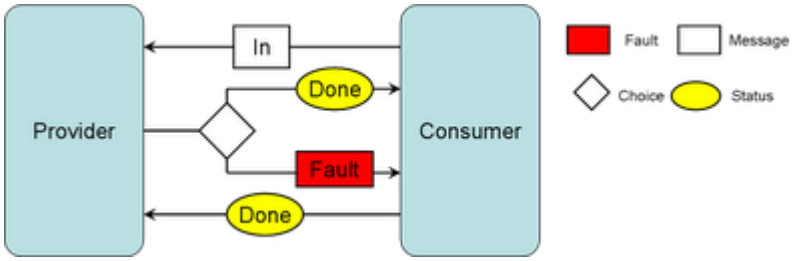


Figure 9.8: Message « robust In only »

Ce pattern d'échange (Message Exchange Pattern – MEP) est positionné par le composant émetteur (SU), et c'est au composant recevant le message de préciser les patterns qu'il supporte et comment il les supporte (par exemple, un BC JMS en mode fournisseur ne fournit pas de réponse s'il ne fait qu'émettre. Aussi, les MEPs qu'il supporte ne peuvent être que In-Only ou Robust In-Only).

A3.5.2 Cycle de vie du composant JBI

Un composant après son déploiement sur le système peut passer de trois états :

- démarré
- arrêté
- fermé

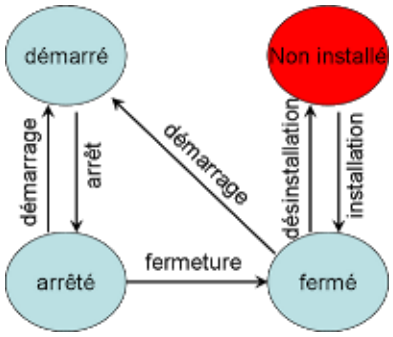


Figure 9.9 : cycle de vie d'un composant JBI

A3.5.3 Les composants : conteneurs de services

Pour qu'un service puisse s'exécuter dans un conteneur JBI il doit être en quelque sorte configurés et instanciés pour s'exécuter. Les composants agissent ainsi comme des conteneurs pour des instances de service :

- les instances des services sont packagées sous la forme de Service Unit (SU). Ces SU sont ensuite déployés dans l'environnement d'exécution JBI qui va installer chaque instance dans le composant (BC ou SE) fournissant le service.
- un SU d'un service exposé contient les parties suivantes :
    - Jbi.xml *deployment descriptor* : ce descripteur de déploiement est consommé par le moteur CXF qui est responsable de l'instanciation et l'activation du service.
    - service *implementation* : les classes d'implémentation du service incluant le code stub de WSDL.
    - WSDL *contract* : la copie serveur du contrat qui utilise le format *xformat binding* et je transport JBI destiné à communiquer avec le NMR.

Les SU doivent être packagés sous la forme de Service *Assembly* (SA) pour être « déployable ».

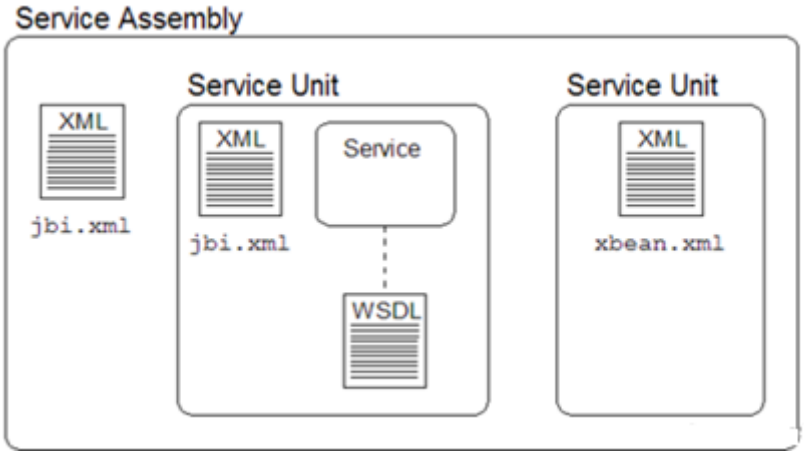


Figure 9.10 : Empaquetage de composant

ANNEXE 4 : Publications de notoriété nationale et internationale

A4.1 Modelling of Knowledge of Management

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 03, March-2017

Management Tool of Information System Modelling of Knowledge of Management

Razafindraibe Tovohery. A. A[1] , Rastefano E.[2] , Rabeherimanana L.I.[3]
Embedded Systems - Instrumentation - Modelling of the Systems and Electronic Devices (SE-I-MSDE)
ED STII - University of Antananarivo
BP on 1500, Ankatso - Antananarivo 101 – Madagascar

***Abstract*— A tool for management an information system is very sensitive organ for the survival of the organization. Using the concept of the cognitive domain, we propose the introduction of guided learning and Case remembrance in the control system to complement the indicators dashboards.**

This capability of the information system becomes possible by adopting a service-oriented platform based on a BPMS engine. Each decision to improve or optimize becomes a memorable case and can reuse according to the context present in the dashboard and in the structure of the whole system.

Defining a cross-cutting decision to achieve an objective is a delicate and complex process; it requires an understanding of the current state of the system and the effect of this or that action on the basic axis of the system: resources, processes, business, finance, client, etc.

Keywords*— *Information System(IS), Knowledge, Case, reasoning, management, decision, BPMS

I. INTRODUCTION

In the domain of information system, the knowledge is a data structure which crossed the stage of information, transformation and check to be able to define and control a defined activity.

This kind of data represents the most important aspect of the information and its value in an information system guaranteed the maturity and the stability of the organization.

As the system is based on various entities (resources, processes, actors), the most of them possess several types of knowledge embedded on a sub-system or linked to an actor or specific resource. Technically, these knowledge and skill are managed by the ERP and the processes related to the sub system.

According the role of system of piloting in the management of IS: key element of the innovation and implementation processes; the orientation of the evolutions of the system needs a tool to manage and remember piloting knowledge. It becomes possible with the combination of three systems: business process management, business intelligent and Case remember.

II. MODELLING INFORMATION SYSTEM

A. Data management

As the knowledge is a data structure, the feasibility of the modelling depends on the management of the existing data source:

- data input (problems, pattern and targets)
- data output (process, application, and resolution):

In our approach of modelling which is based on the experience, we categorize these data in three classes:

- design pattern
- resource
- case of decision

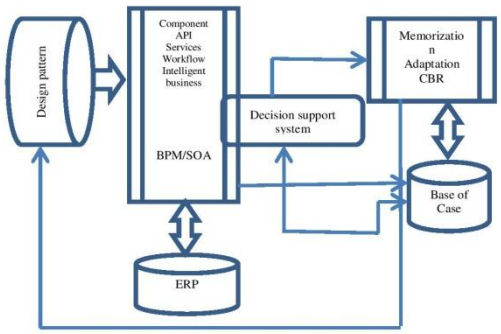


Fig. 1. Relation between data source and Tools in IS

By adopting the concept of management by process and architecture SOA, the information system which is a dynamic and distributed system can be defined by the following system:

- process management
- workflow
- business intelligent
- service provider

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 03, March-2017

The pilotage system includes:

- decision support system : Dashboard and Cockpit
- memorization system : sore and remember case of decision

B. Model of Information Sytem

The modelling based on the concept process gives a clear and simple view of the information system. The mapping of process allows to identify the sequence of treatment on the whole system and it offers a big flexibility to pilot and manage changes.

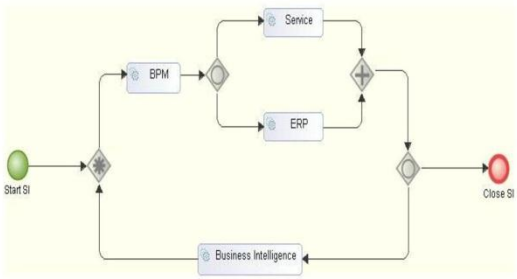


Fig. 2. Model of Information System

The system is defined as a closed loop:

- the modelling of the system starts with design of the processes. This approach is iterative because the system changes in the time and in the space. The BPM (Business Process Management) present as a kind of interface and artery which connects the operational system and the management system.
- the engine workflow interacts with a system SOA (Oriented Architecture service) to supply data to the pilotage systems (BI).
- the BI handles indicators and generates the dashboard. With this information, the pilotage system defines a new decision and new resource to reach the goals.

III. SYSTEM ORIENTED SERVICE

The architecture SOA offers to the system its capacity of interaction and flexibility.

A. Distributed system

The engine workflow generates discreet independent task that the oriented system service allows to have a dialogue to implement the job via the services and the resource in ERP (Enterprise Resource Planning).

The oriented service system defined the kernel of the system; it allows to connect all the elements of the system by bringing the physical implementation of each treatment.

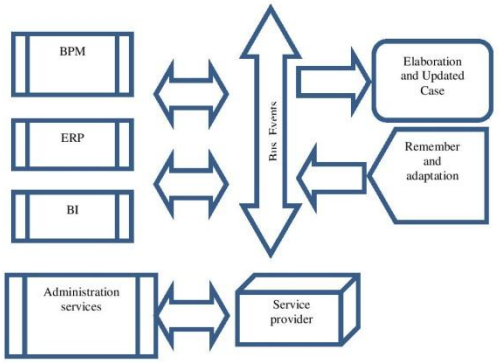


Fig. 3. Coupling of IS by a oriented service system

With the SOA architecture, a Bus event assumes the following responsibilities:

- listen and manage event
- implement interface
- transport data
- manage connection
- save log

B. Framework

By using the design pattern concept, we can define software layers capable to support the recommendations defined by the coupling of BPMS, SOA and ERP.

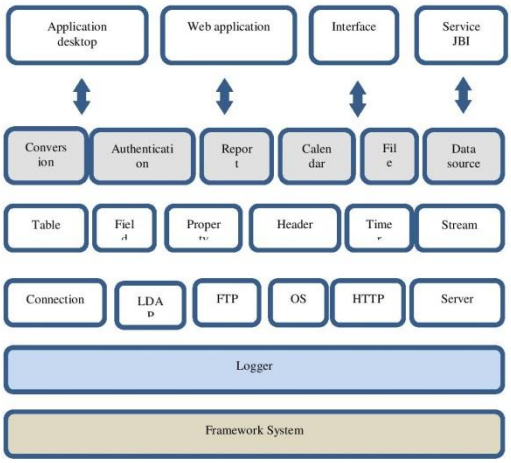


Fig. 4. Framework for basis data treatment

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 03, March-2017

The information system has two kinds of applications:

- static: Include all applications of the core of the system.
- dynamic: Include all services and application used by the process

To assure the cohabitation of these applications, they have to use the same framework.

IV. STATE OF THE SYSTEM

The pilotage knowledge involves the knowledge of the state of the system at the time of the decision-making. By using four axes of the model of the system (resource, organization chart, processes, services), we can define the state of the system by adding the fifth axis which is the axis of indicators.

A. Axis process

This axis is defined by the couple of processes and product. It shows the relation between the list of the products of the system with its processes and the relations inter- process (parent and sub process).

With this axis, we can estimate the performance of a process according to its nature and its relations with the final products.

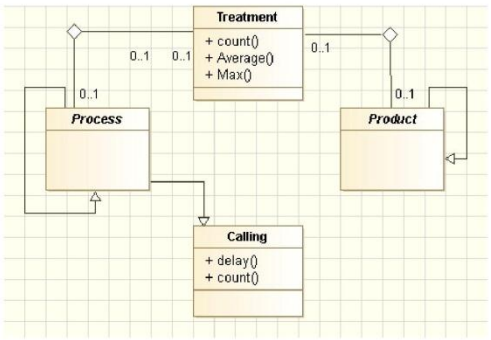


Fig. 5. Dimension of axis process

B. Axis organization

The axis organization highlighted the relation between the organization chart (hierarchy and area) and the processes which they use. The competence of each entity is defined by this axis.

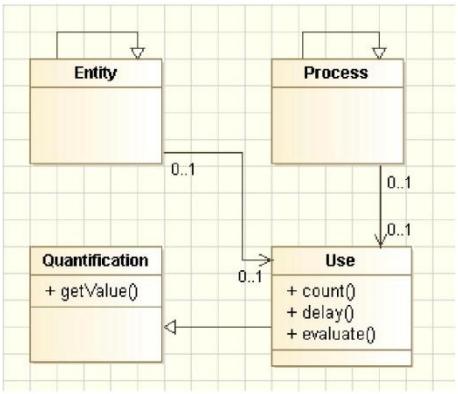


Fig. 6. Dimension of axis organization

With a specific tool, the IS manage the list of all profiles that can use (or run) a task in a process. This axis use this tool to fill and compute data.

C. Axis application

In the information system, data generation is always via an application and service. This axis allows to define the relation between the list of the applications which turn in the system and the entities (organization chart) which use it.

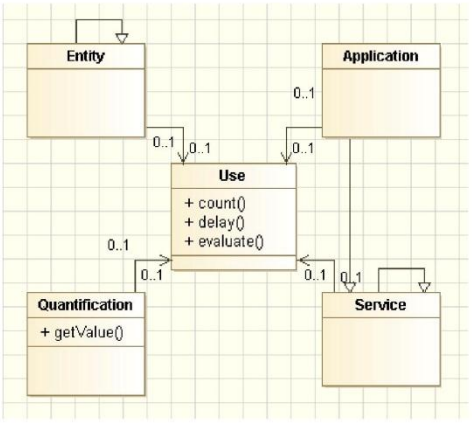


Fig. 7. Dimension of axis application

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 03, March-2017

D. Axis resource

This axis is identical to the axis application but it concerns only the resources non application software:

- material,
- automat,
- raw material,
- etc. …

The data concerning this axis have very different life cycles because its depreciation depends of the type and the process defined in the ERP.

V. STEERING - DASHBOARD COCKPIT

Dashboards Cockpit allows to translate the strategy into operational terms to improve the method of decision-making and the performance of the organization.

They allow understanding the strategy and the business to identify:

- key indicators,
- the information and the needs to make the operational decisions,
- the performance

To decrease in the maximums the risks, it is necessary to add the management of the experience in this methodology. The technique CBR (case-based reasoning) allows to search into the knowledge base of the organization and to propose a solution similar to previous Cases.

A. Data processing

The question concerning the possibility of efficient data processing and the use of electronically collected information has led to a staggering number of different systems, technologies, methods and standards. In its function as the central nervous systems of a company, data warehousing and business intelligence represent a connection between data quantity and quality of information and are essential for making business decisions.

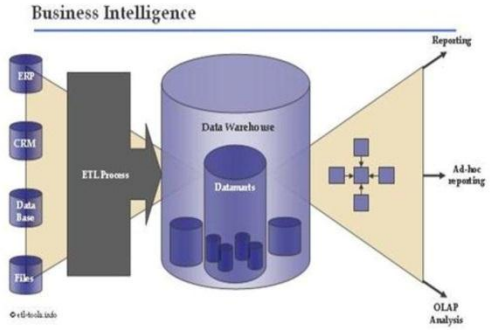


Fig. 8. Stage of transformation of the data for the elaboration of indicators

During the phase of calculation and elaboration of indicators, IS use tools of data processing to extract and transform the data. The function that we use in this type treatment is not generally reversible; it involves that the exact source of an indicator is not traceable 100%.

B. Cockpit

The cockpit is a place reserved only for specialists and decision-makers. It contains generally three things:

- list of indicators (real time situation),
- list of the possible actions (after a decision),
- communication means (to communicate the situation internally and externally)

C. Decision / objective

The analysis of a problem and the situation urges the team of piloting to make a decision and to define an objective. The interpretation of the existing according values of indicators and final state to be reached are both main factors which urge the leaders to take measures and actions to bring successfully the changes:

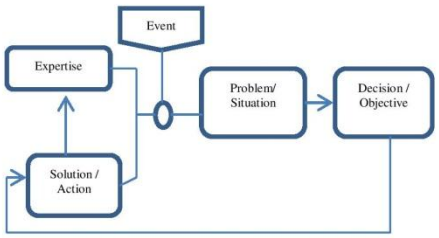


Fig. 9. Cycle of improvement of the knowledge of the system

The choice of the solutions and the actions depends generally on calculation of risk and the experience of the team of steering. This case involves a technical solution to save these experiences and to remember it when needed.

D. Modelling of knowledge by Object

1) Steering knowledge

A steering knowledge is a data structure that encapsulated the initial state (E_i) and final state (E_f)) of the system, its characteristics are the list of actions to bring the E_i state to E_f

An action defines one operation on the basic axis of the system according indicators selected:

- change on organization: structure, area, human resources
- change on process definition: process, application, profile,
- change on resource management: investment, charge reduction, etc. …

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 03, March-2017

2) Object

Modelling by object simplifies the presentation of the properties of the data:

- encapsulation of the properties and the methods (indicator and action)
- inheritance of the properties of the parents (tree of the values)
- instantiation (composition of other entity)

3) State Model

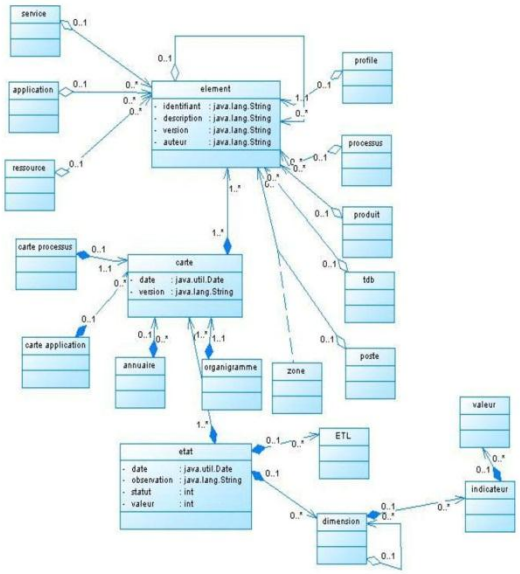


Fig. 10. State model

The state of the system contains essentially of an element, an association of element (map) and properties of the State

4) Decision model

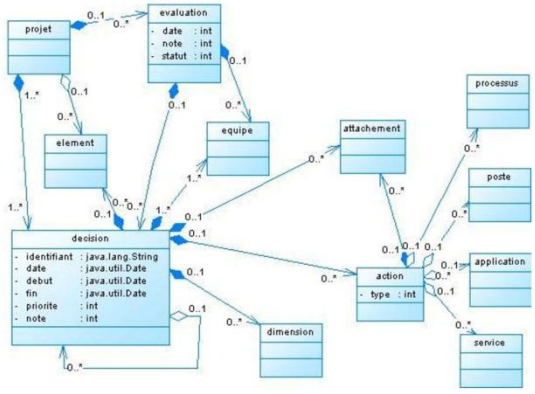


Fig. 11. Decision model

Decison combined three main entities :

- Project and Evaluation : to follow up the change
- Action : define a change
- Decision : contains properties

VI. CONCLUSION

The control of the management tool of information system requires a decomposition of the system. All informations produced by the system must be classified according to the list of analysis axis of the dashboard. With this configuration, the information system can adapt itself to the combination of the technology SOA, BPM, and CBR

The Case-Based Reasoning is a problem solving paradigm based on the reuse of past experiences to solve new problems. By using the model object, the decision-makers feeds a loop of processing to straighten the resources and the processes to reach the goal.

This loop is possible by memorizing the state of the system at the time of decision-making. If the actions are ended and that the decision is proved effective; We can consider the set of the information as state a knowledge which we can reuse every time the same Case appears.

REFERENCES

[1] Angot Hugues, "Information system of the company. Flows of information in the automatic information system,"November 2005.

[2] Robert REIX, "Information systems and management of organizations", Gestion vuibert, September 2011.

[3] Gilbert Paquette, Françoise Crevier, Claire Aubin "Method of modelling of the knowledge", LICEF, 1998.

[4] Karine Arnout, « From Patterns to Components», ETH Zurich. March 2004

[5] Souad Guessoum, Nadjette Dendani "Approach of Search using the Reasoning from case applied to the diagnosis of Obstructive Chronic Broncho Pneumopathie", Laboratory LRS, Annaba's university.

[6] Valentina CEAUSU, Sylvie DESPRES, "Using semantic resources for the elaboration and the remembrance of case", René Descartes university, 2005.

[7] Dima MUFTI-ALCHAWAFA, "Modelling and representation of the knowledge for the design of a decision-making system in an IT environment of learning in surgery", Joseph Fourier de Grenoble university, 2008.

A4.2 Pilotage de Système d'Information par raisonnement à partir de Cas

MADA-ETI, ISSN 2220-0673, Vol.2, 2016, www.madarevues.gov.mg

Pilotage de Système d'Information par raisonnement à partir de Cas

Razafindraibe T.A.A [1], Rastefano E.[2], Rabeherimanana L.I.[3]

Laboratoire Systèmes Embarqués – Instrumentation – Modélisation des Systèmes et Dispositifs Electroniques (SE-I-MSDE)

Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'innovation

Université d'Antananarivo
BP 1500, Ankatso - Antananarivo 101 - Madagascar

[1]andriampamonjy@gmail.com, [2]rastefano_el@yahoo.fr,[3]rabehlyliane@gmail.com

Résumé

Le raisonnement à partir de cas (RÀPC) est un paradigme de résolution deproblèmes fondé sur la réutilisation d'expériences passées pour résoudre de nouveauxproblèmes.

Le design pattern et le concept objetsontdes cas particulier d'implémentation de ce raisonnement. Il utilise le principe de similarité pour déduire la solution.

Le SI accumule des multiples de connaissances pendant son cycle de vie. Avec l'utilisation des patterns de gestion de processus et un moteur de gestion des indicateurs clés, le pilotage du SI d'information peut se fier à sa base de connaissance pour orienter ses décisions et d'optimiser le système.

Un système d'aide à la décisionpossèdequatre niveaux hiérarchiquesselon la maturité du système:

- classification
- diagnostic
- aide à la décision
- base de connaissance

La phase initiale de classification est une étape très importante vuqu'elle conditionne les autres étapes du RàPC. Il fautavoir uneindexation entre les méthodes de stockage de cas et la méthode remémoration pour assurer l'optimisation des résultats.

Mots-clés : processus, connaissance, adaptation, décision.

Abstract

The Case-Based Reasoning (CBR) is a problem solving paradigm based on the reuse of past experiences to solve new problems.

The design pattern and object concept are particular cases of implementation of this reasoning. It uses the principle of similarity to deduce the solution.

SI accumulates knowledge during its life cycle. With the use of business processes patterns and an engine management of key indicators, steering information SI can rely on its knowledge base to guide its decisions and to optimize the system.

A decision support system has four hierarchical levels depending on the maturity of the system:

- classification
- diagnostic
- support to the decision
- knowledge base

The initial classification phase is a very important step because it affects other stages of

the CBR. You have to have indexing between cases of storage methods and memory method to ensure optimization results.

Keywords: process, knowledge,adapter, decision.

1. Introduction

Le raisonnement qui consiste, face à un problème que l'on ne sait pas résoudre, à remémorerun problème déjà résolu et à en adapter la solution, est un raisonnement à partir de cas.

Ce type raisonnement repose sur trois opérations principales:

- la récupération
- l'adaptation
- la mémorisation

L'adaptation représentele cœur du processus RÀPC. De plus,l'adaptation est généralement considérée comme une opération de domaine de l'intelligence,qui est complexe et difficile à comprendre et à appréhender.

Dans cepapier, nous proposons une vue générique et opérationnelle d'adaptation quiest conçu pour être réutilisé dans le contexte de pilotage et de prise de décision:

2. Raisonnement à partir deCas

2.1. Définition

Un cas est un ensemble d'informations contextuellesenseignant une leçon et encapsule les éléments suivants:

- problème
- solution
- impact

Cette technique de raisonnement a été inspirée d'un modèle cognitif réel observé dans le comportement humain, qui consiste à s'appuyer sur les expériences du passé pour apporter solution à un nouveau problème. En effet, il a été démontré en psychologie que dans beaucoup de situations, l'homme commence à résoudre ses problèmes en se basant plus sur son expérience passée que sur sa connaissance du domaine en question.

2.2.Base de connaissance

Pour mener à bien un raisonnement, le système a besoin de disposer de connaissances. L'ensemble des expériences passées est représenté sous forme de base de cas, où le cas est un couple descripteur du problème et de sa solution.

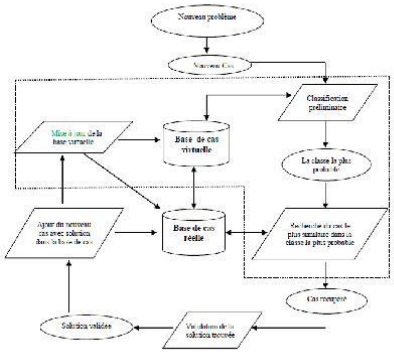


***Figure 1** : Architecture d'acquisition des cas*

L'évènement déclencheur de ce processus est la spécification d'un nouveau problème: en utilisant la base de connaissance existante, le système élabore un nouveau cas ou utilise de cas similaire:

2.3Les métriques de similarité

Cette étape permet de mesurer l'écart entre deux cas:

$$\boldsymbol{Similarité(ac,nc)} = \frac{\sum_{i=1}^{n} w_i * \boldsymbol{sim}(\boldsymbol{p_i(an)}, \boldsymbol{p_i(nc)})}{\sum_{i=1}^{n} w_i}$$

Où :
an : l'ancien cas stocké dans la base de cas,
nc : le nouveau cas à traiter,
wi : le poids attribué au paramètre p,

sim(pi(an), pi(nc)) : une autre fonction qui calcule la similarité entre les deux valeurs du paramètres pi de l'ancien cas et du nouveau cas respectivement. Elle est estimée par la formule suivante :

Pour les paramètres de type symbolique ou binaire, nous avons :

$$sim(p_i(\mathbf{an}), p_i(\mathbf{nc})) = \begin{cases} \mathbf{0} & \mathbf{si}\ p_i(\mathbf{an}) \neq p_i(\mathbf{nc}) \\ \mathbf{1} & \mathbf{si}\ p_i(\mathbf{an}) = p_i(\mathbf{nc}) \end{cases}$$

Pour les paramètres de type numérique, nous avons

$$\mathbf{sim}(p_i(\mathbf{an}), p_i(\mathbf{nc})) = \mathbf{1} - \mathbf{distance}(p_i(\mathbf{an}), p_i(\mathbf{nc}))$$

ou

$$distance(p_i(\mathbf{an}), p_i(\mathbf{nc})) = \frac{p_i(\mathbf{an}) - p_i(\mathbf{nc})}{\mathbf{écart\ max\ i}}$$

Où écart max représente l'écart entre les deux bornes supérieure et inférieure de l'intervalle des données du paramètre p_i.

3. Adaptation de Cas

L'étape d'adaptation permet de modifier les cas candidats sélectionnés lors del'étape de récupération, pour qu'ils répondent aux mieux à la description du cas cible.

Dans la plupart des systèmes, l'étape d'adaptation nécessite une intervention humainepour compléter une solution partielle ou tout simplement pour générer une solutionentièrement à partir des cas

On a deux types d'adaptation de cas :

- transformation
- dérivation

3.1. Transformation de Cas

La transformation consiste à modifier ou remplacer une partie de Cas pour atteindre la solution :

- Ré instanciation
  - Transposition du problème passé dans le contexte présent
  - Création d'une nouvelle instance de la classe du meilleur cas proposé
- Ajustement de paramètres
- Substitution à base de cas

3.2. Dérivation de Cas

Principe : Au lieu d'adapter la solution au problème, on adopte un processus menant du problèmevers la solution

Processus de dérivation = plan menant d'un étatinitial (le problème) vers un état final (la solution)par un ensemble d'étapes

4. Processus de remémoration organisationnelle

Par extension de la dérivation de cas RAPC, on peut supposer que le système d'aide à la décision est basé sur :

- la base de Cas de l'organisation
- processus de remémoration Cas
- processus d'adaptation du système

Le processus de remémoration organisationnelle se définit comme l'interaction de la remémoration de Cas (recherche de Cas similaire) et l'adaptation de Cas pour résoudre le Cas présent.

Ce processus doit être vu comme un acte de (re)construction (de recréation) de traces d'expériences, d'événements passés dans le but qu'elles fassent sens dans la situation présente.

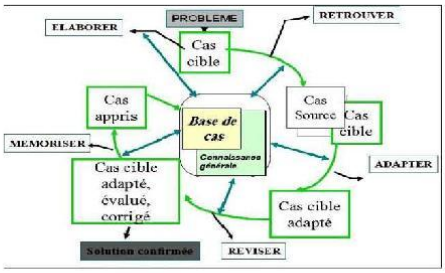


Figure 2 : Processus de remémoration

4.1. Pilotage de SI

L'objectif de toute entreprise (et plus généralement de toute organisation)est de garantir durablement son développement : l'entreprise doit s'assurerun avantage compétitif durable, une performance durable, une conformitéconstante aux obligations réglementaires du marché dans lequel elle évolue.

La nécessité d'aligner les processus de l'entreprise sur ses objectifsstratégiques conduit à la mise en place de « pilotes de processus ».

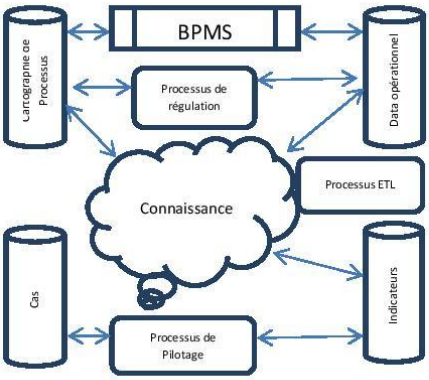


***Figure 3** : Base connaissance de l'organisation*

La conception, la régulation et l'adaptation de l'ensemble de processus du système d'information nécessite la base de connaissance de l'organisation.

Cette base nécessite un moyen technique pour le contenir afin qu'il puisse continuer son apprentissage pendant le cycle de vie du SI.

4.2. Apprentissage du système

Avec la technique RAPC, le système de base de connaissance peut mémoriser les Cas et augmenter la liste des leçons du système.

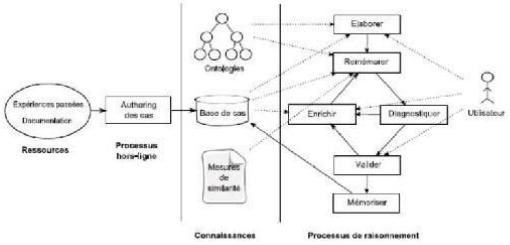


***Figure 4** : Processus de raisonnement*

Un processus de raisonnement permet au système de guider son apprentissage. La structure de base de Cas devrait être une arborescence (même modèle que le tableau de bord) pour pouvoir spécifier un niveau d'analyse lors de recherche et remémoration de Cas.

5. Conclusion

La maitrise du système d'information passe par sa décomposition. Elle consiste à concevoir une liste de processus pour chaque composant du système à l'aide son expérience et ses activités.

Le management du système en générale est divisé en deux :

- régulation : qui concerne la partie technique et métier
- pilotage : qui concerne la partie stratégie

L'utilisation de la base de connaissance pour la prise de décision sur ces deux types de management est un facteur important afin d'assurer l'avenir du système.

Références

[1]***Mill. A.,*** **«** Tutorial CBR : Etat de l'art de raisonnement à partir de cas. Plateforme »AFIA'99, Palaiseau, 1999.

[2]***Ivana Rasovska,*** «Contribution a une méthodologie de capitalisation desconnaissances basée sur le raisonnement a partir de cas : Application au diagnostic dans une plateformed'e-maintenance.», 2008

[3]***Souad Guessoum, NadjetteDendani***«Vers une Approche de Recherche utilisant le Raisonnement à partir de cas appliqué au diagnostic de la Broncho Pneumopathie Chronique Obstructive», 1Laboratoire LRS, Département d'Informatique, Université de Badji Mokhtar d'Annaba

[4]***Aurélien CODET DE BOISSE,*** «Aide à la décision exploitant de la connaissance générale etcontextuelle : application à la maintenance d'hélicoptères», Université de Toulouse, 2013

[5]***Amélie Cordier, Béatrice Fuchs,*** « Un assistant pour la conception et ledéveloppement des systèmes de RÀPC », Université Claude Bernard, 2005

[6]***Valentina CEAUSU, Sylvie DESPRES,*** «Utilisation de ressources sémantiques pourl'élaboration et la remémoration de cas»,Université René Descartes,2005

[7]***Club des Pilotes de Processus***, « Les Dossiers du Club – Pilotage de Processus etGouvernance Informatique », 2007

A4.3 Couplage BPM – SOA dans une architecture distribuée

MADA-ETI, ISSN 2220-0673, Vol.1, 2016, www.madarevues.gov.mg

Couplage BPM – SOA dans une architecture distribuée

Razafindraibe T.A.A [1], Rastefano E.[2], Rabeherimanana L.I.[3]

Laboratoire Systèmes Embarqués – Instrumentation – Modélisation des Systèmes et Dispositifs Electroniques (SE-I-MSDE)

Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'innovation

Ecole Supérieure Polytechnique d'Antananarivo
Université d'Antananarivo
BP 1500, Ankatso - Antananarivo 101 - Madagascar

[1]andriampamonjy@gmail.com, [2]rastefano_el@yahoo.fr,[3]rabehlyliane@gmail.com

Résumé

Dans une architecture distribuée, on n'a pas de contrôle centralisé des ressources (mesure), chaque implémentation offre son service tout en ignorant les contraintes des autres. Ce problème représente un risque énorme pour un système d'information car on a la tendance de dupliquer ou d'ignorer des informations et on ne maitrise plus leurs cohérences.

Une architecture orientée service est un cas particulier, une implémentation, d'une architecture distribuée. SOA crée des composants métier modulaires qui encapsulent le logique métier et données dans un service.

Le Business Process Management est un concept qui place au centre du système le processus métier. Il permet d'offrir au SOA un moyen de centraliser la compréhension du système et d'en établir les contrôles des données et des traitements.

Le couplage de ces deux architectures ne sous offre que des avantages :

- maitrise du traitement
- réutilisation
- maintenabilité

Mots-clés : processus, réutilisation, services, architecture.

Abstract

In a distributed architecture, there is no centralized control of resources (metrics); each implementation offers its service while ignoring other constraints. This problem represents a huge risk for an information system because it has the tendency to duplicate or ignore the information and no way to verify their consistencies.

A service-oriented architecture is a special case, an implementation of a distributed architecture. SOA creates modular business components that encapsulate the business logic and data in a service.

Business Process Management is a concept that uses process like the main frame in the system. It allows SOA a way to centralize the business of the system and to establish controls over data and processing.

The result of these two architectures offers advantages like:

- control of treatment
- reuse
- maintainability

Keywords: process, reuse, services, architecture.

1. Introduction

La conception d'une plateforme logicielle pour un système complexe nécessite des moyens et des stratégies spécifiques. Le concept génie logiciel offre une vue de haut niveau de la structure du système et de ses contraintes à l'aide d'une représentation abstraite. Le choix et la maitrise de l'architecture du système est un facteur clé pour orienter la conception et de piloter la réalisation et la spécification des ressources.

La présentation d'un système passe par la vue globale du système au sous-système et ainsi de suite. Cet aspect abstrait de l'architecture permet la combinaison et collaboration de plusieurs types d'architectures dans une même solution.

Dans le cadre de la mise en place d'une solution de gestion de système d'information, il est avantageux de combiner des types d'architectures afin faire face à la nature dynamique et exigent du système.

2. Discrétisation du métier

Le métier d'une organisation est une corrélation des activités pour atteindre un objectif clair. L'approche discrétisationpermet d'identifier et puis modélisercette structure d'activités.

Cette démarche consiste à décomposer le système et d'adopter de méthode d'identification des sous-systèmes. Cette méthode devrait fixerl'objectif de chaque sous-système qui va caractériser la nature des activités qui le compose.

2.1. Méthode SMART

Il est une méthodologie de formulation d'un objectif :

- Spécifique : Un objectif spécifique indique clairement ce qu'on souhaite obtenir.
- Mesurable : L'objectif doit contenir un critère chiffré que l'on veut mesurer
- Accessible :L'objectif est abordable, il faut éviter les illusions.
- Réaliste. Il faut être en mesure d'atteindre l'objectif depuis la situation actuelle, avec les ressources dont on dispose.
- Temporellement défini. Se fixer une date butoir, un temps maximal de réalisation.

2.2 Tâche

Une tâche est un élément basic d'une activité. Elle reflète la partie réalisation dans le moteur BPM.

Elle peut prendre plusieurs statuts pendant son cycle de vie :

- Crée
- En cours d'exécution
- En entente
- Assignée
- Terminée

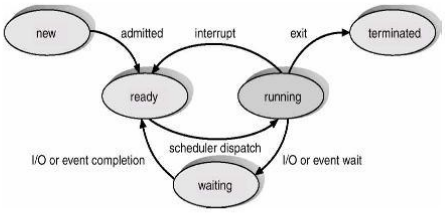


***Figure 1** : Evolution du statut d'une tâche*

3. BPM

La gestion de processus métier (BPM) offre une meilleure visibilité des activités d'une organisation.

Les relations inter- activités et acteur- activité sont définis au premier plan des digrammes de processus, elles permettent la traçabilité des traitements et la mesure des performances (acteur et processus).

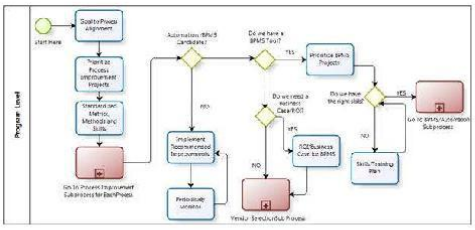


Figure 2 : *Business processengineering*

3.1. Simulation

La simulation de processus est l'un de fonctionnalité du BPM qui favorise la fiabilité et la robustesse du système.

Chaque élément du processus contient des paramètres qui peuvent être utilisés pour simuler la performance du processus face à des divers cas.

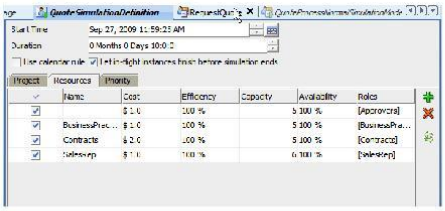


Figure 3 : *Paramétrage d'une simulation*

On peut saisir des valeurs et faire tourner le processus avec pour mesurer les indicateurs du système.

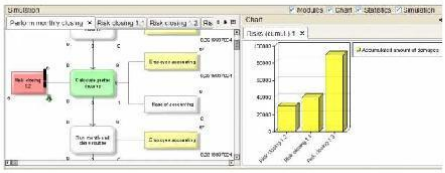


Figure 4 : *Simulation d'un processus*

La structure finale du processus peut être définie avec la validation du résultat des simulations.

3.2. Problème de détail de traitement

Malgré la puissance de l'outil BPM, elle n'est pas complète pour gérer les métiers d'une organisation. Un processus peut être défini comme une macro-activité à plusieurs étapes faisant appel à toutes sortes d'acteurs, de telle façon que chacune des tâches soit accomplie par un utilisateur du système ou par un sous-ensemble logiciel. Mais la réalisation d'une tâche réelle (service métier) est encore une procédure complexe qui n'est pas défini dans le processus.

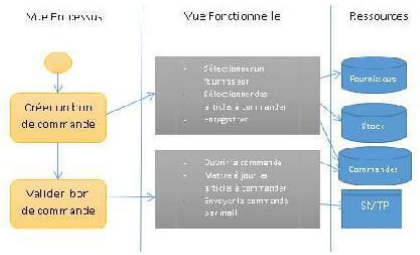


Figure 5 : *Niveau de définition de métier*

4. SOA

Cette architecture (Service orientedArchitecture) permet de structurer les traitements fonctionnels d'une organisation dans un environnement distribué et modulaire (indépendant entre eux).

Un service est une entité autonome qui s'échange avec une entité extérieure (Service, Application, BPM) via la règle de communication « request/response »

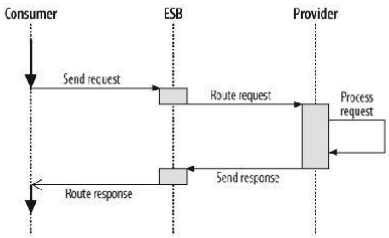


Figure 6 : Pattern d'échange de message « *Request/response* »

4.1. Catalogue de services

Le référentiel de service est un incontournable dans l'architecture SOA. Cette fonctionnalité

comprend deux grandes catégories de fonctionnalités :

- **Registre** qui vise à faciliter le fonctionnement de l'infrastructure de services :
  - Contrôle d'accès.
  - Correspondance entre les différents noms que peut porter chaque service dans les différents sous-systèmes.
  - Localisation et routage.
  - Uniformisation

- **Annuaire** qui vise à consolider la connaissance métier :
  - Etre la référence unique portant la connaissance des services de l'entreprise.
  - Porter la connaissance des formats d'échanges.
  - Distinguer les services qui nécessitent une réponse de ceux qui n'en attendent pas.
  - Identifier l'appartenance fonctionnelle de chaque flux.
  - Gérer les versions.

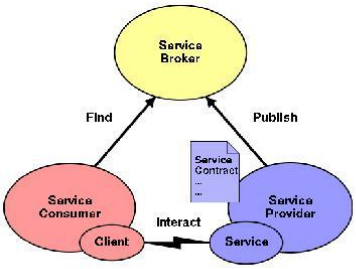


***Figure 7** : Référencement des services*

4.2Stateless / Stateful

La gestion de la communication entre le client et le service est gérer par le protocole « Request/Response » :

- Si on n'enregistre rien entre deux appels, le service est de natureStateless
- Si on enregistre une session, le service est de nature Stateful

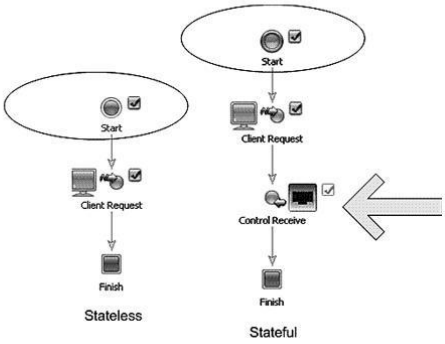


***Figure 8** : « Etat » de service*

Une session est une structure de données capable de stockerdes informations pour identifierle client et le statut de sa connexion.

5. Architecture Event bus

Cette architecture permet de relier des systèmes hétérogènes. On l'utilise pour assurer la gestion decommunicationinter-architectures.

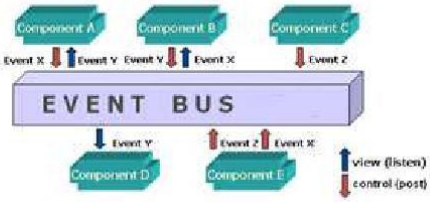


***Figure 8** : Fonctionnement de l'architecture par événement*

5.1. Fonctionnement

Cet outil d'intégration et de médiation fait le routage des messageséchangés entre les applications. Le message est toujours déclenché par un événement :

- Collecter les événements
- Dispatcher les événements
- Sauvegarder la session jusqu' à ce que le traitement soit fini

5.2. Approche événementielle

Un événement est un message instantané qui se produit par un changement de la valeur d'une propriété d'un objet vue par un observateur.

Si l'événement apparait, l'observateur diffuse l'information partout où cet est objet est utilisé.

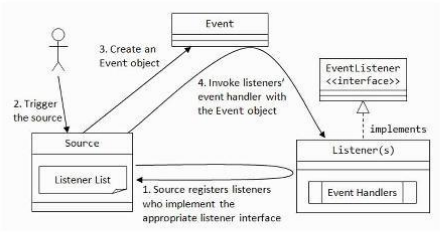


***Figure 9** : Implémentation d'un gestionnaire d'événement*

6. Couplage des architectures

La combinaison d'une architecture logicielle SOA avec un BPM nécessite l'implémentation d'une interface bus d'événements.

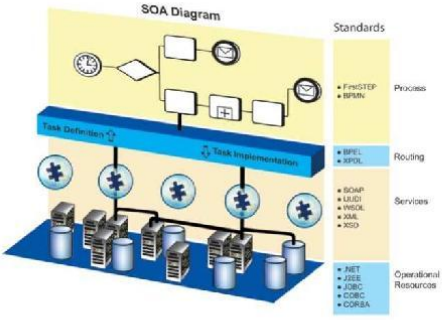


***Figure 10** : Couplage d'architectures*

La solution de l'assemblage de ces architectures donne naissance à une architecture par couche :

- Couche pilote : BPM
- Couche de routage
- Couche de transformation : services
- Couche ressources

Cette architecture a une forte cohésion avec le principe de fonctionnement d'un SI

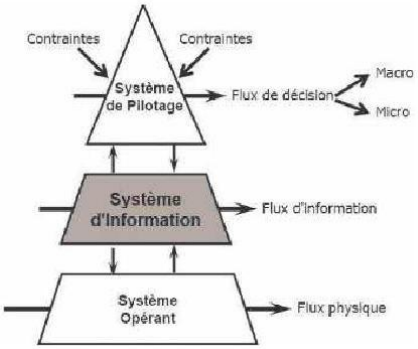


***Figure 11** : Système d'information*

La tâche principal du SI est de fournir un flux d'information qui d'une part, reflète le plus fidèlement possible le flux physique, et d'autre part fournit au système opérationnel les éléments nécessaires pour son fonctionnement quotidien et au système de pilotage les éléments nécessaires à une prise correcte de décision.

6. Conclusion

En associant BPM et architecture SOA, les entreprises améliorent leurs processus métier, au service de leur croissance. Sans la BPM, on ne fait que la moitié du chemin, la démarche se limitant alors aux seuls aspects techniques.

La BPM permet de faire l'autre moitié du chemin : utiles au pilotage des activités opérationnelles de l'entreprise, les outils de BPM permettent en effet aux managers de s'affranchir des considérations purement techniques et de déployer des processus métier à la fois plus simples, plus souples et plus efficaces.

Les logiciels de BPM sont les premières applications à tenir vraiment compte les besoins des dirigeants.

Mais il nous reste encore des problèmes à résoudre tant à l'intégration de cette solution dans le système d'information que la gestion

des ressources transversales pour l'ensemble du système.

En utilisant à fond notre expérience sur l'ingénierie logicielle, il est possible de trouver une solution optimale avec le design pattern pour améliorer la gestion des ressources d'une organisation et de piloter son système d'information.

Références

Manuel Eveno,Christophe Heubès, «Comprendre et savoir utiliser un ESB dans une SOA ». Xebia, 2007.

Vincent Lepot, «Comment l'architecture événementielle révolutionne la communication dans le SI», 2014

Marc Fiammante«Best Practices for Business Process Management and SOA Agility», IBM Press/Pearson, 2010

Judith Hurwitz, Robin Bloor, Marcia Kaufman, «Service Oriented Architecture (SOA) For Dummies », Wyley, 2009

Fred A. Cummins, « Building the Agile Enterprise: With SOA, BPM and MBM», 2010

Bernard Debauche, Jean-François David, « Pilotage par les processuset gouvernance informatique»,Lub,2007

A4.4 Design Pattern to Component

MADA-ETI, ISSN 2220-0673, Vol.2, 2015, www.madarevues.gov.mg

Design Pattern to Component

Razafindraibe T.A.A. [1]***, Rastefano E.***[2]***, Rabeherimanana L.I.***[3]

Laboratoire Systèmes Embarqués – Instrumentation – Modélisation des Systèmes et Dispositifs Electroniques (SE-I-MSDE)

Ecole Doctorale en Sciences et Techniques de l'Ingénierie et de l'innovation (ED-STII)

Université d'Antananarivo
BP 1500, Ankatso - Antananarivo 101 - Madagascar

[1]*andriampamonjy@gmail.com,* [2]*rastefano_el@yahoo.fr,*[3]*rabehlyliane@gmail.com*

Résumé

La technologie sur les systèmes d'informations utilise de plus en plus un système virtuel pour stocker ses services et ses données (Cloud).

La méthode de conception de logicielle devrait suivre cette évolution et on doit passer de l'étude de l'existant habituel par l'étude de réutilisation de services. Ce concept nécessite une nouvelle plateforme et une nouvelle méthode pour produire un outil selon le besoin de client.

Cette mondialisation de traitement par service permet de redéfinir notre concept en matière de programmation et de génie logiciel. L'objectif principal est de fournir des outils plus robuste et plus rentable dans un environnement préalablement remplie des services.

Mais un tel environnement nécessite d'autre support technologique tel que la communication et la virtualisation.

Mots-clés : Design Pattern, réutilisation, services, noyau, processus.

Abstract

Technology on information systems increasingly uses a virtual system to store its data and services (Cloud).

The software design method should follow this development and one must go from the study of the regular current software through the study of reusing services. This concept requires a new platform and new method to produce a tool according to the needs of customers.

This globalization of processing by service allows us to redefine our concept in terms of programming and software engineering. The main objective is to provide more robust and efficient tools in a prefilled services environment.

But such an environment requires different technological support such as communication and virtualization.

Keywords: Design Pattern, reuse, services, kernel, process.

1. Introduction

Le matériel informatique et logiciel font une paire inséparable. Chaque avancé technologique sur un terminal ou un serveur entraine un boom de nouvelle version sur la partie software.

Le cycle de vie d'un logiciel est devenu de plus en plus fine d'où la nécessité d'une

nouvelle technique sur la conception et réalisation des applications.

La réutilisation semble évident la solution idéale mais elle implique des applications génériques. Il faut donc mettre en place unmodèle bâtie sur le concept de la réutilisation (design pattern) mais ouvert pour que les développeurs puissent l'adapter aux besoins finaux.

Généralement un traitement générique nécessite des éléments génériques. Cette étude se limite sur le domaine de système informatique qui est un système régie par de processus. En effet un processus cautionne la réutilisation et le traitement générique.

2. Approche

Pour réduire la complexité du système informatique, il faut le subdiviser en plusieurs sous-systèmes. Cette démarche consiste à identifier et à modéliser des composants logiciels susceptibles d'être utilisés pour construire un système portable et réutilisables. Pour y parvenir, il faut utiliser des solutions déjà prouvées : Pattern.

2.1. Design Pattern

Un design Pattern définie d'une manière abstraite la solution à un problème générique. Cette solution est indépendante de son implémentation.

On a trois grandes familles de Pattern :

- *Créateur* : initialisation et configuration
- *Structuraux* : séparation des préoccupations et abstraction pour la réutilisation
- *comportementaux*: combiner interactions et structure

La base de toute forme de réutilisation (côté développement) dans cette étude se forge à partir des design Patterns.

2.2 Processus au sein d'un système d'information

Le système d'information (S.I.) se définit comme un « Système utilisateur machine intégré qui produit de l'information pour assister les êtres humains dans les fonctions d'exécution, de gestion et de prise de décision »

Le SI permet donc de :

- traiter les informations selon des processus bien définis
- fournir des indicateurs pertinents pour l'organe décisionnel du système

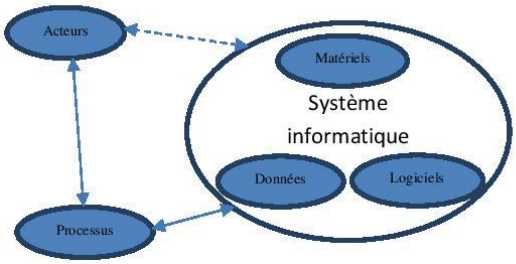


***Figure 1** : Système d'information*

Un processus est définie comme un ensemble d'activités corrélées ou interactives qui transforme des éléments d'entrée en éléments de sortie.Un processus métier est une transformation qui produit une valeur ajoutée tangible à partir d'une sollicitation initiale. Il est divisé en activités (ou tâches), réalisées par des acteurs (humains ou automatiques) avec l'aide de moyens adaptés, qui contribuent chacune à l'obtention du résultat escompté. Le caractère métier du processus s'exprime par la nature du résultat, qui doit avoir un sens pour un client (client au sens large, interne ou externe) et mesurable.

Chaque nœud du processus peut être lié à une application spécifique pour réaliser la tâche qui lui correspond.

La réutilisation au niveau de processus peut se présenter sous deux formes :

- Dans le temps : la définitiondu processus se fait une seule fois mais la réalisation de traitement n'est limitée que par l'activité de l'entreprise
- Par la nature du traitement : un processus existant peut être appelé par un autre processus

3. Etapes

Cette étude comporte trois étapes de modélisation qui consiste à faire évoluer le dégrée de réutilisation des objets produits à chaque niveau:

- Conception des composants à partir de design Pattern
- Conception par service
- Conception par processus

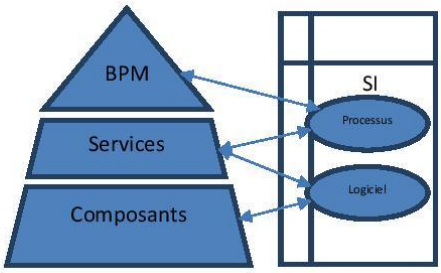


***Figure 2** : Niveau de réutilisation dans le système d'information*

On commence par construire des composants et des Framework à partir des design Pattern : Cette étape permet d'avoir les avantages suivants:

- réduction de temps de développement
- réduction de tempsde maintenance
- augmentation de la qualité et la fiabilité du logiciel

Par la suite, on définit une plateforme logicielle à l'aide des composants ainsi crées. Cette centralisation de traitement permet de gérer :

- les services
- les processus
- les flux de traitement
- les applications tierces

Chaque entité de ce noyau est indépendante entre elles, et la conception devrait suivre les règles définies par les Pattern.

La dernière partie de cette étude sera consacrée à la mise en place des services des bases qui permettent :

- la sécuritéet l'accès
- la gestion des ressources
- l'avancement des flux

4. Réalisation : Componentization

Cet étape de modélisation permet de :

- Identifier une entité logicielle réutilisable
- Donner plus de degré de portabilité aux applications
- Accélérer la réalisation d'une application
- Diminuer le coût de la maintenance

Mais la plus importante caractéristique offerte par la componentization est la possibilité de relier deux instances d'applicationavec la technique de sérialisation : deux applications différentes peuvent traiter et échanger des mêmes composants de même nature.

La réutilisation des applications est basée sur ce principe d'échange des composants logiciels couplée par le partage des données de gestion.

4.1. Composant de base

Ces composants constituent le Framework principale de l'environnement à modéliser. Ils dictent les règles et les protocoles pour écrire une application et pour gérer ces ressources.

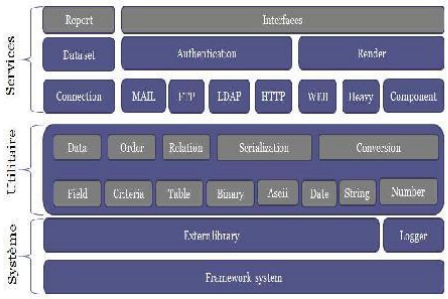


***Figure 3** : Structure en couche des composants*

Ce Framework est structuré en trois couches :

- Système : définit le pont entre le système et l'environnement externe.
- Utilitaire : classe de traitement basic et classe des données natives
- Service : définit des interfaces entrel'application finale et les serveurs : serveur des données, serveur de transfert de données, serveur d'authentification, etc…

4.1.1 Composant Système

Il définit l'ensemble de classe et des librairies qui permettent d'assurer *l'interface* de l'application avec le système d'exploitation.

Il est constitué essentiellement des couches fournis par le langage de programmation. Pour adapter les traitements aux composants utilitaires et services, on a modélisé les traitements suivants :

- lancement de tâche répétitive et planifié (Scheduler)
- gestion de communication évènementiel inter composant (Event)
- accès aux ressources et aux bases de données
- enregistrement des logs (traceability)

a. Scheduler

Cette modèle permet de gérer une liste des tâches programmées dans le temps de manière simple ou périodique.

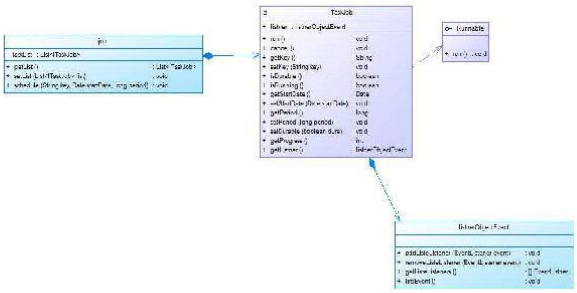


***Figure 4** : Modèle Scheduler*

b. Resource

Une ressource est définie comme une entité capable de stocker et de restituer des données informatique. Cette entité est caractérisée généralement par son type et son mode d'accès. Un composant ressource constitue une couche de filtre qui génère les autorisations pour accéder à des ressources via une application. On a deux types de filtre :

- Filtre native : c'est la ressource elle-même qui filtre les utilisateurs
- Filtre artificielle : c'est un ensemble de code qui gère l'accès

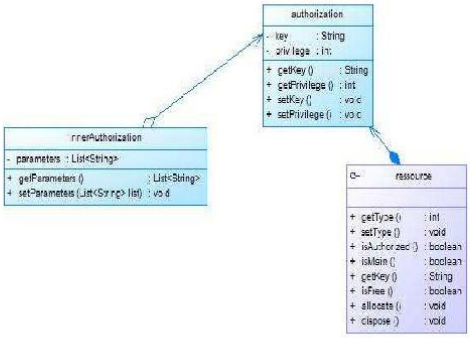


***Figure 5** : Modèle Authorizer*

c. Logger

L'administration et la maintenance d'un logiciel nécessitent le traçagede ces activités

lors de l'utilisation. Ces informations sont indispensables pour contrôler le bon fonctionnement d'un outil.

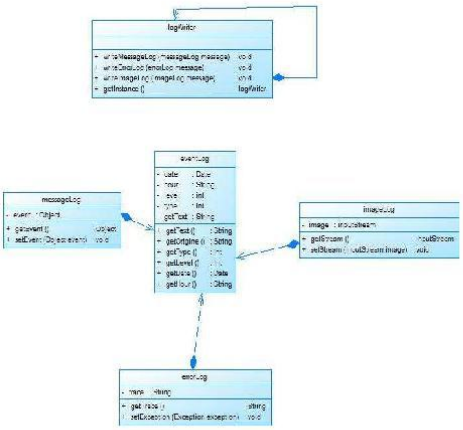


***Figure 6** : Modèle Logger*

4.1.2. Composant utilitaire

Les développeurs ont besoin des boites à outil pour bâtir une application. Ces outils devraient constituer des classes qui manipulent le conteneur, la conversion et le formatage des données.

a. Conteneur d'enregistrement :

Cet ensemble de classe a pour responsabilité de contenir les informations échangéesavec une base de données :

- champ
- table
- relation table
- enregistrement
- clause where
- clause order

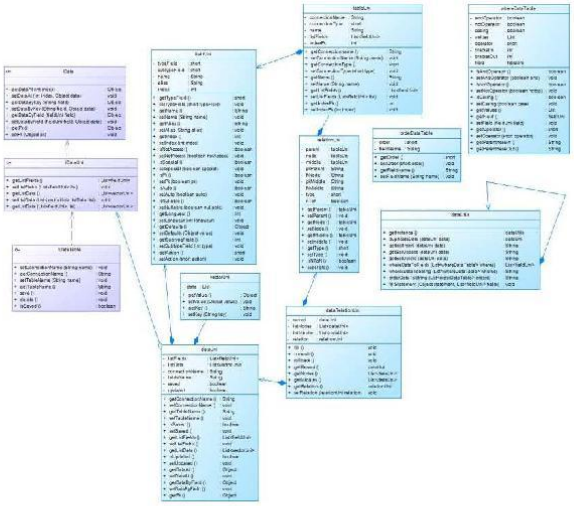


***Figure 7** : Modèle Data Container*

b. Conteneur de flux

Les données issues d'un fichier ou d'un périphérique sont traitées sous forme de flux. On distingue deux types de flux :

- binaire
- caractère

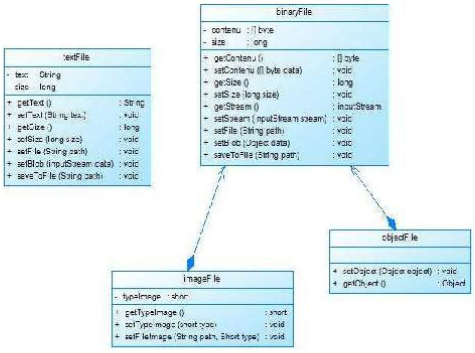


***Figure 8** : Modèle Stream*

c. Manipulation des objets natifs

Chaque langage de programmation a sa propre méthode pour manipuler les données natives. Les composants suivants donnent des moyens d'adaptation pour généraliser les traitements :

- date
- tableau
- Chaine de caractère
- Chiffre
- byte
- sérialisation

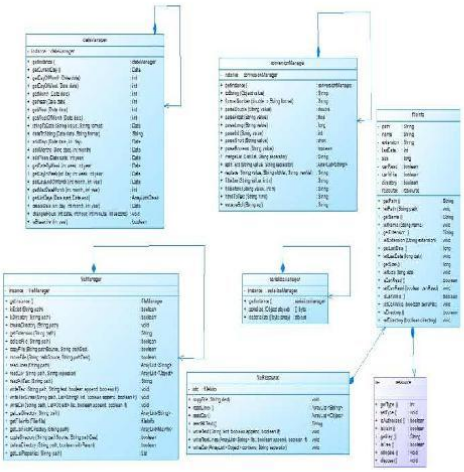


Figure 9 *: Modèle Utils*

4.1.3. Composant service

Cet ensemble de classe offre des fonctionnalités qui permettent de manipuler, transférer et présenter des données.

a. Service base de données

Il permet de gérer les transactions et les traitements sur des enregistrements.

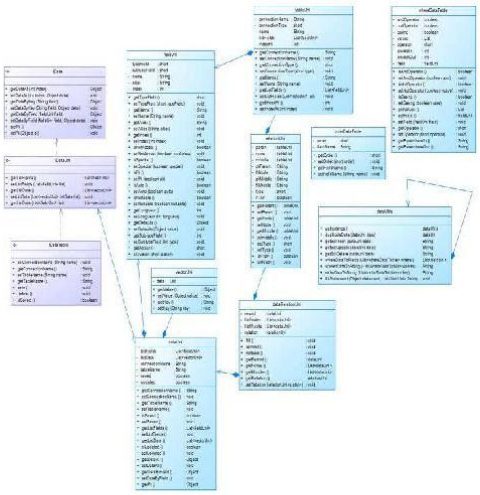


Figure 10 *: Modèle Datasource*

Chaque type de base de données doit implémenter ce modèle pour que le système puisse l'adapter aux couches inférieures.

b. Service client serveur

Cette entité offre la possibilité d'utiliser d'autre serveur pour transiter des données :

- Mail
- FTP
- LDAP
- Serveur d'application

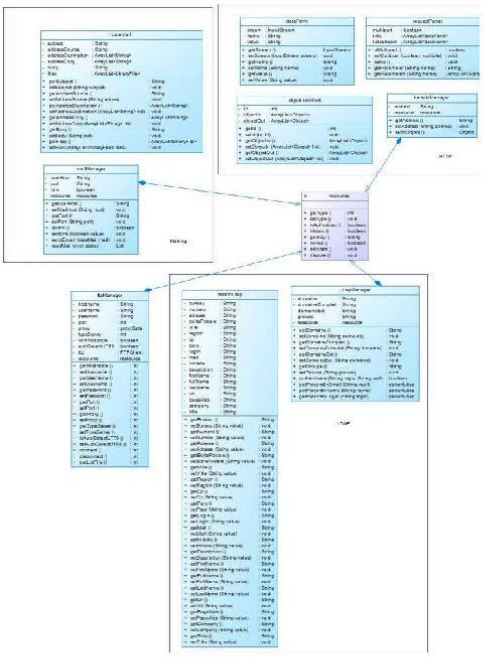


***Figure 11** : Modèle Network*

c. Service de présentation et d'affichage

Un composant graphique est un élément qui regroupent à la fois technique et art. L'ergonomie et l'organisation sont les critères principaux du choix d'un composant.

On distingue trois groupes de composant graphique :

- Lié aux données : affichage et/ou saisie des données
- Conteneur : regroupe les composants des données
- Actionnaire : bouton ou menu d'action

La gestion d'un composant graphique varie selon la nature de l'application :

- application desktop (lourde)
- application web (léger)

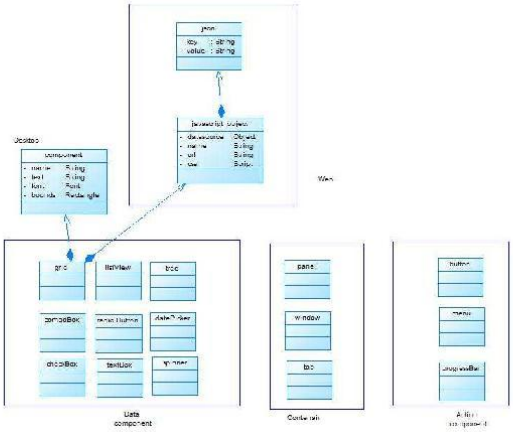


***Figure 12** : Modèle Component*

2.4.4. Service reporting

Ce service de présentation des données est modélisé à l'extérieur et indépendamment des autres modèles car il constitue un pont entre le système avec d'autres applications de présentations des données (Microsoft Office, Open Office, Adobe Pdf, etc...). Son implémentation dépend entièrement de la structure de l'application hôte.

5. Conclusion

La complexité croissante des SI et leur incessante évolution rend leur développement plus laborieux, plus coûteux et moins fiable.

L'approche par processus couplé à une plateforme logicielle ouverte permet d'apporter une nouvelle solution àce problème. La modélisation de ce système a besoin d'une base solide réutilisable. En adoptant les définitions et les solutions décrites par les designs patterns, on arrive à définir l'architecture de Framework de basequi sera utilisé pour construire un PBM.

5. Perspective : modélisation du moteur de Workflow et le noyau du système

5.1 Workflow

Un système Workflow est divisé en deux parties :

- La routine qui est capable de stocker les versions de processus et de faire avancer les instances de ces processus

- L'assistant qui permet de tracer et modéliser des processus

Dans un Workflow, on définit trois entités fondamentales :

- Les types de nœuds possibles

- La liste des actions possible sur un nœud

- Les utilisateurs

5.2 Noyau

L'objectif principal de cette étude est la réutilisation des applications. La solution qu'on propose est la réutilisation à travers d'un Workflow et des composants logiciel.

La modélisation d'un tel système nécessite la maitrise de tous éléments qui évoluent dans un système d'information.

Références

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, «Design Patterns: Elements of Reusable Object-Oriented Software». Addison-Wesley, 1995.

Angot Hugues, « Système d'information de l'entreprise. Des flux d'information au système d'information automatique », 2006

Mohamed Jaouad El Qasmi et Abdelaziz Kriouile « Vers une nouvelle relation : stratégie/système d'information », Revue de l'innovation dans le secteur public, volume 8 (4), 2003

Robert REIX, « SYSTEMES D'INFORMATION ET MANAGEMENT DES ORGANISATIONS »

KarineArnout, « From Patterns to Components», thèse de Doctoraten sciences, Swiss Federal Institute of Technology Zurich

En admettant que la maitrise de l'outil de gestion de système d'information passe par sa décomposition en processus et que chaque information produite dans le SI est mesurable selon des axes finis; le pilotage de ce système peut être assimilé à un boucle ouvert de redressement pour atteindre et stabiliser un objectif(état) et pour surpasser les différentes épreuves(risques et changements). Cette boucle est possible en mémorisant l'état du système au moment de prise de décision. Si les actions et les changements sont terminées et que la décision prise est prouvée efficace; on peut considérer l'ensemble des informations relative au pilotage comme état une connaissance que l'on peut réutiliser à chaque fois que le Cas apparaisse.

Tovohery Alain Andriampamonjy Razafindraibe, Ingénieur en électronique, études de modélisation des systèmes autonomes à l'Université d'Antananarivo, responsable de développement d'application de Dynatec Madagascar (projet Ambatovy).

978-3-639-62019-1

ÉDITIONS UNIVERSITAIRES EUROPÉENNES

Tovohery Andriampamonjy Alain Razafindraibe

Du Pattern au Système distribué

De l'Information à la Connaissance

BIBLIOGRAPHIE

[1.1] Robert REIX, « *Systèmes d'information et management des organisations* », CREGO, 2004.

[1.2] Angot Hugues, « *Système d'information de l'entreprise. Des flux d'information au système d'information automatique* », 2006.

[1.3] Mohamed Jaouad El Qasmi, Abdelaziz Kriouile, « *Vers une nouvelle relation : stratégie/système d'information* », Revue de l'innovation dans le secteur public, volume 8 (4), 2003.

[1.4] Bernard ESPINASSE, « *Introduction aux décisions et processus décisionnels* ». Université d'Aix-Marseille, 2009.

[1.5] Yves Donato, « *Indicateurs clés de performance des Technologies de l'information* », Faculté des sciences de Sherbrooke Québec, Canada, février 2013.

[1.6] Matthieu LAFARE, « *Business Information* », HEC Paris, 2006.

[2.1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, « *Design Patterns: Elements of Reusable Object-Oriented Software* ». Addison-Wesley, 1995.

[2.2] Steven John Metsker, William C. Wake « *Design patterns in Java - Les 23 modèles de conception fondamentaux* », Pearson Education France, 2006.

[2.3] Pierre-Alain Muller, « *Représentation des patterns avec UML* », ENSISA.

[2.4] Jean-Marc Jézéquel. « *Patrons de conception* », HAL, 2006.

[2.5] Thierry Lecomte, « *Patrons de conception prouvés* », Université Henri Poincaré Nancy 1, 2010.

[2.6] Alexander C., Ishikawa S., Silverstein M., Jacobson M., FiskdahlKing I., Angel S. « *A Pattern Language* », Oxford University Press, 1977.

[2.7] O. Boissier, G. Picard, « *Design Patterns* », Mines Saint-Etienne, 2009

[3.1] Mortier Mélanie, «*Patrons d'architecture des Systèmes d'Information* », ENSG, 2008

[3.2] BRANDON ADAM GUMP, « *Automated transforms of Software models: a design pattern approach* », Wright State University, 2009.

[3.3] Jeremiah Y. Dangler, « *Categorization of Secure Design Patterns* », East Tennessee State University, Mey 2013.

[3.4] Jean - Philippe Retaillé, « *Refactoring des applications Java/J2EE* », Groupe Eyrolles, 2005.

[3.5] Philippe Collet, « *Option Architecture logicielle* », SHOPIA ANTIPOLIS, Université Nice, 2010.

[3.6] Andreas Meier, « *Le CRM analytique Les outils d'analyse OLAP et le Data Mining* », Université de Fribourg, 2008.

[3.7] Ouafa Hachani, « *Patrons de conception à base d'aspects pour l'ingénierie des systèmes d'information par réutilisation* », laboratoire LSR-IMAG - Université Joseph Fourier – Grenoble I, 2006.

[4.1] ZITOUNI Abdelhafid, « *Utilisation des design Patterns et des méthodes formelles dans le développement des systèmes d'information* », Université Mentouri Constantine, 2008.

[4.2] Hervé Albin-Amiot, Pierre Cointe, Yann-Gaël Guéhéneuc, « *Un méta-modèle pour coupler application et détection des design patterns* », École des Mines de Nantes, France.

[4.3] Karine Marguerite Alice ARNOUT, « *From Patterns to Components* », Swiss Federal Institute of Technology Zurich, 2004.

[4.4] Arnaud SIMON, « *Implémentation d'un éditeur BPMN au sein d'un outil de métamodélisation* », Université de Namur, 2014.

[4.5] Yoran Maxim Bosman, « *Incorporating Functional Design Patterns In Software Development* », University of Twente - Maria Laura Ponisio - Jos van Hillegersberg, 2007.

[4.6] Lilia GZARA, « *Les Patterns pour l'Ingénierie des Systèmes d'Information Produit* », Institut National Polytechnique de Grenoble, 2000.

[4.7] Adam C. Jensen, « *Using Evolutionary Computation to Automatically Refactor Software Designs to Include Design Patterns* », Michigan State University, 2010.

[4.8] Gaël Blondelle, « *l'ESB JBI au cœur de l'initiative SOA*», CTO EBM WebSourcing, 2007.

[4.9] Xebia IT Architect, « *Comprendre et savoir utiliser un ESB dans une SOA* », 2007.

[4.10] Egov Innovation Center, « *Veille Technologique : Outils du BPMN 2.0* », HEIA-FR, 2016.

[4.11] Object Management Group Inc, « *Business Process Model and Notation*», OMG, 2011.

[5.1] Fadi Badra, « *Extraction de connaissances d'adaptation en raisonnement à partir de cas* », Ecole doctorale IAEM Lorraine, 2009.

[5.2] Diala Dhouib, « *Aide multicritère au pilotage d'un processus basée sur le raisonnement à partir de cas* », Université de Paris 8 Vincennes-Saint-Denis, 2009.

[5.3] Gilbert Paquette, Françoise Crevier, Claire Aubin, « *Technique de modélisation des connaissances* », Centre de recherche LICEF, 1998.

[5.4] Gilbert PAQUETTE, « *Modélisation des connaissances et des compétences* », BF Conseil et Formation, 2008.

[5.6] Philippe Fournier-Viger, « *Un modèle de représentation des connaissances à trois niveaux de sémantique pour les systèmes tutoriels intelligents* », 2005.

[5.7] AAMODT A. & PLAZA E. « *Case-based reasoning: Foundational issues, methodological variations, and system approaches.* », AICOM, 1994.

[5.8] Christel DARTIGUES, « *Ingénierie Concourante* », UFR d'Informatique Université Claude Bernard Lyon 1, 2001.

[5.9] Ricco RAKOTOMALALA, « *Introduction au data science du data mining au big data* », Université Lumière Lyon 2.

[5.10] Alain Fernandez, « *GIMSI – Le projet Business Intelligence clés en main* » Groupe Eyrolles, 2014.

[5.11] André CHARDONNET – Dominique THIBAUDON, « *Le guide du PDCA de Deming* », Éditions d'Organisation, 2003.

[6.1] Tim Stojanovic, « *Guide pour la mise en œuvre d'un système d'information local littoral* », CorePoint, 2007.

[6.2] Coopers and Lyband « *Managing Information Systems Risk: results of an international survey of large organisations* », London, 1996.

[6.3] Lionel Di Maggio, « *Management par les processus – les éléments structurants* », Université Nice - Sophia Antipolis, 2011.

[6.4] Henri CHELLI, « *Urbaniser l'entreprise et son système d'information, Entreprendre informatique* », 2003.

[6.5] Anis FERCHICHI, « *Contribution à l'intégration des processus métier : application à la mise en place d'une référentielle qualité multi-vues* », Ecole centrale de Lille, 2008.

Tovohery Alain Andriampamonjy RAZAFINDRAIBE

Logt 61 résidence Tsararivotra près Cité Vohitsara - Toamasina
Téléphone : ***+261 33 37 651 64***
Adresse de messagerie : **andriampamonjy@gmail.com**
Compagnie : ***Dynatec Madagascar***
Position : ***Responsable de développement des outils internes (IT)***

Directeur de thèse

Professeur RASTEFANO Elisée - ESPA

Docteur RABEHERIMANANA Lyliane Irène - ESPA

Titre :

Du Design Pattern aux applications Entreprises

Nombre de page :

226

Nombres de tableaux :

88

Nombres de figures :

98

Mots clés :

Design pattern, processus, système d'information, business intelligence, Framework, métier, indicateur de performance, bus événement, architecture, Cas, workflow, service.

RESUME

Un *Design Pattern* par son définition permet de modéliser une solution à un problème afin de réutiliser cette solution dans des divers cas sans jamais le faire deux fois de la même manière. Les *Design Patterns* sont des solutions réutilisables au niveau de conception mais ils ne sont pas totalement réutilisables au point de vue code. Les programmeurs doivent les réimplanter à chaque nouveau développement. Même en utilisant des *API* qui contiennent une partie ou la totalité de l'implantation de la solution modélisée, la réutilisation s'arrête au niveau des composants et chaque application possède son propre manière d'intégration de ces composants.

L'objectif de cette thèse était d'apporter des nouveaux **degrés de réutilisabilité** : transformer les patrons de conception en une application réutilisable que les programmeurs peuvent utiliser et réutiliser sans avoir à réécrire les mêmes fonctionnalités à chaque intégration d'une nouvelle solution.

Certes, chaque domaine d'activité possède ses propres besoins en matière de logiciel et même dans une activité spécifique, il est pratiquement impossible de couvrir toutes les fonctionnalités afin qu'on puisse apporter de solution générique. Pour cette thèse, nous nous intéressons sur l'étude du domaine de « **Système d'Information ».** Le système d'information se construit autour de processus « métier » et ses interactions, il permet de gérer les ressources d'une entreprise avec un moyen de communication bien défini pour partager et faire évoluer les informations.

Cette thèse examine les différents problèmes qu'on se rencontre fréquemment lors de la mise en place d'un système d'information afin de construire des *Patterns*. La réalisation et la programmation utilisent l'environnement Java car ce langage fait partie des langages qui permettent la conversion de *Pattern* en composant.

ABSTRACT

By its definition, a Design Pattern allows to model a solution to a problem in order to re-use that solution in diverse cases without using it the same way again. Design Patterns are solutions that can be reused at the conception level but which are not fully usable at a programming level. Programmers have to relocate them for each new programming. The reuse of the Design pattern remains at the components level and each application possesses each own way of components integration even though APIs, which contain a part or the totality of the implementation of the modeled solution, are used.

The objective of this thesis was to bring new **degrees of reusability**: transform design patterns to a reusable application that programmers could use and reuse without having to rewrite the same functionalities for a new solution integration.

Certainly, each activity field has each own software needs and even within a specific activity, it is quasi impossible to cover all the functionalities in order to provide a generic solution. For this thesis, we are interested in the field study of « **Information System** ». The information system is built around the process « business » and its interactions. It allows the management of an enterprise resource with a well-defined communication means for sharing and evolving information.

This thesis looks at the different issues frequently encountered during the implementation of an information system in order to build Patterns. The realization and the programming use the Java environment because this language is among the languages that allow the conversion of Pattern into components.