République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Système Informatique

Thème :

Une Approche pour l'interrogation des données des processus métiers

Encadré Par :

Dr.Khebizi Ali

Présenté par :

Hamici Fatima

Juillet 2019

Table des matières

Table des figures

Liste des tableaux

Liste des symboles

- **BAM** :*BusinessActivityMonitoring.*(Supervision des activités des processus)
- **BDD** :*BasedeDonnes.*
- **BPEL** :*BusinessProcessExecutionLanguage.*(Langage d'exécution des processus métier)
- **BPM** :*BusinessProcessManagement* (Gestion de processus).
- **BPMS** :*BusinessProcessManagementSystme* (Gestion de processus).
- **BPML** :*BusinessProcessModelingLanguage.*(Langage de modélisation de processus métier)
- **BPMI** :*BusinessProcessManagementInitiative.*(Initiative de gestion des processus métier)
- **BPMN** :*BusinessProcessModelingNotation.*(notation de gestion des processus métier)
- **EXPDL** :*EnablesaProcessDefinitionEcosystem.*(Permet un écosystème de définition de processus)
- **IBM** :*InternationalBusinessMachines.*
- **MOOC** :*massiveopenonlinecourse.*(cours en ligne ouvert à tous)
- **NoSQL** :*NotOnlySQL.*(Pas seulement *SQL*).
- **OLAP** :*OnlineAnalyticalProcessing.*(Processus analytique en ligne)
- **PAIS** :*ProcessAwareInformationSystems.*(systèmes d'information sensibles aux processus)
- **PM** :*ProcessusMtier.*
- **RDF** :*ResourceDescriptionFramework.*(Cadre de description de ressource)
- **SGBD** :*SystmedeGestiondeBasesdeDonnes.*
- **SGML** :*StandardGeneralizedMarkupLanguage.*(langage de balisage généralisé standard)
- **SI** :*SystmeInformatique.*
- **SPARQL** :*StandardProtocolandRDFQueryLanquage*).(Protocole standard et langage de requête *RDF*)
- **SQL** :*StructuredQueryLangage.*(langage de requête structuré)
- **XML** :*eXtensibleMarkupLanguage.*(langage de balisage extensible)
- **YAWL** :*YetAnotherWorkflowLanguage.*(Encore un autre langage de flux de travail)

Dédicace

Je dédie ce modeste travail
A mes parents qui m'ont soutenu et encouragé durant mes années d'études.
A mes frères, qui ont partagé avec moi tout les moments d'émotion lors de la réalisation de ce travail. Ils m'ont chaleureusement supporté et encouragé tout au long de mon parcours.

A mes proches amies, qui m'ont donné de l'amour et de la vivacité.

A tous ceux que j'aime

REMERCIEMENTS

En préambule à ce mémoire nous remercions ALLAH qui nous aide et nous donne la patience et le courage durant ces langues années d'études.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Ces remerciements vont tout d'abord au corps enseignant et administratif de notre Faculté, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Nous tenant à remercier sincèrement Monsieur **Khebizi Ali**, qui en tant que Directeur de mémoire, il se montre toujours à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'ils a bien voulu nous consacrer et sans lui ce mémoire n'aurait jamais vu le jour

On n'oublie pas nos parents pour leur contribution, leur soutien et leur patience.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragé au cours de la réalisation de ce mémoire. Merci à tous et à toutes.

Résumé

Les exécutions des processus métiers par les différents clients génèrent différentes traces d'exécutions qui sont dotées de données relatives à divers aspects de l'environnement d'invocation.

Dans ce projet de fin d'étude nous étudions le problème de l'interrogation des données des processus métiers modélisés sous forme d'automates d'états finis déterministes.

Nous nous concentrons essentiellement sur les données semi-structurées et nous considérons les traces d'exécutions modélisées sous format XML.

Dans notre approche, un processus métier n'est qu'un fichier XML contenant différentes balises spécifiques aux états et aux transitions de l'automate. Par ailleurs, les traces d'exécutions sont à leur tour représentées par des fichiers XML, dont les éléments correspondent à différents attributs descriptifs des traces.

Ayant en notre possession de telles représentations, nous déployons l'outil **XQuery** pour interroger les bases de données semi-structurées à la recherche de certains patterns ou motifs d'intérêt.

L'approche proposée a été implémentée et expérimentée en utilisant différentes données de test synthétiques issues d'un processus métier réel.

Introduction générale

L'avènement de le technologie des processus métiers (**PM**) a complètement bouleversé la conception et l'implémentation des systèmes d'informations (**SI**) modernes. En effet, la technologie de la gestion des PM (**BPM**) permet de gérer tout le cycle de vie des **PM**, leur évaluation, leur conception, aussi bien que leur configuration et leur mise en œuvre.

Dotées de telles technologies, les entreprises contemporaines peuvent profiter de la distribution des applications basées sur les processus métiers pour faciliter l'intégration des SI hétérogènes. Dans le perspective de l'amélioration des SI orientés vers les (**PM**), beaucoup de travaux de recherches se sont focalisés essentiellement sur la manière de concevoir, de modéliser et de vérifier les (**PM**), tout en utilisant différents modèles de représentation formels qui leur sont associés.

Néanmoins, l'aspect données qui sont afférentes aux PM a été reléguée à un deuxième plan et n'a pas bénéficié de tout l'intérêt et toute l'importance qu'il revête. En effet, l'analyse des travaux ayant plus ou moins abordé la prise en compte des données relatives à l'exécution des PM fait ressortir un déficit flagrant quant à la gestion et l'exploitation de ces données.

Les peu de travaux qui existent se concentrent essentiellement sur l'aspect analyse des données par le déploiement des outils $OLAP$ qui permettent de fournir des indicateurs de gestion et de supervision des (**PM**), par la fourniture d'un ensemble de variables décisionnelles. Dans cette perspective, les $SGBD$ relationnels avec leurs langage SQL sont souvent utilisés pour atteindre les objectifs de production de tels paramètres. Cependant, le constat est que les (**BPM**) peuvent utilisés différents types de modèles pour le stockage des données persistantes, tels que les données semi- structurées au format (**XML**), des bases de données orientées graphe ou des bases de données non structurées...etc.

Cette variété des formats de stockage des données des (**BPM**) exige d'autres approches pour que leur exploitation soit effective et opérationnelle. D'autre part, les données d'exécutions ont des caractéristiques qui leur sont spécifiques, telles que l'exécution répétitive d'une tâche, l'invocation des activités imbriquées dans d'autres structures . . .etc.). Donc, une vision plus efficace et plus rentable pour l'exploitation de ces données s'impose de manière forte et incontournable.

Dans ce mémoire, nous abordons le problème du stockage et de l'interrogation des données de (**BPM**). Notre approche est basée sur la formalisation des processus métiers en tant que automates finis déterministes qui seront stockés sous forme de fichiers XML. Dans notre approche le langage Xquery est utilisé comme langage d'interrogation des données semi-structurées.

En plus, de cette introduction générale et de la conclusion générale, le mémoire est structuré en deux parties. La première partie est un état de l'art qui se compose de trois chapitres qui sont les suivants :

— Le premier chapitre est réservé aux processus métiers, à leurs modèles de représentation et aux technologies les concernant.

— Puis dans le deuxième chapitre, on aborde les données des processus métiers et les technologies de leurs stockages.

— On termine la première partie par une étude approfondie des travaux connexes ayant abordé la problématique de la gestion et de l'interrogation des données des processus métiers en chapitre trois.

La deuxième partie constitue notre contribution à la résolution du problème de l'exploitation des données d'exécution des processus métiers. Elle est organisée autour des deux chapitres suivants.

— Dans le chapitre quatre, on présente notre approche d'interrogation des données des processus métiers. On expose la formalisation de la solution et la modélisation des données et des requêtes.

— Le chapitre cinq et dédié à l'implémentation de l'approche proposée.

On termine le manuscrit par une conclusion générale. Par ailleurs, une riche bibliographie du domaine et des travaux connexes est listée en fin du document.

Première partie

État de l'art

CHAPITRE 1

Processus métiers et systèmes d'informations

1.1 Introduction

Pour atteindre leurs objectifs de production ou de vente des biens et des services, les entreprises réalisent des activités quotidiennes qui satisfont une certaine logique métier relative à la nature de l'activité de l'entreprise. C'est ce qu'on appel communément un processus d'entreprise ou plus précisément **un processus métier** qui sera désigné dans la suite du mémoire par l'abréviation **PM**.

De manière très simple, un **PM** est vu souvent comme une séquence de tâches réalisées en coordination afin d'atteindre un objectif organisationnel. De nos jours, les **PM** profitent d'un regain d'intérêt de la part des gestionnaires, des décideurs et des informaticiens. En effet, la maîtrise des ressources de toute organisation passe inévitablement par une prise en charge efficace de la gestion de ses *PM*. Dans cette perspective les **PM** permettent de donner une vue globale de l'ensemble des ressources et procédures prise en charge par l'organisation lors de l'accomplissement de ses fonctions.

D'autre part, les **PM** constituent la pierre angulaire pour les *PAIS* et contribuent fortement à l'intégration des systèmes d'informations hétérogènes et distribués, ce qui favorise les échanges et les coopérations entre les différents acteurs internes ou externes à l'entreprise.

Dans ce chapitre, on présente la définition précise des **PM**, leurs domaines d'utilisation, leurs modèles de représentation ainsi que les mécanismes permettant la gestion de leur cycle de vie.

Nous commençons par définir le concept de processus métier et nous l'illustrons avec un exemple réel.

1.2 Qu'est-ce qu'un processus métier ?

La notion de processus métier manipule deux concepts : le processus et le métier qui sont définis ci-après.

Processus

Un processus est une suite continue d'opérations ou tâches exprimant la manière de fabriquer, de réaliser quelque chose.

Métier

Un métier est l'exercice d'une activité par une personne dans un domaine professionnel. D'une manière plus claire, on définit un processus métier, comme suit :

Définition 1.1 *Un processus métier est un ensemble de taches liées les unes aux autres qui prennent fin à la livraison d'un service ou d'un produit à un client. Le processus métier a également été définit comme une séquences d'activités et de tâches réalisées en coordination afin d'atteindre des objectifs de gestion [1].*

Exemple 1.1 *L'exemple simple de la figure ci-dessous, permet d'illustré le concept de **PM**. On y présente les tâches à réaliser. La fonction du processus de l'exemple permet le retrait d'une boisson pour se rafraichir. On remarque que l'ordre des tâches est séquentiel. La séquence d'activités débute par l'introduction d'une ou de plusieurs pièces de monnaie, après l'utilisateur doit choisir la boisson, qui se passe à l'étape de préparation. Une fois la boisson sera prête, elle sera desservie à l'utilisateur qui peut se rafraichir.*

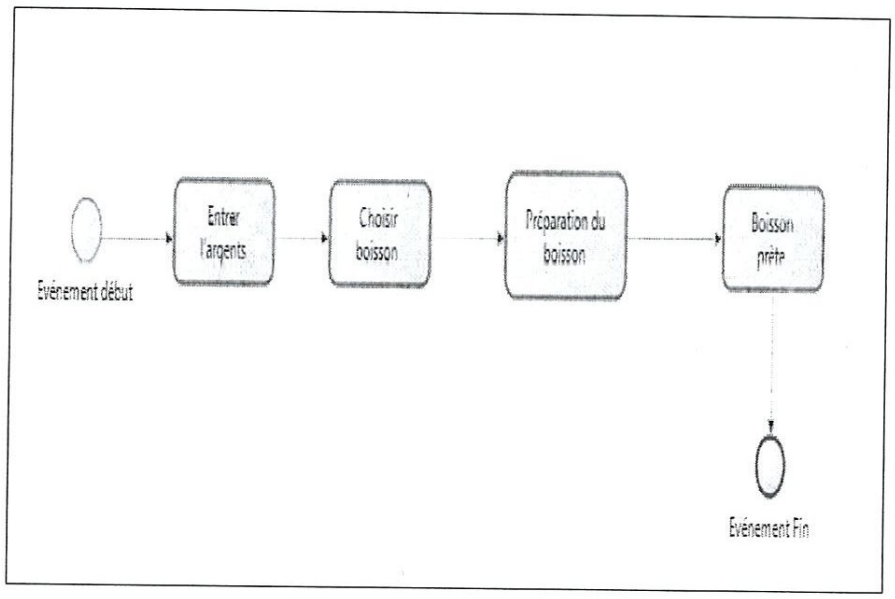

FIGURE 1.1 – Modélisation du processus métier réalisé par un distributeur de boissons.

1.3 Domaines d'utilisation des processus métiers

Les processus métiers sont utilisés de manière intensive dans les administrations, les banques, les entreprises économiques et les organismes de services. Nous distinguons essentiellement les domaines d'utilisation suivants :

— Gestion administrative : Les différentes fonctions de l'entreprise peuvent être gérées comme des processus (paie, gestion des ressources humaines, comptabilité, ...).

— Processus industriels :Processus d'assemblage des produits,processus d'emballage, suivi de la maintenances, gestion des stocks, ...

— Services :Processus de réservation d'un voyage, processus de gestion des vols.

— Banque :Processus de vérification des crédit, gestion des prêts bancaires, suivi de compatibilités, gestion des comptes, ...

Une question évidente qui peut se poser est la suivante : *comment représenter, modéliser et spécifier les processus métiers ?*

La section prochaine vise à répondre à cette question.

1.4 Modélisation des processus métier

La représentation d'un processus se fait par un modèle qui détaille les différentes étapes par lesquelles passe le **PM** afin de réaliser l'objectif attendu. Différents modèles, dotés de niveau d'expressivité variés ont été proposés dans la littérature de recherche pour représenter les différentes abstractions utiles aux fonctionnement des **PM**. Dans ce qui suit, on expose de manière non exhaustive ces modèles et chaque modèle sera illustré par un exemple.

1.4.1 Les automates finis déterministes (AFD)

Définition 1.2 *Les automates sont des objets mathématiques très utilisés en informatique. Ils permettent de modéliser un grand nombre de systèmes informatiques [2]. Un automate est un ensemble d'états du système, reliés entre eux par des transitions qui sont marquées par des symboles. On utilise les automates fini déterministe, parce que à un instant donné, ayant atteint un état du processus métier, l'utilisateur doit être précis sur les futures actions à exécuter. Par conséquent l'automate doit être déterministe [3].*

Exemple 1.2 *Processus représenté par un AFD*

La figure 1.2, expose une séquence d'activités d'un processus d'inscription en ligne, modélisée par un automate fini déterministe. Le fonctionnement de ce processus est le suivant :

Après une simple connexion, on accède au système pour remplir un formulaire d'inscription. L'inscription sera soit rejetée soit acceptée. Si elle est validée, on passe à d'autres tâches qui sont à leur tour séquentielles (affectation du groupe, affichage listes, ...) avant d'atteindre la fin de la procédure d'inscription.

1.4.2 Réseaux de pétri RDP

Définition 1.3 *Un réseau de Pétri est un modèle formel très utilisé pour représenter divers systèmes dynamiques travaillant sur des variables discrètes. Formellement, il est spécifié par un quadruplet($\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{W}$),[4] où :*

— $\mathcal{P}$ *:est un ensemble fini de places ;*

— $\mathcal{T}$ *:est un ensemble fini de transitions ;*

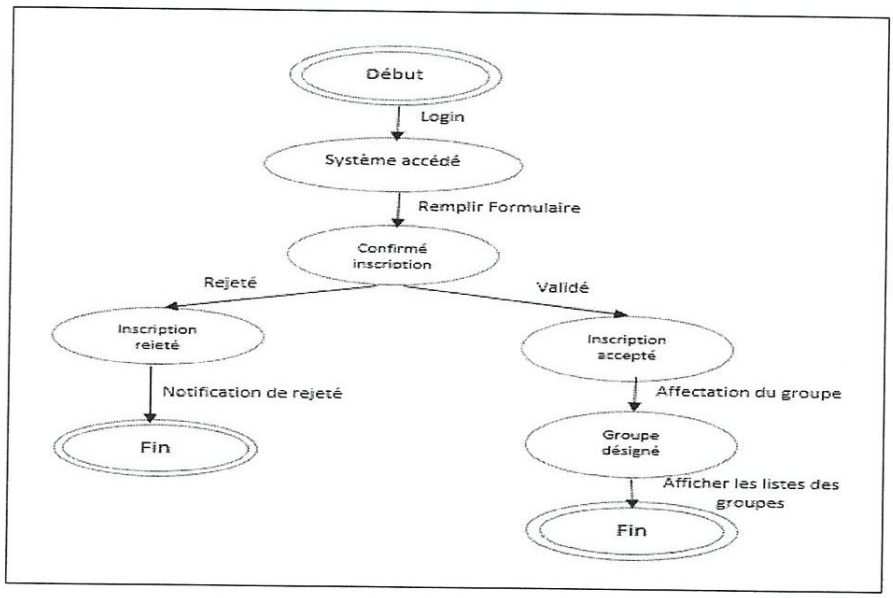

FIGURE 1.2 – Processus d'inscription en ligne modélisé par un AFD.

— $\mathcal{F}$ *:est un ensemble fini d'arcs pondérés connectant une place à une transition et vice versa ;*

— $\mathcal{W}$ *.est une fonction qui associe à chaque arc un poids sous forme d'un entier strictement positif,* $\mathcal{W} : \mathcal{F} \longrightarrow N$*, les poids sont égaux à 1 par défaut.*

Exemple 1.3 *La figure 1.3 suivante permet d'illustrer l'ensemble des tâches à suivre afin de réserver un billet d'avion. L'utilisateur accède au système par une simple inscription. Après la sélection du choix de vol, l'utilisateur passe à la procédure du paiement. Lorsque le vol est confirmé puis payé, le billet sera enfin prêt. Il est possible aussi que l'utilisateur puisse annuler sa réservation avant de passer à la procédure du paiement. En cas d'annulation un message lui sera envoyé.*

1.4.3 Unified Modeling Language UML

Définition 1.4 *Langage de modélisation unifié est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donne chacun une vision différente du projet à traiter [5].*

UML nous fournit donc des diagrammes pour représenter le logiciel à développer, on distingue, notamment deux diagrammes utiles à la représentation des processus métiers, le diagramme d'activité et le diagramme de séquence.

1.4.3.1 Diagramme d'activités

— Un diagramme d'activités est une variante du diagramme d'états-transitions. Il permet d'avoir une vue sur l'enchainement des opérations à réaliser.

— Un diagramme d'activités visualise un graphe d'activités qui modélise le comportement interne d'une méthode, d'un cas d'utilisation ou plus généralement d'un processus.

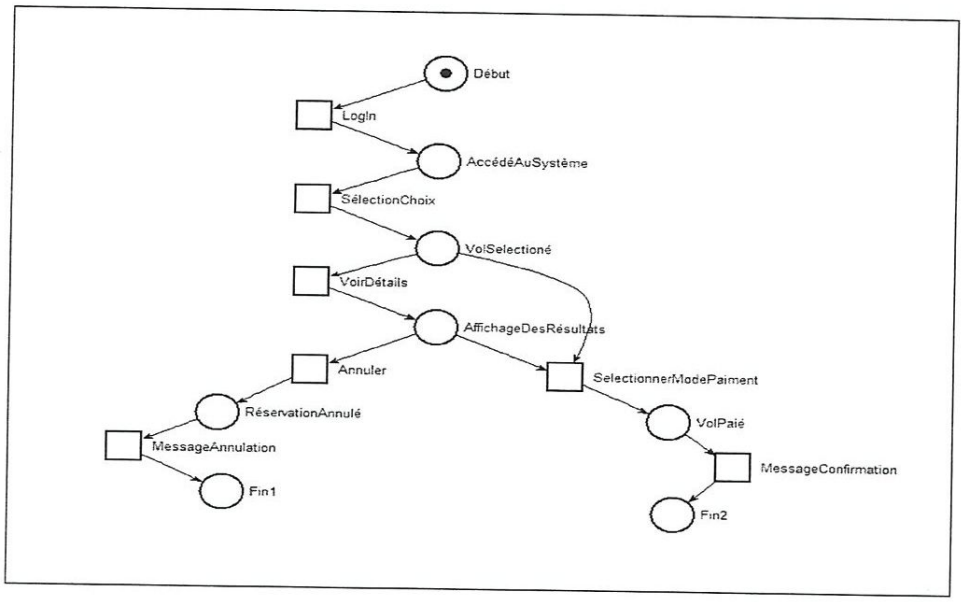

FIGURE 1.3 – Processus de réservation du vol en ligne modélisé par un RDP.

Exemple 1.4 *La figure 1.4 suivante illustre une validation d'une commande modélisée par un diagramme d'activité. Cette validation passe par différentes étapes pour atteindre la fin de la procédure. Ces étapes commencent, tout d'abord, par une vérification de la commande, ensuite la commande sera rejetée ou enregistrée.*

1.4.3.2 Diagramme de séquences

Les diagrammes de séquences permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs. Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie(Une ligne de vie représente un participant, une instance à une interaction), un message définit une communication particulière entre des lignes de vie (objets ou acteurs).

Exemple 1.5 *La figure 1.5 ci-dessous, illustre une validation d'une commande modélisée par un diagramme de séquences. Elle illustre les échanges de messages (les activités réalisées) entre les tâches de la validation.*

En plus des modèles graphiques, certains langages issus du domaines des processus métiers sont dotés de capacités de représentation. Il s'agit des deux standards **BPMN** et **BPEL** qui sont présentés dans ce qui suit.

1.4.3.3 Utilité d'UML

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture du logiciel à développer. Les différents éléments représentables sont " les activités d'un objet/logiciel, Acteurs, Processus,

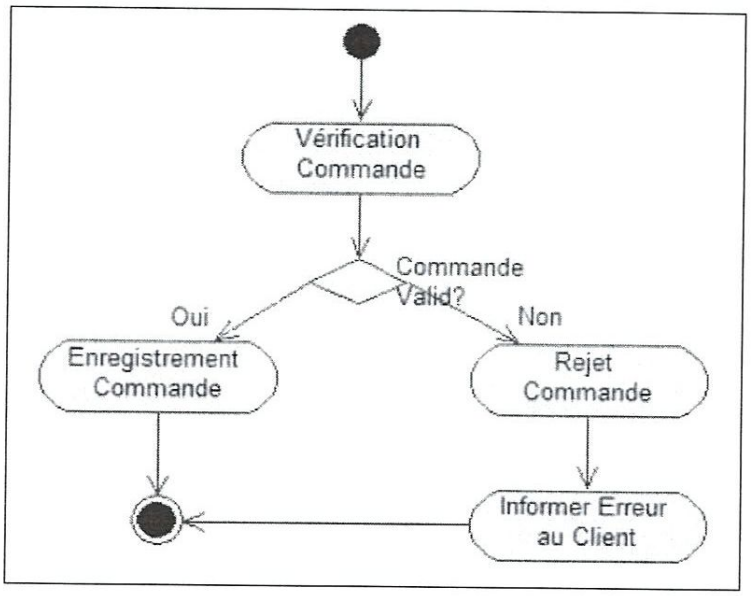

FIGURE 1.4 – Validation d'une commande modélisée par un diagramme d'activités

Schéma de base de données, Composants logiciels et Réutilisation de composants". Grâce aux outils de modélisation UML, il est également possible de générer automatiquement une partie de code, par exemple en langage Java, à partir des divers documents réalisés [6]. Parmi les diagrammes offerts par UML, deux sont particulièrement intéressants dans le contexte des processus métiers.

1.4.4 Business Process Model and Notation BPMN

BPMN est une représentation graphique des processus métiers. Il propose un ensemble d'objets graphiques et de règles définissant les connexions disponibles entre les objets [7] spécifiant le **PM**. *BPMN* comprend les blocs de construction de base suivants :

— Objets de flux : événements (cercles), activités (rectangles aux angles arrondis) et passerelles (losanges).

— Objets de connexion : composés principalement de flèches, ils indiquent un flux de séquence (flèches pleines), un flux de messages (flèches en pointillés) et des associations.

— Voies de natation : piscines (conteneur graphique) et voies (sous-partition de la piscine)

Artefacts : objets de données, groupes et annotations.

Exemple 1.6 *La figure 1.6 ci-dessous illustre un exemple de processus avec le langage* ***BPMN****.*

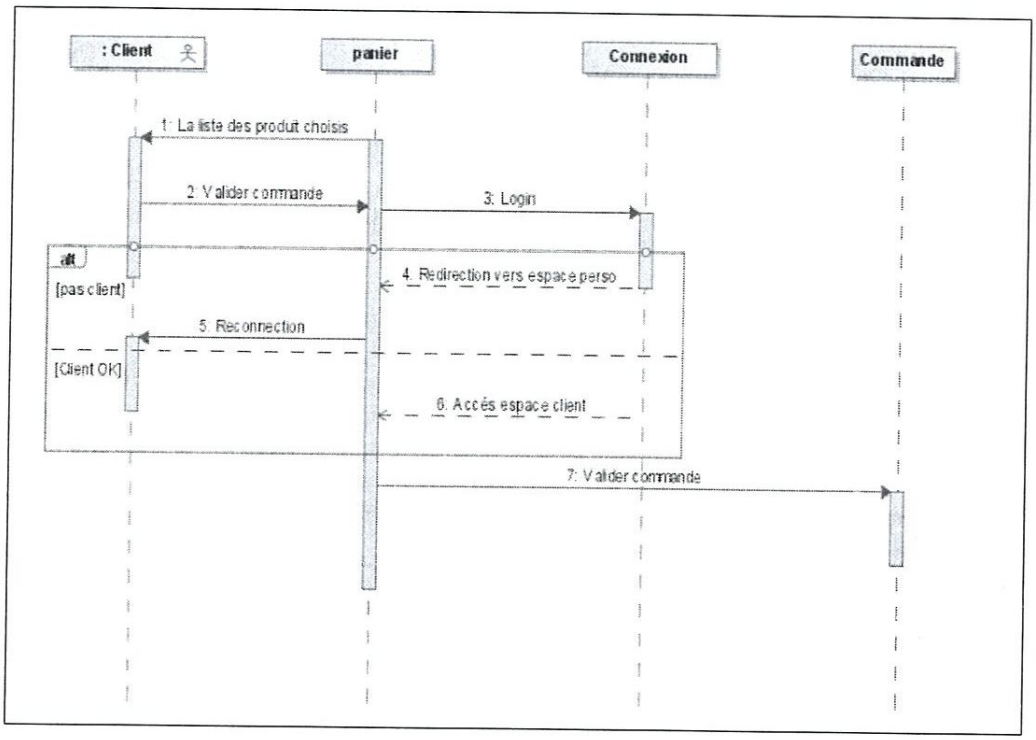

FIGURE 1.5 – Validation d'une commande modélisée par un diagramme de séquences

1.4.5 Business Process Execution Language BPEL

BPEL (Business Process Execution Language) est un langage basé sur **XML**. Les programmeurs utilisent **BPEL** pour définir le mode d'exécution d'un processus métier impliquant des services Web. *BPEL* est souvent associé à **BPMN** (Business Process Management Notation), une norme permettant de représenter graphiquement les processus métier. Dans de nombreuses entreprises, les analystes utilisent **BPMN** pour visualiser les processus métier et les développeurs transforment les visualisations en **BPEL** pour exécution [8].

1.5 La gestion des processus métiers

La gestion des processus métiers comprend des concepts, des méthodes et des technique permettant de prendre en charge la conception, l'administration, la configuration, l'activation et l'analyse des processus métiers. La base de la gestion des processus métiers est la représentation explicite des processus métiers avec leurs activités et leurs contraintes d'exécution. Une fois les processus opérationnels définis, il peuvent être analysés, améliorés et mis en œuvre[9].

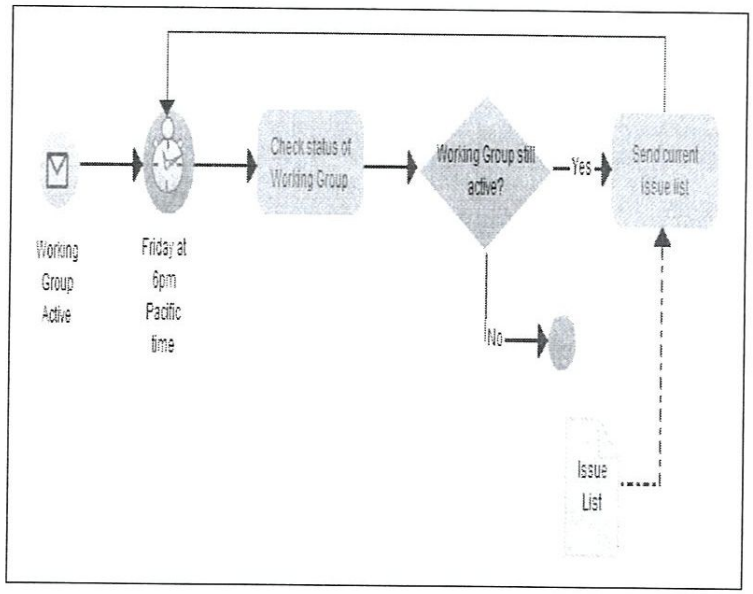

FIGURE 1.6 – Un processus métier modélisé à l'aide de **BPMN**.

1.6 les bénéfices de la gestion des processus métiers

Aujourd'hui les entreprises cherchent des moyens d'améliorer leur fonctionnement global pour en augmenter les bénéfices à plusieurs niveaux. Par la mise en place d'une gestion des processus métiers de l'entreprise et des outils associés, les entreprises ont la possibilité de formaliser, exécuter, automatiser et tracer tout ou parties soient encore manuelles et qui ne seront d'ailleurs très probablement, jamais automatisables.[10]

1.7 Les enjeux de la gestion des processus métiers

Le " Business Process Management" **BPM** s'est imposé comme un outil d'innovation indispensable pour assurer l'automatisation et l'amélioration du fonctionnement des entreprises quelque soient leurs secteurs d'activité. Il offre la possibilité d'optimiser les différents types de processus de travail existants dans chaque société. Face à une concurrence toujours plus forte, le maintien de la compétitivité des entreprises passe par une parfaite maîtrise des processus métiers et de leurs évolutions et, sur cet aspect, les apports du **BPM** sont décisifs. En effet, les solutions de **BPM** accompagnent les entreprises sur toutes les étapes de mise en œuvre, de la conception à la mise en production [11].

1.8 Cycle de vie des Processus Métiers

Le cycle de vie des processus métiers est illustré à la figure 1.8, il se compose de phases qui sont liées les unes aux autres. Les phases sont organisées dans une structure cyclique, montrant leurs dépendances logiques. Ces dépendences n'impliquent pas un ordre temporel strict dans lequel les phases doivent être exécutées [12].

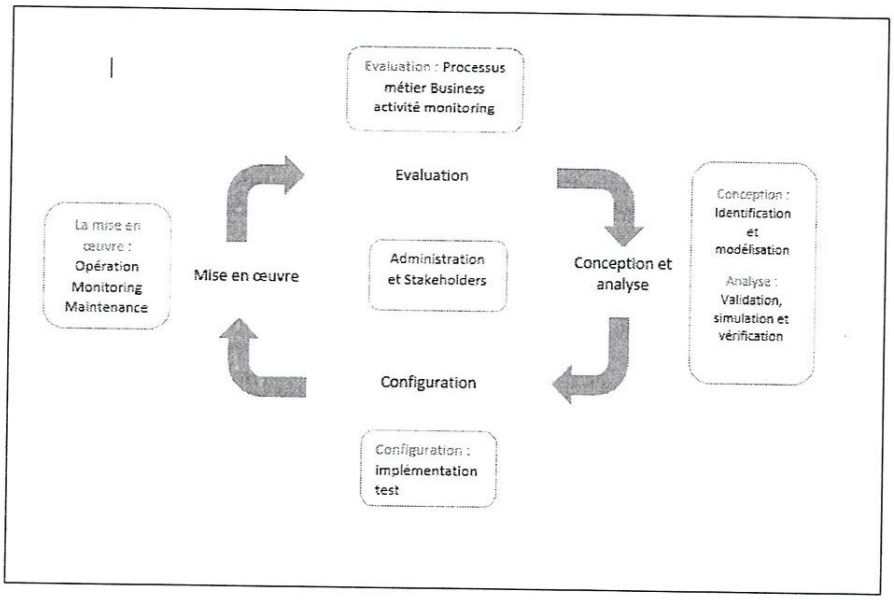

FIGURE 1.7 – Cycle de vie d'un processus métier.

1.8.1 Évaluation

La phase d'évaluation utilise les informations disponibles pour évaluer et améliorer les modèles de processus métiers et leurs implémentations, les journaux d'exécution sont analyses en utilisant un module spécifique de surveillance des activités **BAM** et les techniques d'extraction de processus (Process mining techniques), ces techniques visent à identifier la qualité des processus métier et à l'adéquation de l'environnement d'exécution.

1.8.2 Conception et analyse

Le cycle de vie de n'importe quel processus métier commence toujours par cette phase en se basant sur des enquêtes menées sur les processus métiers et leur environnement organisationnel et technique. Ces processus seront identifiés, revus, validés et représentés par des modèles. Les techniques de modélisation des processus métier ainsi que les techniques de validation, simulation et vérification sont utilisées durant cette phase. La modélisation des processus métier est au cœur de la sous-phase technique lors de la conception du processus. Une fois que la conception d'un processus métier est terminée, on passe directement à l'étape de validation, en utilisant un atelier de travail (*Workshop*), pour vérifier si toutes les instances des processus métier validées sont reflétés par le modèle. Les techniques de la simulation peuvent être utilisées pour soutenir l'étape de validation, car certaines séquences d'exécution non désirées peuvent être simulées pour montrer des déficits dans le modèle, Il permet également aux partenaires de parcourir le processus étape par étape pour vérifier si le processus expose réellement le comportement souhaité. Au cas contraire, des opérations de redressement sont a opérer. Ainsi, la modélisation des processus métier a un caractère évolutif dans le sens que le modèle est analysé et amélioré afin de représenter le processus métier

désiré, sans aucune lacune.

1.8.3 Configuration

Après avoir terminé l'étape de conception et d'analyse, le processus métier sera prêt pour être implémenté. Si on veut utiliser un système logiciel pour l'implémentation, ce dernier doit être configuré selon l'environnement organisationnel de l'entreprise et le processus métier dont il devrait contrôler le fonctionnement. Cette configuration comprend les interactions des employées avec le système ainsi que l'intégration des systèmes existants avec le système de gestion des processus métiers.

1.8.4 La mise en œuvre

La phase de mise en œuvre, ou déploiement du processus englobe l'exécution réel du processus métier par des instances lancer par des clients. Chaque instance remplit les objectifs métier d'une entreprise et l'initiation d'une instance de processus suit généralement un événement déclencheur, par exemple, la réception d'une commande envoyée par un client. Le système de gestion des processus métier contrôle activement l'exécution des instances des processus métiers définies dans le modèle, la mise en œuvre du processus doit répondre à une coordination de processus correcte, garantissant que les activités de processus sont exécutées en fonction de l'exécution des contraintes spécifiées dans le modèle de processus. Durant la mise en œuvre du processus métier, les données d'exécution utiles sont collectées, généralement dans une certaine forme de fichiers journaux, constituent de séries ordonnées du journal d'entrées, indiquant les événements qui se sont produits au cours des processus métier. Les journaux d'événement constituent la base de l'évaluation des processus dans la phase suivante du cycle de vie des processus métier. C'est à dire la phase de supervision et d'évaluation.

1.9 Conclusion

Dans ce chapitre, nous avons présenté les **PM**, les concepts qui leur sont associés ainsi que les outils BPM permettant de gérer leur cycle de vie. Par ailleurs, les déférents modèles de représentation des PM ont été abordés en détails avec des exemples illustratifs pour chaque modèle. Dans le prochain chapitre, on abordera les données des processus métiers.

CHAPITRE 2

Les données des processus métiers

2.1 Introduction

Comme il a été expliqué dans le chapitre précédent, un processus métier est une succession d'étapes et de tâches réalisant des activités de l'entreprise. L'exécution de ces activités permettra de produire un ensemble de données, avec lesquelles les acteurs de l'entreprise pourront évaluer les performances de l'organisation. A cet effet, les acteurs impliqués dans le **PM** (clients, fournisseurs, partenaires, ...) vont instancier le **PM** et invoquer les différentes activités offertes. Après avoir instancier et exécuter le PM, on aura pour chaque instance un historique, ou une trace d'exécution, à laquelle plusieurs données d'exécutions sont associées.

Pour pouvoir manipuler ces données, il faut impérativement connaitre et maitriser leurs structures. En ce sens, la question qui se pose est la suivante :

Comment les données d'exécution sont stockées et sous quelle forme ?

Dans ce chapitre on aborde cette question de manière détaillée par l'étude des différentes données des **PM**, tout en explicitant les concepts d'instances et traces d'exécution. Par ailleurs, les différentes techniques de stockage des données seront exposées ainsi que les mécanismes de leurs interrogation.

2.2 Notion d'instance d'exécution de PM

Tout processus métier est destiné à être invoqué par des utilisateurs internes ou externes. Chaque exécution du processus métier génère une instance d'exécution. En ce sens, il est possible d'avoir des centaines, des milliers, voire des millions d'instance d'un même PM en cours d'exécution à un instant donné.

Dans ce qui suit, nous allons définir le concept d'instance de processus et on va expliquer les données qui lui sont associées.

2.2.1 Qu'est-ce qu'une instance de processus ?

Instancier correspond à une exécution ou une invocation d'un processus métier par un client, afin d'atteindre les objectifs stratégique [13]. Donc, il s'agit tout simplement d'une réalisation ou un cas particulier du **PM**. A noter que chaque cas sera doté de ses propres propriétés (*utilisateur, historique, état atteint, ...*).

2.2.2 Types d'instances

A chaque nouvelle invocation d'un processus, une nouvelle instance est générée. A un moment donné, on distingue deux types d'instances :

— Les instance actives : Se sont des instances dont l'exécution est encore en cours. C'est qu'elles n'ont pas encore atteint un état final du processus métier exécuté, i ;e. leur état actuel n'appartient pas à l'ensemble des états finaux.

— Les instances achevées : Se sont des instances qui ont terminé leur exécution et dont l'état actuel appartient à l'ensemble des états finaux du processus.

2.2.3 Chemin d'exécution d'une instance

Un chemin est une suite d'étapes ou phases empruntées par une instance d'un processus métiers. Dans le domaine des **PM**, lors de son interaction avec le processus métier, chaque instance peut emprunter un ensemble d'activité qui lui sont spécifiques et qui expriment des choix de gestion de l'utilisateur. La séquence des activités exécutées par une instance représente le déroulement du processus pour cette instance. En effet, c'est une particularisation du processus pour cette instance. Plus précisément une trace. Ainsi, il est possible que deux ou plusieurs instances soient au même niveau d'exécution (même état), mais ayant emprunté des chemins d'exécution différents.

2.2.4 Trace d'exécution d'une instance

Les systèmes de gestion des *PM* permettent de garder l'historique des exécutions des différentes instances lancées. Dans cette optique, toute les exécutions des activités, les chemins d'exécutions et les informations relatives aux ressources doivent être enregistrées dans des fichiers particuliers qu'on appelle : *fichiers logs*. Les informations contenues dans ces fichiers représentent les **traces d'exécutions**.

Une trace d'exécutions contient les différents nœuds représentant l'activation et les événements de la fin des activités, aussi bien que les ressources déployées pour réaliser les tâches. Des traces peuvent varier dans la quantité d'informations qu'elles enregistrent. En général, on peut distinguer trois familles de traces d'exécution [14].

— Traces naïves : Elles fournissent un dossier complet de l'activation des événements (activités) associées à la réalisations de ces activités.

— Traces semi-naïves : les événements d'achèvement des tâches sont tous enregistrés, mais il est possible qu'une information partielle soit non prise en compte car elle ne représente pas un intérêt pour le gestionnaire.

— Traces sélectives : pour ce type, seul un sous-ensemble des événements est enregistré.

Exemple 2.1 *Traces d'exécution*
Dans la figure 2.1, ci-dessous, nous exposons un exemple simple d'un processus métiers imaginaire modélisé par un AFD, puis nous illustrons la notion de trace d'exécution en donnant des exemples de traces. Dans cet exemple, les lettres alphabétiques représentent des activités du processus métiers et les chiffres sont les états par lesquels passe les instances. Ainsi, on peut donner les traces d'exécution suivantes.

— *Trace 1 : a.b.c (l'instance est à l'état 3)*

— *Trace 2 : a.b.c.d (l'instance est à l'état final 4)*

— *Trace 3 : a.e (l'instance est à l'état 5)*

— *Trace 4 : a.e.f (l'instance est à l'état final 6)*

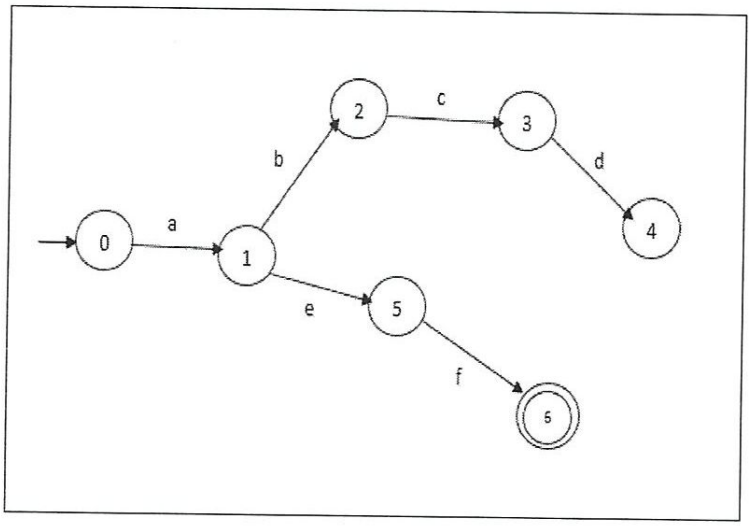

FIGURE 2.1 – Exemple d'automate et de trace d'exécution.

2.2.5 Utilité des traces d'exécution des instances

les traces d'exécution sont exploitées et les informations qu'elles contiennent sont très utiles. En fait, analysées les informations enregistrées permettra de répondre à des questions sur les aspects suivants :[15]

- Le fonctionnement du système : Comment marche-t-il ?
- L'évaluation des performances : Le système est-il performant ?
- La validation : le système respecte t il les spécifications ?
- Le dysfonctionnement : Quelles sont les erreurs de conception et d'exécution ?
- L'optimisation : Comment pourrait-il marcher mieux ?

2.3 Exemples illustratif

Pour illustrer les concepts précédents, nous avons choisi le domaine "Gestion Hôtellerie" permettant la réservation d'une chambre d'hôtel à distance pour un client. Ce processus consiste en une série d'actions ou activités à réaliser pour atteindre l'objectif de réservation d'une chambre par le client.

2.3.1 Processus métier pour la Réservation d'une chambre

Dans la figure 2.2, on présente les grandes étapes qui expriment la modélisation du PM de réservation d'une chambre d'hôtel.

2.3.2 Réseau de Pétri pour la Réservation d'une chambre

La procédure de réservation est modélisée par un réseau de pétri. La figure 2.3 illustre l'ensemble des étapes à suivre pour réserver une chambre. La procédure commence par la sélection de la ville désirée par le client, ensuite l'hôtel, en plus le client

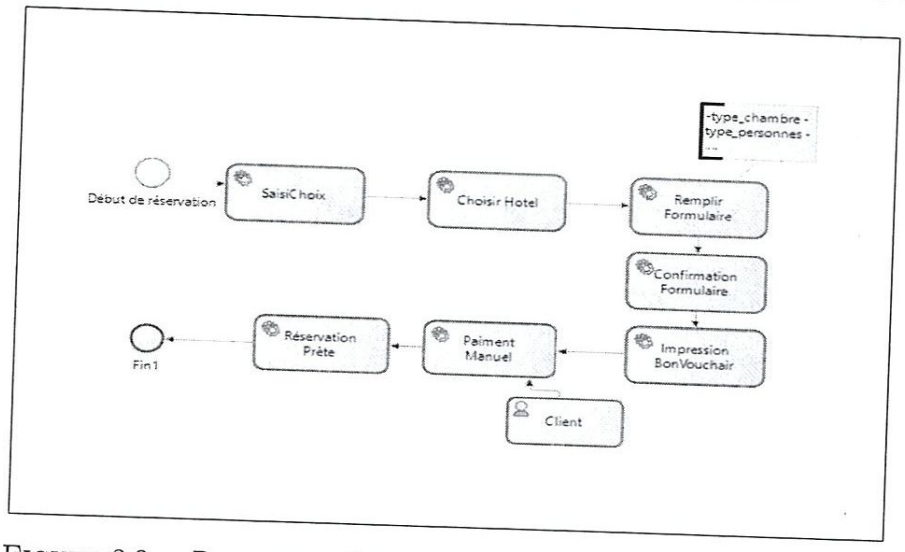

FIGURE 2.2 – Processus Métier pour une Réservation d'une chambre.

doit remplir un formulaire selon ses besoins. Enfin, la chambre est réservée et le client peut passer à la phase de paiement.

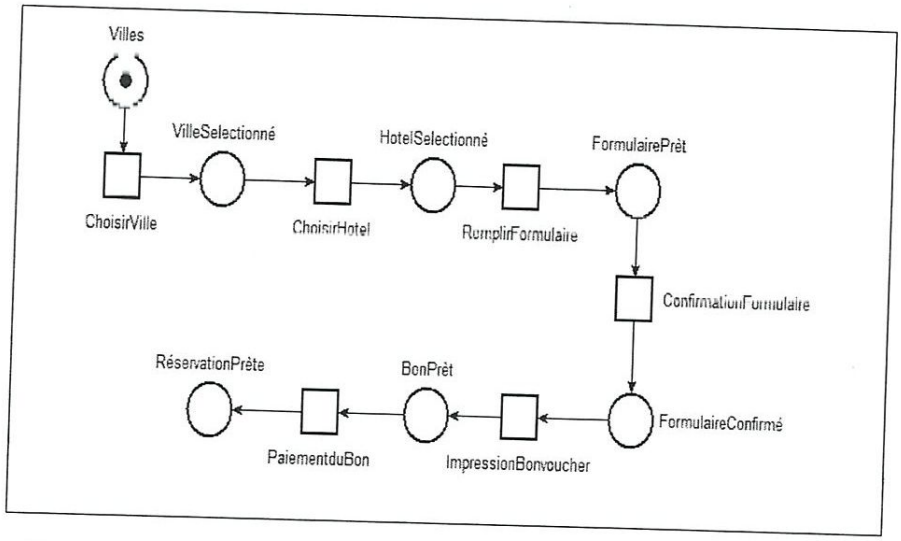

FIGURE 2.3 – Réseau de Pétri pour la réservation d'une chambre.

2.3.3 Exemples de traces

En se basant sur le modèle de l'exemple précédent, on peut donner les quatre traces d'exécution suivantes.

— δ_1= ChoisirVille.ChoisirHotel.

— δ_2= ChoisirVille.ChoisirHotel.RemplirFormulaire.

— δ_3= ChoisirVille.ChoisirHotel.RemplirFormulaire.ConfirmationFormulaire

— δ_4= ChoisirVille.ChoisirHotel.RemplirFormulaire.ConfirmationFormulaire. ImpressionBon.PaiementBon.

Parmi ces traces les trois premières sont actives, par contre la dernière est achevée.

2.4 Les données des processus métiers

D'une manière générale, dans le domaine informatique une donnée est la représentation d'une information dans un programme. Soit dans le texte du programme lui même, soit en mémoire durant l'exécution [16]. Cependant, le concept de données des processus métiers relève certaines particularités. En effet, plusieurs types de données sont liées au domaine des **PM**. Dans ce qui suit, on présente la classification des données du domaine des **PM**.

2.4.1 Données relatives aux modèles

Un modèle de données est une représentation qui décrit de façon abstraite comment sont représentées les données dans un système d'information ou une base de données [17]. Donc, les données relatives aux modèles des processus métiers expriment la spécification de la logique métier, l'ordonnancement des tâches ainsi que les contraintes qui gouvernent leur exécution. Ainsi, les informations relatives aux modèles sont des données nécessaires à l'exécution du processus métier.

Exemple 2.2 *Données du modèle réservation hôtel*
Le processus métier de réservation d'une chambre d'hôtel de la figure 2.3 est représenté par un réseau de Petri contient un ensemble de données qui sont :

— *Les différentes étapes du processus.*

— *Les activités permettant de passer d'une étape à une autre.*

Les conditions et contraintes utiles à l'exécution des activités. Par exemple, les contraintes d'ordre et de temps.

Ces informations sont stockées dans des structures de données adéquates. Par exemple, on peut utiliser des tableaux, des listes, des piles, des arbres ou des bases de données relationnelles ou autres types de BDD.

2.4.2 Les données métiers

Pour qu'on puisse lancer l'exécution d'un processus métier, on a besoin de ce qu'on appel les données métiers, afin d'accomplir les différentes activités qui lui sont associées. Elles expriment **les règles de gestion** spécifiques à l'organisation. Par exemple, il peut s'agir de règles de calcul, de délais imposés pour accomplir certaines tâches ou simplement de contraintes que doivent vérifier les instances actives.

Exemple 2.3 *Processus d'inscription universitaire*
Le formulaire d'inscription doit être rempli par l'étudiant, les frais d'inscription varient suivant la catégorie des étudiants, les photos de l'étudiant sont exigées lors de la première inscription, l'année du bac doit être l'année en cours, le nombre d'inscriptions ne doit pas dépasser un seuil, ... Ces informations sont utiles à l'inscription. Donc, elles constituent des données métiers.

2.4.3 Données d'exécution

Chaque exécution d'un processus métier, par une instance particulière, est enregistrée et stockée dans les fichiers logs, sous la forme de traces qui représentent les historiques des exécutions. Ces données fournissent des informations clés sur ce qui s'est produit à chaque exécution de chaque tâche, chaque choix que l'utilisateur a opéré lors de l'évolution des exécutions des tâches [18].

Exemple 2.4 *Données d'exécution*

Après avoir instancier un processus métier plusieurs fois, en lançant son exécution, à un instant donné, on peut faire une extraction des traces exécutées par les différentes utilisateurs.

Le tableau de la figure 2.4 ci-dessous montre une trace générée avec son identificateur, les dates de début et de fin de chaque tâche réalisée par cette instance ainsi que le nom de l'utilisateur.

ID ^	Process name	Version	Start date	Started by	End date
4	Exemple1	1.0	03/26/2019 1:10 PM	Walter Bates	03/26/2019 1:10 PM
3	Exemple1	1.0	03/26/2019 1:10 PM	Walter Bates	03/26/2019 1:10 PM
2	Exemple1	1.0	03/26/2019 1:07 PM	Walter Bates	03/26/2019 1:07 PM
1	Exemple1	1.0	03/26/2019 1:05 PM	Walter Bates	03/26/2019 1:05 PM

FIGURE 2.4 – Données d'exécution d'une instance.

2.4.4 Données sur les ressources

Les processus métiers manipulent différentes ressources au niveau de l'organisation. Il peut s'agir de ressources humaines, matérielles ou financières qui sont utiles à l'exécution des processus. De ce fait, il est impératif que les données sur ces ressources soient stockées de manière adéquate et accessibles aux instances qui vont les consommer. Les ressources associées aux processus métiers peuvent être internes ou externes à l'organisation.

Exemple 2.5 *Ressources du processus métier gestion des vols d'une compagnie aérienne*

- *Ressources matériels : base de données des vols, avions, des agences de voyages et équipement imprimante(pour imprimer les billets, par exemple).*
- *Ressources humaines : Personnel, pilotes et opérateurs aériens, ...*
- *Ressources financières : gestion des comptes pour rembourser les comptes, faire des partenariats, payer les fournisseurs et les réparateurs, ...*

A noter que même si ces données ne sont pas spécifiques aux processus métiers eux même, mais leur disponibilité est incontournable pour la gestion des processus.

Dans notre travail on s'intéresse uniquement aux **données d'exécutions**, vue leur importance. Et à présent, nous allons exposer dans la prochaine section les différentes techniques utilisées pour leur stockage.

2.5 Les techniques de stockage de données d'exécutions

Les bases de données relationnelles ont fait leur preuve dans différentes domaines de l'informatique pour le stockage des données persistantes. Leur avantages relatives à la sécurité, au contrôle d'accès et aux niveaux de privilèges (vues) sont fortement présents. Dans cette perspective, les processus métiers ont profité de la technologie des bases de données relationnelles pour le stockage de différents types de données, et particulièrement les données d'exécution. Dans les environnements de production, il est recommandé d'utiliser une base de données externe pour stocker les données d'exécution. Cela permet à l'administrateur d'avoir l'avantage du contrôle sur les allocations d'espace disque, la haute disponibilité, la reprise après incident et l'optimisation de base de données [19].

Mais cela n'empêche pas d'avoir recours à d'autres types de techniques de stockage des données d'exécution. Dans cette section on expose certaines techniques de stockage des données d'exécution.

2.5.1 Bases de données relationnelles

Une base de données relationnelle est une collection de données organisées sous la forme de tables définies de façon formelle, à partir desquelles les données sont accessibles et assemblées sans avoir à réorganiser les tables de la base de données. L'interface standard pour une base de données relationnelle est le langage SQL[1]. Les commandes SQL sont utilisées pour interroger de façon interactive les informations contenues dans la base et pour rassembler des données dans le cadre de rapports.

Plus concrètement, une base de données relationnelle est un ensemble de tables contenant des données qui rentrent dans des catégories pré-définies. Chaque table contient une ou plus plusieurs catégorie(s) dans des colonnes. Chaque ligne renferme une instance unique de donnée des catégories définies par les colonnes [20].

Exemple 2.6 *Tables relationnelles du domaine commercial*
Une base de données du domaine commercial peut contenir les tables de ***commandes***, ***Produits*** *et* ***Clients***. *La table* ***Client*** *contient les attributs suivants :*
Id, Nom, Prnom, AdresseetNumrodetIphone. *Par contre, la table* ***Commande*** *est décrite par les cinq propriétés suivantes :*
Id – client, Id – produit, Quantit – produit, Prix – unitaire *et* Date – commande.

2.5.2 Données semi-structurées (Fichiers XML)

Le langage **XML**, pour eXtensible Markup Language, est un métalangage informatique de balisage générique qui est un sous-ensemble du Standard Generalized Markup

1. SQL : Structured Query Language

Language **SGML**. Sa syntaxe est dite "extensible", car elle permet de définir différents langages avec chacun son vocabulaire et sa grammaire, comme **XHTML, XSLT, RSS, SVG,** .... Cette syntaxe est reconnaissable par son usage des chevrons (<,>) encadrant les noms des balises. L'objectif initial de **XML** est de faciliter les échanges automatisés de contenus complexes entre les systèmes d'informations hétérogènes issus de différents acteurs [21].

Exemple 2.7 *fichier XML stockant des données d'un compte bancaire*
Pour montrer l'importance du langage XML dans le domaine des processus métiers, le fichier suivant au format XML illustre le stockage d'un enregistrement relatif à la création d'un compte client au niveau d'une banque.

```
< ?xml version="1.0" encoding= "UTF-8"? >
    <Compte>
        <numéro> 12345000/21 </numéro>
        <montant> 1000 </montant>
        <monnaie> dinars </années>
        <années> 20 </années>
        <taux-intérêt> 4.5 </taux-intérêt>
        <Taxe-annuel> 200 < Taxe-annuel>
    </Compte>
```

2.5.3 Bases de données orientées graphes

Une base de données orientée graphes, est un type de base de données **NoSQL**[2] qui utilise la théorie des graphes pour stocker et interroger des données schématisées par des relations. Elle se compose, essentiellement, d'un ensemble de nœuds et d'arêtes. Chaque nœud représente une entité et chaque arête, une connexion ou une relation entre deux nœuds.

Les bases de données orientées graphes conviennent, particulièrement, à l'analyse d'interconnexions, ce qui explique le grand intérêt qu'elles suscitent pour l'exploration des données issues des réseaux sociaux, des réseaux d'ordinateurs, et tout récemment au domaine des processus métiers, vue l'évolution et l'interconnexion des données. Elles sont également utiles pour manipuler les données dans des disciplines impliquant des relations complexes et des schémas dynamiques, comme les systèmes de recommandation en ligne [22].

Exemple 2.8 *Schéma d'une Base de données graphe*
Prenons un exemple simple, une personne travail dans une entreprise.

2.6 Interrogation des données d'exécutions des PM

Après la phase de stockage des données d'exécution, il faut passer à la phase de leur exploitation. Dans cette perspective, le soucis est d'extraire le maximum d'informations utiles pour le gestionnaire. Autrement, on se pose la question suivante :

2. NoSQL est l'acronyme de Not Only SQL.

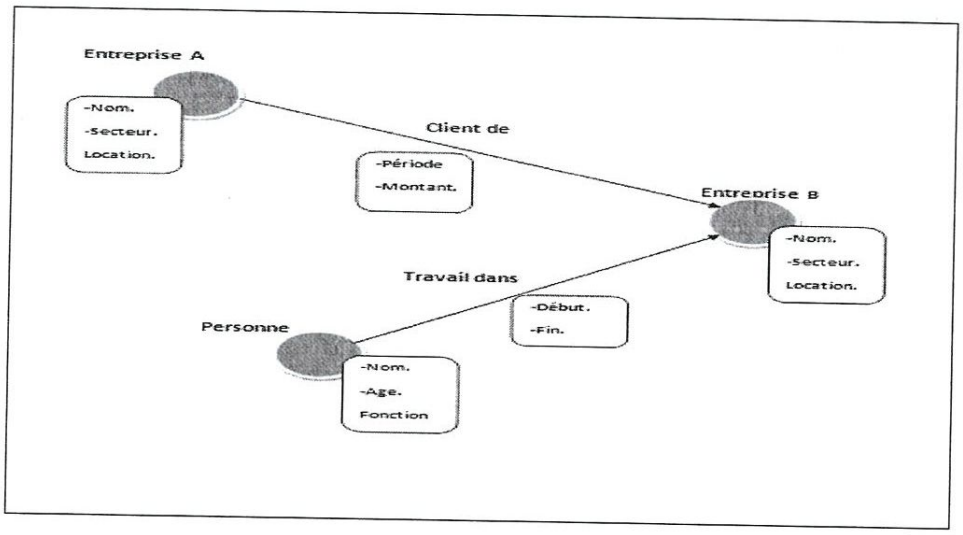

FIGURE 2.5 – Exemple de BDD graphe.

Qu'est-ce qu'on peut faire avec les données d'exécutions des PM ?
En considérant l'aspect exploitation de données d'exécutions après la phase d'acquisition, la plupart des solutions offertes consistent en des librairies fournissant des primitives d'accès aux données enregistrées. Dans le cas le plus simple, il s'agit des accès en lecture et en écriture. Dans d'autres cas, des opérations plus complexes peuvent être fournies comme, par exemple, le filtrage, le calcul et l'extraction de métriques statistiques basées des requetes d'agrégation de données ; i.e ; sommes, moyenne, ... Néanmoins, dans la majorité des cas, ces librairies d'exploitation ont pour objectif la représentation visuelle de l'information et non l'extraction de connaissances d'un niveau sémantique plus élevé. Par conséquent, l'extraction d'informations utiles et la détection d'anomalies dépendent majoritairement de l'expérience et du savoir faire des développeurs [23].

Vu qu'il y a de différentes techniques pour le stockage, cela implique différentes méthodes d'extraction des données stockées, et cela se fait par des outils logiciels spécifiques.

Dans ce qui suit, nous allons aborder cette question, en étudiant la façon dont les données sont exploitées pour les trois types de systèmes de stockage précédents.

2.6.1 Le langage SQL pour les bases de données relationnelles

Les logiciels qui permettent de créer, utiliser et maintenir des bases de données relationnelles sont des systèmes de gestion des bases des données relationnels ou **SGBDR**. Pratiquement, tous les systèmes relationnels utilisent le langage **SQL** pour l'interrogation des bases de données. Ce langage permet de formuler des opérations de l'algèbre relationnelle, telles que l'intersection, la sélection et la jointure [24]. Aujourd'hui **SQL** est un langage très populaire et connu de tout informaticien. Pour cette raison, il ne sera pas traité dans notre travail.

2.6.2 XQuery et Xpath pour les fichiers XML

Un document XML est un arbre composé d'un ensemble d'éléments structurés en balises. Cette structuration hiérarchisée ouvre la voie au traitement automatique du document. Les fichiers *XML* sont encodés en texte brut, c'est pourquoi on peut les ouvrir, les éditer et les lire avec n'importe quel éditeur de texte [21]. Par ailleurs, avoir une structure clairement définie et extensible favorise la prolifération d'outils de traitements des documents XML pour l'extraction de leur structure, aussi bien que de leur contenu. Aussi, le test de validité d'un document par rapport à une structure préalablement définie (*DTD ou XML schema*) est alors possible.

Il existe une panoplie d'outils d'exploitation des documents XML, comme : *Xpath* [25], Xquery [26] pour interroger des documents et d'identifier un ensemble de nœuds dans un document XML, ou encore XSL, XSLT pour transformer des documents, ...

Une base de données **XML** nécessite d'indexer non seulement le contenu des éléments mais aussi la structure, les relations entre éléments pour que des requêtes **XPath** comme les sélecteurs de nœuds ou de chemins puissent utiliser l'index [27].

2.6.3 Neo4J et Cypher pour les bases de données graphes

Les triple-store, qui sont une forme particulière de base de données graphe, fonctionnent exclusivement avec le langage *SPARQL* [22]. A noter que ce modèle est inspiré de celui utilisé dans le domaine des bases de données relationnelles. Néanmoins, la différence réside dans le fait que les données persistantes peuvent être stockées dans des bases de données non relationnelles, c'est à dire des bases de données orientées graphe. Dans ce cas le filtrage à opérer, sera exprimé en tenant compte des structures arborescentes relatives au schéma de la base de données [28]. Le mécanisme d'interrogation des données graphes reste valable et sera efficient indépendamment du type de base de données utilisé. En d'autres termes, nous pouvons extraire des données en explorant les bases de données orientées graphe, par le déploiement d'un langage dédié à ce type de modèle et qui est le langage Cypher.

2.7 Conclusion

Dans ce chapitre, nous nous sommes focalisés sur l'aspect données des processus métiers. Dans un premier temps, nous avons mis en évidence l'importance des données d'exécution puis nous avons exploré les différentes techniques de leurs stockage et de leur interrogation.

Dans le prochain chapitre, on positionnera notre problème de recherche par rapport à l'état de l'art du domaine, puis on explorera les travaux de recherche qui ont traité notre problématique.

CHAPITRE 3

Travaux connexes

3.1 Introduction

Ces dernières années, beaucoup de travaux de recherche du domaine informatique ont traité les questions de modélisation, d'analyse, d'évolution et d'optimisation des processus métier. Cependant, on remarque que ces travaux se sont focalisés beaucoup plus sur les données relatives aux modèles des processus métiers sans donner une grande importance aux données exprimant les exécutions réelles des instances associées aux processus métiers.

Dans ce chapitre on va explorer les travaux qui ont, plus ou moins, abordé le problème de la gestion des données des processus métiers et on va mettre l'accent, particulièrement, sur les données d'exécution.

Avant d'étudier les travaux existants, on commence tout d'abord par exposer le problème de recherche traité dans ce projet de fin d'études.

3.2 Problématique

Les processus métiers constituent le pilier des systèmes d'informations modernes. En effet, ils renforcent les systèmes d'information conscients des processus métier **P.A.I.S**[1] qui sont la nouvelle tendance pour l'urbanisation et l'intégration des **S.I.** Cela est dû, fondamentalement, au fait que les processus métiers intègrent, à la fois les deux dimension de l'organisation, à savoir :

— La dimension statique représentant (**les données**) ;

— La dimension dynamique relative aux (**les traitements**).

D'autre part, lors de leur invocation par les clients, les **PM** génèrent différentes données [29]. Dans notre travail, on s'intéresse particulièrement aux données d'exécution vue leur importance.

Puisque les données d'exécutions revêtent une importance particulière, donc leur exploitation exige des mécanismes et des outils spécifiques. A titre d'exemples, les questions suivantes sont des préoccupations majeurs pour les gestionnaires de processus et les décideurs de l'organisation.

Quels sont les niveaux atteints par les instances et quels sont les chemins historiques parcourus ?

— Comment formuler des requêtes de mise à jour complexes, ou plus précisément, comment peut-on substituer certaines traces des instances ? Autrement, peut-on remplacer une expression de chemin par une autre, dans le cas ou le processus devient coûteux, obsolète ou bien en cas de pannes ?

1. Process Aware Information systems

— Lors des évolutions, suite à des changements de la logique métier dû à des modifications de la règlementation ou en raison des changements des lois, que faire pour localiser les instances concernées et comment procéder à leur actualisation ?

La réponse à ce type de questions apporte une valeur ajoutée pour toute l'entreprise et constitue un facteur de performance et de compétitivité.

La figure 3.2 ci-dessous schématise la problématique traitée dans ce mémoire.

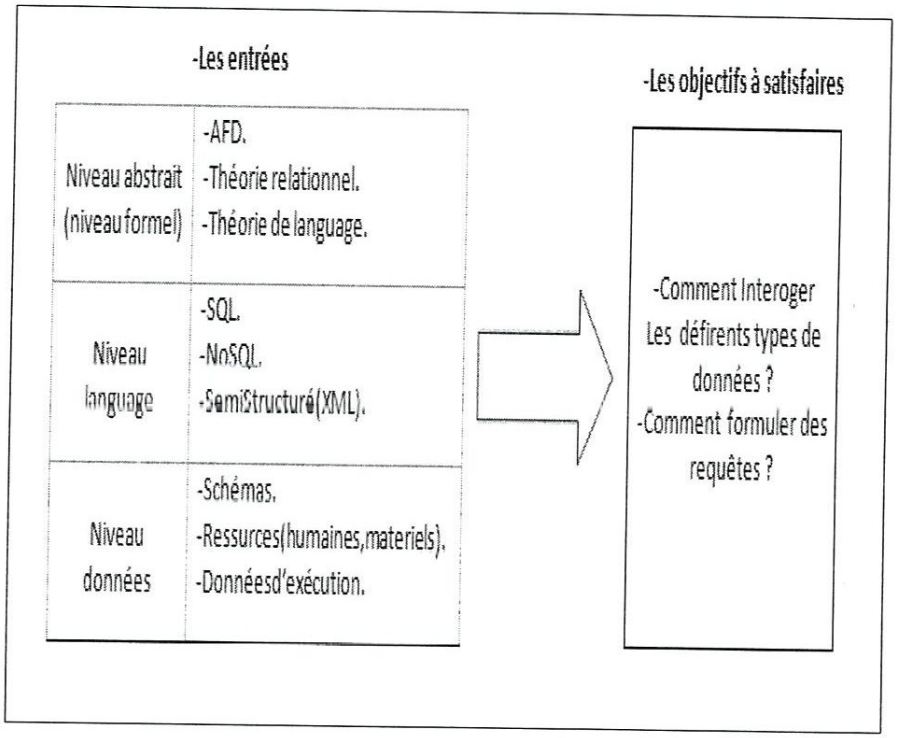

FIGURE 3.1 – Schématisation de la problématique.

Dans ce qui suit, on va explorer les travaux connexes ayant traiter cette question. Notre analyse des travaux existants sera abordée suivant trois dimensions complémentaires. On commence par l'étude des travaux de la recherche théorique, puis on exposera les langages associés au domaine de la gestion des processus métiers. Et enfin, on terminera par un survol de quelques outils et suites logicielles existants sur le marché industriel.

3.3 Étude des travaux de la recherche théorique

Les travaux de la recherche académique qui ont traité la question des données dans les processus métiers peuvent être répartis en trois catégories, à savoir :

1. La première famille s'intéresse uniquement aux données du niveau modèle, ou les données du schéma (flux d'activités, contraintes sur l'ordonnancement, ...).
2. La deuxième famille de travaux traite les données des processus métiers, proprement dites.

3. Une troisième catégorie de travaux combine les deux approches.

Ci-après, ces différents travaux seront analysés et discutés.

3.3.1 Analyse au niveau modèle

Beaucoup de travaux se sont intéressés à l'analyse, à la recherche et à la vérification des données relatives aux spécifications des processus métiers [1],[7],[8]. Avec l'accroissement des modèles des processus métiers publiés et accessibles par les différents intervenants, différentes techniques ont été proposées pour la recherche de certaines fonctionnalités répondant à des besoins spécifiques des utilisateurs, tout en déployant des mécanismes de recherche adéquats.

Dans [30], les auteurs font un état de l'art sur l'interrogation des bases de données contenant des modèles des processus métiers et dressent les défis associés. Le fondement de ce travail est basé sur la comparaison des requêtes sur les modèles de processus métier avec les requêtes sur les graphes. Pour opérer cette comparaison, les auteurs identifient trois critères pertinents, à savoir : les structures de graphes, la sémantique du comportement (*les activités*) et la sémantiques des opérations réalisées. En conclusion, les auteurs arrivent à la distinction entre les types de requêtes de recherche dans les modèles de processus métier en 5 types :

1. Requêtes exactes basées sur la structure du graphe ;
2. Requêtes de similitude basées sur la structure du graphe ;
3. Requêtes exactes basées sur la sémantique du comportement ;
4. Requêtes de similitude basées sur la sémantique du comportement ;
5. Requête basée sur la sémantique des opérations.

Dans [31], les auteurs on proposé un cadre basé sur des graphe pour interroger et réutiliser des activités des processus métiers. Le modèle proposé offre aux utilisateurs la possibilité d'obtenir, non seulement, un modèle de processus parfaitement adapté à leurs requêtes, mais aussi un modèle avec une grande similitude. Le but de ce travail est de présenter un cadre évolutif pour aider les concepteurs de processus métiers à réaliser une tâche de modélisation efficace, en interrogeant et en réutilisant les modèles de processus existants. Les apports de cette contribution se résument aux points suivants.

1. Le cadre proposé est basé sur une nouvelle requête intuitive et sur un langage visuel pour les modèles de processus métier. Il permet aux utilisateurs de définir leurs propres modèles de processus métier et les requêtes qui les manipulent en utilisant un ensemble très similaire de notations.
2. Le cadre est amélioré avec une extension de requêtes sémantiques composées. Il emploie une dimension ontologique dans le processus de correspondance de requêtes et aborde le problème de l'application terminologiques différentes lors de la modélisation des processus ontologiques.
3. Pour atteindre une performance efficace, le processeur de requêtes utilise une infrastructure d'indexation robuste garantie par le système de gestion des bases de données relationnelles. Il utilise une nouvelle base de décomposition et un mécanisme de traduction sensible à la sélectivité des requêtes basées sur des graphes dans des scripts SQL.

4. L'architecture du cadre est conçue de manière très flexible.

Les travaux de V.D. Aalst [32] ont traité de manière consistante la question de fouille des processus métiers et de la reconstruction des modèles (*schémas*) à partir des données d'exécution contenues dans les fichiers logs. L'auteur propose différents algorithmes $\alpha, \alpha^{+}, \alpha^{++}$ permettant de rechercher les différents états et transitions contenus dans les fichiers d'exécutions. Ces éléments seront représentés par des Réseaux de Petri qui expriment le modèle du processus métier.

3.3.2 Analyse au niveau données

Les travaux d'analyse des processus métiers par rapport au niveau données s'intéressent uniquement aux données générées lors de l'exécution des processus métiers par les différentes instances. Ces données sont souvent stockées dans des fichiers ou dans des bases de données adéquates.

Dans [33], les auteurs font une comparaison entre les données des processus métiers et les bases de données. Ils formalisent des modèles de données et formulent un langage graphique pour traiter séparément les bases de données et représenter les données sous forme hiérarchique. Ces travaux mènent vers un nouveau cadre conceptuel prometteur pour une approche intégrée dans la modélisation des processus métiers et des données persistantes dans les systèmes d'entreprise.

Dans [29], les auteurs développent un cadre qui organise systématiquement les processus métier et les données liées, afin d'organiser l'évaluation de plusieurs langages de modélisation. Le cadre de l'évaluation consiste en 23 critères répartis en quatre groupes :

— fonction de modélisation de processus.

— capacité de modélisation de données.

modélisation des flux et des données.

— exécution sémantique des processus.

Le travail effectué dans [34] aborde pour la première fois l'étude et la gestion des types d'informations pour une classe importante de données semi-structurées. Les données traitées sont relatives à des traces d'exécutions. Plus précisément, les auteurs examinent la possibilité d'inférer les types des données et la vérification de ces types pour les requêtes sur les traces d'exécution du partenaire. Les traces d'exécution sont modélisées sous forme d'un graphe acyclique dirigé imbriqué, en outre ils ont défini et caractérisé trois classe de trace : NAIF, SEMI-NAÏF et SÉLECTIF.

D'autre part, dans [35], la modélisation des **PM** est faite de façon à permettre de faciliter et de comprendre le fonctionnement des systèmes de données correspondants. Du fait que les différentes étapes des processus nécessitent des données d'entrée et génèrent de nouvelles données de sortie, les **BPMS** assurent les fonction de modélisation, configuration et exécution des **PM**. Comme les données dans les **PM** sont issues de ressources multiples, avec une variété de formats, elles sont générées à haute vitesse. Cela exige que l'infrastructure qui les supporte offre des mécanismes qui rendent ces données prêtes pour l'analyse nécessaire. Par ailleurs, les aspects de flexibilité et de dynamique de l'exécution du processus vont au delà des techniques de stockage classiques (*BDDR, XML*). Ainsi, les bases de données **NoSQL** doivent être prise en

considération. Donc l'idée est de combiner les processus métiers avec les bases de données **NoSQL**. Enfin, dans cet article les auteurs ont proposé l'adoption d'un schéma de base de données **NoSQL** avec son langage *MongoDB* pour modéliser les données d'apprentissage dans le contexte de *MOOC*.

3.3.3 Analyse des approches mixtes < Modèles-Données >

Les auteurs dans [36] développent un langage pour la spécification du comportement d'un nouveau concept appelé : **artefacts**. Ce concept est basé sur la combinaison des données et des comportements dans un seul format. Mais ce modèle n'est pas évident pour deux raisons principales. Tout d'abord, les attributs des artefacts et les variables de descriptions des processus peuvent être illimitées. La seconde raison est que même le nombre de ces attributs et les variables sont liées. Cela engendre une situation dans laquelle les descriptions de processus peuvent lire des valeurs externes et créer un nombre infini de nouvelles valeurs lors du calcul.

Un autre travail similaire est présenté dans [37]. Dans le but d'apporter des améliorations à la description des processus métiers, les auteurs proposent une analyse qui prend en considération le niveau d'abstraction relatif aux données. Mais, le problème de la porté des changements rend difficile la prise en compte des changements fréquents dans les sources de données de processus. Afin de résoudre ces problèmes, les auteurs développent une solution générale et réutilisable pour le stockage des données des processus. La solution proposée est applicable à la plupart ou à la totalité des processus d'une société (*une solution générique*). Ils ont vérifié par l'expérience que la conception de cette solution générique est non trivial mais elle est possible.

Dans [38] les auteurs décrivent une approche automatique pour la construction d'un schéma flux de travail (**Workflows**) qui satisfait une condition spécifiée par un objectif. Les auteurs utilisent un modèle de workflows basé sur des artefacts métiers, qui représentent des clés métiers commerciales. Ces construction (*artifacts*) incluent à la fois les données pertinentes pour l'entreprise et leurs modèles de représentation. Le cadre fournit par les auteurs, est un cadre générale pour l'étude des problèmes en dessous :

— Chaque exécution est soit complète, soit incomplète. Une exécution complète satisfait l'objectif attendu.

— L'étude du problème complémentaire dans lequel la gestion des exceptions est utilisée pour traiter les exécutions.

Dans ce travail, les auteurs montrent une relation étroite entre les systèmes Workflows qui sont spécifiés en utilisant des formalismes logiques et leurs capacités à construire des schémas maximaux avec les propriétés désirées.

Après avoir examiner les travaux de la recherche théorique, on va à présent s'intéresser aux langages de modélisation des PM par l'étude de leurs capacités à gérer les données et leurs aptitudes à prendre en charge cet aspect.

3.4 Aspect données dans les langages des processus

Certains langages ont émergé les dernières décennies et ont prouvé leurs efficacités dans le domaine des **PM**. Ces langages sont dédiés aussi bien pour la modélisation que pour l'exécution des processus métiers. Généralement, ils ne sont pas utilisés directement dans les phases de conceptions, mais ils exploitent le format d'échange standard XML, pour interagir avec des systèmes hôtes, comme les services Web ou les systèmes BPM. Ainsi, ils disposent d'un format d'échange natif. Cependant, aucun d'entre eux ne propose une notation graphique standardisée [39].

Dans ce qui suit, on s'intéressera à quatre langages, considérés comme les standards les plus représentatifs qui sont : **XPDL, BPML, YAWL** et **BPEL**. Pour chaque langage, on discutera de sa capacité à gérer les données des processus métiers.

3.4.1 BPEL

BPEL (*Business Process Execution Language*) [40] définit une notation pour spécifier le comportement des processus métiers basés sur les services Web. Les processus métiers peuvent être décrits suivant les deux dimensions complémentaires suivantes.

1. Une partie BPEL abstract qui décrit les protocoles métiers, en utilisant des descriptions qui spécifient le comportement d'échange de messages mutuellement visible de chacune des parties impliquées dans le protocole, sans révéler leur comportement interne.
2. Les processus métiers exécutables modélisent le comportement réel d'un participant dans une interaction, par la manipulation des structures de contrôle.

Un programme *BPEL* est utilisé pour modéliser le comportement des processus exécutables et abstraits et peut comprendre : Il comprend :

- Le séquencement des activités de processus, en particulier, des interactions des services Web ;
- Les structures de corrélation des messages et des instances de processus ;
- Les comportements de reprise en cas de pannes et de conditions exceptionnelles ;
- Les relations bilatérales basées sur le service Web entre les rôles de processus.

Les fonctionnalités de traitement des données de **BPEL** facilitent la gestion des interactions statiques en offrant la possibilité de suivre l'état interne de chaque instance de processus métier.

3.4.2 BPML

(Business Process Modeling Language) [41], le langage *BPML* est un métalangage basé sur un langage *XML*, développé par *BPMI* comme moyen de modéliser les processus métier, tout comme *XML*, est un métalangage avec la capacité de modéliser les données d'entreprise *BPML*, et comprend des spécifications pour les transactions et les transactions compensatoires, le flux de données, les messages et les événements planifiés, les règles de gestion, les rôles de sécurité et les exceptions. *BPMI* a identifié trois aspects cruciaux de la capacité *BPML*, comme il sera utilisé pour des applications stratégiques, il doit prendre en charge les transactions distribuées synchrones et

asynchrones. Comme il va modéliser les processus métier déployés sur Internet, il doit offrir des mécanismes de sécurité fiables, Et comme il sera utilisé dans tous les environnements de développement intégrés, il doit englober des fonctionnalités de gestion de projet.

3.4.3 XPDL

Le *.xpdl* extension de fichier est principalement utilisé par TIBCO iProcess. Il est utilisé pour faire référence à des fichiers qui sont créés avec l'utilisation de ce logiciel, une application qui traite des processus d'affaires. Cette application aide les entreprises et les organisations professionnelles organisent tous leurs processus d'affaires afin qu'ils puissent facilement accomplir leurs tâches et bien gérer leurs entreprises. Surtout, codes $XPDL$ ainsi que le XML, sont ce que ces fichiers sont composés de [42]. L'objectif de $XPDL$ est de stocker et d'échanger le diagramme de processus, de permettre à un outil de modéliser un diagramme de processus et à un autre de lire le diagramme et de le modifier, un autre à" exécuter "le modèle de processus sur un moteur de MPM compatible $XPDL$. bientôt. Pour cette raison, $XPDL$ n'est pas un langage de programmation exécutable comme $BPEL$, mais plus particulièrement un format de conception de processus qui représente littéralement le schéma de la définition de processus [43].

3.4.4 YAWL

Encore un autre langage de Flux de travail, $YAWL$[2] est un système BPM/Flux de travail, basé sur un langage de modélisation concis et puissant, qui gère des transformations de données complexes et une intégration complète avec les ressources organisationnelles et les services Web externes. $YAWL$ propose [44] :

- Le langage de spécification de processus le plus puissant pour capturer les dépendances de flux de contrôle et les besoins en ressources.
- Traitement natif des données à l'aide de XML Schéma, XPath et XQuery.
- Une base formelle qui rend ses spécifications sans ambiguïté et permet une vérification automatisée.
- Une architecture orientée services offrant un environnement pouvant être facilement adapté à des besoins spécifiques.

3.4.5 Synthèse sur les langages des processus métiers

De ce qui précède, il apparait que la grande partie des langages des processus métiers a abordé la gestion des données d'un point de vue modélisation, par la fourniture d'un ensemble de primitives pour la spécification des structures définissant le schéma du processus à modéliser. Par contre, la dimension données a été traitée de façon superficielle, ou certaines types de données avancées ont été utilisés pour le stockage des éléments descriptifs du schéma. Quand aux données d'exécution elles sont gérées, par des interfaces qui permettent d'accéder aux bases de données associées.

2. Yet Another Workflow Language

Langage Etudié	Caractéristiques
BPEL	— Langage d'exécution. — Ne définit pas le diagramme graphique. — Fournit une définition de l'orchestration des services web. — Est un langage basé sur *XML*, permettant de créer des processus d'entreprise. — Il offre la possibilité d'interagir avec les données des services web. — Il offre la possibilité de mettre en place des processus complexes. — Il permet de manipuler les variables d'un processus et de les initialiser.
BPML	— Un méta-langage pour la modélisation de processus tel que *XML*. — Un méta-langage pour la modélisation de données d'entreprises. — N'est pas un langage de processus complet. — Il s'agit d'un pur moteur de traitement simultané et distribué. — Il offre le scénario complet d'échange et de traitement de données.
XPDL	— Permet de stocker et d'échanger le diagramme de processus. — N'est pas un langage de programmation exécutable comme *BPEL*. — Représente le schéma de définition de processus. — Il permet de définir les processus métier à l'aide de *XML* et aussi de les organiser. — Il est extensible, afin de permettre à chaque outil de stocker ses données et de les manipulées convenablement.
YAWL	— Offre un support complet pour les modèles de flux de contrôle. — *YAWL* a une base formelle appropriée. — *YAWL* offre un support unique pour la gestion des exceptions. — Gère les informations de chaque processus et récolte les données nécessaires. — Il sert à décrire les données nécessaire à répartir entre les différents acteurs d'un processus.

TABLE 3.1 – Gestion des données par les langages des processus métiers

3.5 Exploration de quelques outils industriels

A présent on va se focaliser sur les suites logicielles fournies par différents constructeurs. Pour chaque outils dans le domaine industriel on abordera ses caractéristiques, afin de voir quelles sont les possibilités offertes par les outils commerciaux dans le cadre de l'analyse et la gestion des données des processus métiers et leur interrogation.

Quatre outils les plus utilisés sont présentés. Trois propriétaire et un open source.

3.5.1 IBM web-sphère application serveur

Web-sphere est une marque du groupe *IBM* (*International Business Machines*). Il s'agit d'un serveur d'applications Java rapide et très performant qui peut créer, exécuter, intégrer, sécuriser et gérer des applications Web dynamiques des processus métiers, sur site, hors lieux, ou dans les deux environnements à la fois. Conçu dans un souci de célérité et de souplesse, il présente aussi un choix de modèles de programmation à normes ouvertes pour pousser au maximum la productivité du développeur[45].

Web-sphère application est un site web qui réagit aux informations saisies par l'utilisateur et qui sont réellement interactives. Cet outil offre une gamme d'environnement d'exécution pour mieux s'adapter aux divers besoins métier.

D'une manière générale, les outils présentés par *IBM* aident les développeurs à concevoir, développer, assembler, tester et déployer des services web. Ils intègrent des outils de test et d'analyse, qui permettent d'identifier et de corriger les problèmes d'évolution. Cependant, des insuffisances sont constatées pour la gestion de la migration des instances actives. En effet, cette dernière est laissée à la charge de l'utilisateur qui est contraint de prendre des décisions, au cas par cas, pour trancher sur la poursuite des activités quand une nouvelle version du processus métier est déployée.

Caractéristiques et avantages de web-sphère application serveur : Le web-sphere Application Serveur utilise des standards ouverts tels que Java EE, XML, et les services Web. Il fonctionne avec de nombreux serveurs Web, HTTP Serveur, Microsoft Internet Information Services (IIS). et parmi ses avantages on a :[46]

- Support pour les environnements multi-cloud.
- Fonctions de gestion intelligente.
- Sécurité et contrôle améliorés.
- Meilleure productivité des développeurs.

3.5.2 Suite Oracle

Les outils fournis par Oracle, avec leur différentes versions : Oracle Business Process Management Suite, Oracle BPEL Process Manager et BPEL Designer sont des suites logicielles intégrées conçues pour modéliser, automatiser, gérer, simuler, optimiser et exécuter les processus métiers, les systèmes et les applications d'une organisation. Ces suites améliorent l'efficacité et la qualité des processus métiers en renforçant leur utilisation et leurs débits.

A titre d'illustration, l'outil Oracle BPEL Process Manager permet la réalisation de l'orchestration des processus métiers et des services web, par la mise en place d'un

mécanisme d'intégration des éléments existants dans le système. Dans de tels environnements, les services informatiques peuvent déployer des processus opérationnels critiques gérant d'importants volumes, tout en s'appuyant sur les services de l'infrastructure existante [47]. Dans la version BPEL Designer, les utilisateurs peuvent modéliser via une interface graphique leurs processus métiers, en utilisant la spécification du langage BPEL [48]. Ceci garantit la portabilité des définitions des processus métiers. De plus, les développeurs peuvent consulter et modifier directement le code source BPEL. Cet environnement est l'outil de référence des développeurs pour l'implémentation des processus métiers [47].

3.5.3 NetWeaver

En sa qualité d'intégrateur d'applications d'entreprise, SAP[3] a élaboré une série d'outils pour la gestion du cycle de vie des processus métiers.

La plateforme SAP NetWeaver facilite l'intégration des outils en dépassant les frontières organisationnelles et technologiques, et elle réduit le coût de développement et de déploiement des processus métiers [49].

L'environnement SAP Netweaver Business Process Management procure aux spécialistes de la gestion et de l'informatique un environnement partagé pour concevoir, modéliser et exécuter des processus de gestion nouveaux ou adaptés, sans créer de code. Il facilite l'incorporation de règles de gestion claires dans les processus métiers. La modélisation des processus métiers avec les vues d'implémentation SAP assure la correspondance avec la stratégie de l'entreprise, ainsi que le support des objectifs de l'entreprise [49].

L'outil Mega SolMan présente une interface bidirectionnelle qui assure la synchronisation des modèles SAP avec les processus métiers. Cet outil réduit le temps et le coût d'un projet SAP, en identifiant en amont les besoins réels de l'organisation [50].

Il est constaté que dans les environnements concurrentiels actuels, où l'agilité et l'adaptabilité sont des impératifs, les solutions proposées par SAP ne sont pas en capacité d'apporter une plus value pour la prise en charge des changements. En effet, les entreprises modernes ont besoin d'une plateforme unique et flexible qui intègre, en plus de l'ensemble des collaborateurs et des informations manipulées par les processus métiers, les éventuelles évolutions et la gestion de la migration des instances de ces processus métiers.

3.5.4 Bonita BPM

Bonita BPM Community est un outil de gestion des processus métiers (BPM) *Open Source* qui permet de modéliser les différentes étapes d'un processus métier, avec la prise en charge des tâches automatiques, des actions manuelles et des formulaires pour la saisie ou la validation des informations. L'outil intègre, aussi, la gestion des différents groupes et les rôles des utilisateurs. Les utilisateurs finaux l'exploitent pour créer, suivre et réaliser les différentes tâches, activités et évènements liés à l'automatisation de leur processus métiers [51].

3. **SAP** : Systems, Application and Products for data processing

Bonita permet de modéliser les différentes étapes d'un processus métier, avec la prise en charge des taches automatiques, des actions manuelles et des formulaires pour la saisie ou la validation des informations. Cette solution logicielle s'articule autour de trois composants :

- Un studio de modélisation de processus "Bonita Studio", qui tente de relever le défi du standard $BPMN$ avec une solution graphique "simple et intuitive", reprenant les concepts d'un tableau blanc.
- Un moteur d'exécution des processus. Celui-ci est assez flexible pour s'adapter à diverses architectures de systèmes d'information et suffisamment extensible pour intégrer de nouveaux services ou standards qui peuvent émerger dans le BPM.
- Une interface utilisateur de contrôle permettant de lancer l'exécution de processus et de vérifier sous forme de boîte aux lettres de réception le résultat des processus.

Fonctionnalités de Bonita BPM : L'interface de bonita permet de concevoir, exécuter, optimiser les processus métiers tout en garantissant un niveau de contrôle primordial. Elle vise à la continuité de votre service tout en facilitant la tâche des salariés [52]. La solution proposée par Bonitasoft permet un large panel de fonctionnalités :

- Amélioration continue des processus
- Outils de développement " code bas " accessible soutenant la transformation digitale.

  Adaptative case management (Optimisation de la gestion des taches).
- Automatisation de l'ensemble des processus et applications clients.
- Automatisation des tâches répétitives en grand volume.

3.5.5 Synthèse sur les outils industrielles

L'étude des outils industriels fait ressortir un manque en matière de prise en charge des données des processus métiers. Généralement, les environnements **BPM** stockent les données issues des exécutions des PM dans des bases de données relationnelles. Des outils de navigation de type SQL sont offerts et des API pour la connexion avec les bases de données sont mis à la disposition des développeurs qui utilisent des environnements de programmation avancés (Java).

3.6 Conclusion

Dans ce chapitre nous avons commencé par l'exposé de notre problématique, puis nous avons réalisé une étude des travaux connexes ayant traité le problème de la gestion et de l'interrogation des données des processus métiers. Une attention particulière a été accordée aux environnements logiciels et à leur capacités de gérer les données.

La deuxième partie de ce rapport est consacrée à notre contribution.

Deuxième partie

Contribution

CHAPITRE 4

Conception de l'approche d'interrogation

4.1 Introduction

Dans le chapitre précédent nous avons posé la problématique et nous l'avons positionnée par rapport aux travaux connexes.
Dans ce chapitre, nous présentons notre approche pour l'interrogation des données d'exécution des processus métiers. Nous commençons par la spécification du modèle formel de processus métiers qui sera utilisé dans notre approche, puis nous exposons l'ensemble des concepts qui lui sont associés. Après, nous présentons la modélisation de l'approche, nous détaillerons le cadre formel qui la supporte et nous l'illustrons par des exemples réels de requêtes d'interrogation.

4.2 Choix du modèle de processus métier

Dans cette section, nous présentons le modèle formel supportant la représentation des processus métiers, nous l'illustrons avec un exemple réel, puis nous justifions le choix d'un tel modèle.

4.2.1 Spécification du modèle de processus métier

Dans notre approche, nous utilisons les automates d'états finis déterministe **AFD** [1] pour représenter les processus métiers. Une version simplifiée du processus métier qui permet de décrire, essentiellement, les contraintes d'ordre régissant l'exécution des ***activités*** *(abstraites)* fournies par processus est adoptée. Un tel modèle est très adéquat pour répondre à notre problématique, car il permet de décrire le flux d'activités manipulées par un utilisateur *(échange de messages, exécution d'une tâche, état atteint par une instance ...)*. Ainsi, les données générées et manipulée durant l'exécution des instances sont prises en compte par un tel modèle, ce qui permettra plus tard d'analyser et d'interroger les données d'exécution des instances ayant invoqué le processus.

Nous donnons ci-dessous la spécification formelle d'un tel modèle.

Définition 4.1 ***Modèle de processus métier*** *[13]*

Un processus métier est un tuple $\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R})$*, tels que :*

— Q *est un ensemble fini d'états.*

1. ou **FSM** : **F**inite **S**tate **M**achine

— $q_0 \in Q$ *est l'état initial du processus.*

— $\mathcal{F} \subseteq Q$ *est l'ensemble des états finaux du processus.*

— $\mathcal{M}$ *est un ensemble fini d'activités abstraites (tâches).*

— $\mathcal{R} \subseteq Q \times Q \times \mathcal{M}$ *est une relation de transition, Chaque élément* $(q, q', m) \in \mathcal{R}$ *représente une transition d'un état source* q *vers un autre état cible* q'*, suite à l'exécution de l'activité* m*.*

Ainsi, un processus métier (*plus simplement,* ***protocole métier***) est formellement décrit par une machine à états finis déterministe **(AFD)**, où les états représentent les différentes phases qu'une instance d'un processus peut parcourir pendant son exécution alors que les transitions représentent les activités qu'un processus peut réaliser [13].

4.2.2 Exemple d'un processus métier modélisé par un AFD

La figure 4.1, montre un exemple simple d'un processus métier. Ce processus représente la logique métier imposée par le fournisseur de services de Réservation en ligne d'une chambre d'hôtel. Le processus exprime l'ensemble des activités décrivant les étapes de la procédure de réservation. Par contre, les états sont les phases par lesquelles transitent les instances qui invoquent le processus.

Exemple 4.1 *Processu métier de réservation en ligne d'une chambre Le fonctionnement du processus est le suivant.*

La réservation de la chambre commence, une fois que l'utilisateur invoque le processus et génère une instance de ce processus (état début*). La première opération est "Choisir Ville (CV)" qui permet de passer de l'état "Début" à l'état "Ville sélectionnée". Puis, cette instance passe à l'état "Hôtel sélectionnée" lorsque le client invoque l'opération Choisir "Choisir hôtel)CH)". Ainsi, et suivant cette logique de fonctionnement, l'invocation de chaque opération permet de passer d'un état de l'automate à un autre jusqu'à atteindre un des états finaux Chambre réservée ou Réservation annulée.*

A noter l'existence de différentes structures dans la spécification de l'automate, à savoir des boucles, plusieurs chemins d'exécutions, des sous-protocoles, ... D'autre part, à un moment donnée de la vie d'un processus, il peut exister des milliers, voir des millions d'instances de ce processus qui sont en cours d'exécution en même temps, et dont chacune à atteint un niveau d'exécution différent. Ce qui sous-entends que chaque instances du processus en question est dotée de ses propres données d'exécutions.

4.2.3 Motivations pour le choix du modèle de processus

Nous avons déjà discuté dans le premier chapitre qu'il existe plusieurs modèles de représentation des processus métiers, et nous avons mis en évidence que chaque modèle prend en charge différents type de contraintes et de d'abstractions à modéliser (*contraintes d'ordre sur les activités, contraintes temporelles, ...*) (*voir section 1.4*).

Dans notre approche, nous avons opté pour les automates d'états finis déterministes **AFD** pour modéliser les processus métiers. Ce choix n'est pas fortuit, mais il est basé sur un ensemble de motivations qui sont les suivantes.

— Les **AFD** sont des outils simples et populaires.

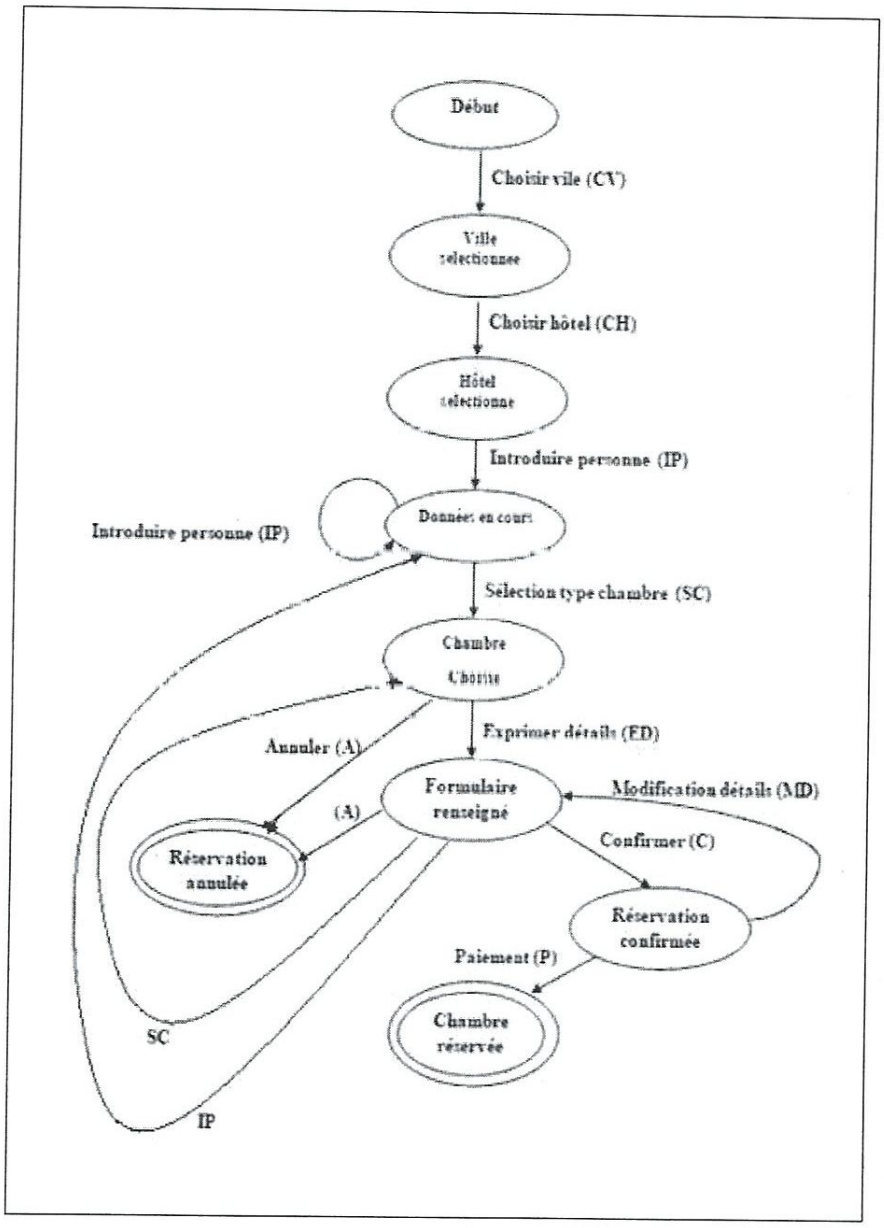

FIGURE 4.1 – Processus de réservation modélisé par un AFD.

— Les **AFD** permettent la vérification et la validation des modèles. En plus, il existe toute une panoplie d'outils logiciels pour leur vérification.

— Les **AFD** sont largement utilisés dans les systèmes dynamiques et réactifs aussi bien que les systèmes de contrôle.

4.3 Concepts liés au modèle de processus métiers

Nous introduisons dans ce qui suit, certains concepts liés aux **PM** et utiles dans notre contexte, tout en spécifiant leur définition formelle. On commence tout d'abord par l'introduction de la notion d'instance d'exécution qui caractérise l'exécution d'un **PM**, puis on définit le chemin d'exécution et la trace d'exécution. Enfin, les notions de sous-protocole et expression régulière seront présentées. Chaque concept présenté sera illustré par un exemple extrait de la figure 4.1.

4.3.1 Instance d'exécution

Une instance d'exécution d'un processus $\mathcal{P}$ est associéée à chaque invocation de $\mathcal{P}$. Elle exprime les activités réalisées depuis le démarrage et dure une période limitée avant d'être archivée par le système. Donc, en plus des instances d'activités elle comprend aussi d'autre données relatives au temps et aux ressources. Toute instance doit être **conforme** au modèle d'activités figurant dans le modèle de processus [12].

Définition 4.2 *Une instance d'exécution I d'un processus métier correspond à une invocation de ce même processus par un utilisateur spécifique. Ce dernier peut être un être humain, une machine ou un autre processus métier.[13]*

Exemple 4.2 *Le processus de réservation de la figure 4.1 peut être invoqué par plusieurs instances émanant de différents utilisateurs désirant réserver des chambres d'hôtels. Ci-dessous, on expose un échantillon d'instances (I) à différents états d'exécution.*

— I_1 *: exécutée par l'utilisateur hamici ; à :* $20H00$ *; qui a atteint l'état :* Chambre choisie *; son exécution a duré :* $03:00$ *; et son état est* active.

— I_2 *: exécutée par client2 ; à :* $10H00$ *; qui a atteint l'état :* Hôtel sélectionné *; son exécution à durée :* $03:19$ *; et son état est aussi :* active.

— I_3 *: exécutée par client3 ; à :* $10:25$ *; qui a atteint l'état :*Réservation annulée *; son exécution a durée :* $00:50$ *; et son état actuel est :* terminée.

— I_4 *: exécutée par client4 ; à :* $14:38$ *; qui a atteint l'état :* Réservation confirmée *; son exécution a durée :*$00:19$ *; et son état est :* active.

4.3.2 Chemins d'exécution

Chaque invocation du **PM** par un utilisateur, la machine, ou un autre processus métier, permet de lancer l'exécution de la première activité, en partant de l'état initial Q_0. Ensuite, elle va exécuter une séquence d'activités menant de Q_0 vers Q_k. Ce passage entre le différentes activités peut être réalisé par le parcours de chemins possibles offerts par la spécification du **PM**.

Dans ce qui suit, on va exprimer de manière formelle la notion de chemin d'exécution emprunté par une instance de processus.

Définition 4.3 *Un chemin d'exécution c d'un processus métier $\mathcal{P}$ exprime une alternance entre les états parcourus et les activités exécutées par l'instance I, lors de son interaction avec le processus métier.[13]*

Exemple 4.3 *Les chemins d'exécution associés aux instances de l'exemple 4.2 sont les suivantes :*

- c_1 = *Début.CV.Ville sélectionnée.CH.Hôtel sélectionné.IP.Données en cours.IP.Données en cours.IP.Données en cours.SC.Chambre choisie.* [2]
- c_2 = *Début.CV.Ville sélectionnée.CH.Hôtel sélectionné.*
- c_3 = *Début.CV.Ville sélectionnée.CH.Hôtel sélectionné.IP.Données en cours.SC.Chambre choisie.ED.Formulaire renseigné.A.*
- c_4 = *Début.CV.Ville sélectionnée.CH.Hôtel sélectionné.IP.Données en cours.IP.Données en cours.SC.Chambre choisie.ED.Formulaire renseigné.C.Réservation confirmée.*

On remarque que parmi ces instances, certaines ont terminé leur exécution alors que d'autres sont encore en cours d'exécution. Les instances terminées ont parcouru des chemins d'exécution **complets** alors que celle en cours ont des chemins d'exécution **incomplets**.

4.3.3 Trace d'exécution

Définition 4.4 *Une trace d'exécution $\delta(I)$ d'une instance I correspond à l'historique ou la séquence d'activités exécutées par I lors de son interaction avec le processus métier, du début de l'invocation de ce processus métier jusqu'à son état actuel [13].*

Concrètement, les traces d'exécution sont obtenues par la suppression des noms des états des chemins d'exécution associées.

Exemple 4.4 *On garde toujours le même sous-ensemble de traces de l'exemple 4.1, et on opère l'extraction des traces des instances, en tenant compte uniquement des noms des activités.*

- $\delta(1)$ = *CV.CH.IP.IP.IP.SC.* ;
- $\delta(2)$= *CV.CH.IP.IP.SC.* ;
- $\delta(3)$ = *CV.CH.IP.IP.IP.SC.ED.ED.A.* ;
- $\delta(4)$ = *CV.CH.IP.IP.IP.SC.ED.ED.IP.SC.ED.C.*

A signaler que dans la réalité, les traces d'exécution ne contiennent pas uniquement les noms des activités, mais elles sont aussi dotées d'autres données qui les caractérisent. A titre d'exemple, on peut citer les attributs suivants.

- InsID : Identificateur de l'instance. C'est un numéro attribué par le système d'une manière automatique.

2. *Le symbole point "." est utilisé pour exprimer la succession d'activités*

— Le nom de l'utilisateur qui exprime l'identité de celui qui a invoqué l'instance.

— Les ressources déployées et les ressources exigées par l'instance pour pouvoir continuer son exécution, telles que le besoin d'une imprimante, du réseau, ...

— Le temps de lancement de l'instance et la durée écoulée depuis son démarrage.

— L'état actuel de l'instance (*terminée ou active*).

4.3.4 Expression régulière

Les expressions régulières sont des formats exprimés dans une syntaxe très précise et très rigoureuse, afin de pouvoir manipuler des séquences de symboles relatives à des langages ou des automates. Dans ce qui suit, on présente cette notion formellement.

Définition 4.5 *Soit* $\mathcal{A}=\{ a_1, a_2, \ldots, a_n\}$ *un alphabet de symboles représentant les activités du* ***PM****, auquel on ajoute les caractères spéciaux suivants :*

— "." *: est l'opérateur de concaténation.*

— $*$ *: est l'opérateur star-Kleene [53] qui exprime la répétition d'un ou plusieurs caractères.*

— "(" *et* ")" *: sont utilisées pour agréger un opérateur et délimiter l'expression ainsi que pour l'imbrication des expressions.*

Une expression régulière r sur $\mathcal{A}$ *est une formule bien formée reconnue par l'automate qui représente le* ***PM*** *qui réalisant l'ensemble des activités* $a_i \in \mathcal{M}$; *(pour* $i = 1 \ldots, n$*).*

En se basant sur la définition précédente, une expression régulière est une chaîne de caractères ou mots. Elle est construite à partir d'un alphabet, et elle utilise des opérateurs d'articulation ainsi que les caractères spéciaux.

Ainsi le pattern "bonjour" représente simplement le mot "bonjour". Il est ensuite possible d'ajouter des caractères spéciaux à un pattern, de façon à enrichir ce qu'il représente. Par exemple, le pattern a* représente toutes les chaînes de caractères constituées d'un nombre quelconque de a (*y compris la chaîne vide*). Ajouter le caractère * à un pattern, signifie que ce pattern peut se répéter. Il est possible ensuite de définir et d'utiliser des classes de caractères. Une classe de caractères est définie par une chaîne de caractères écrite entre crochets, par exemple [abc] veut dire "un unique caractère qui peut être a, b ou c" [54].

Exemple 4.5 *En se basant sur l'exemple 4.1, on peut formuler les expressions régulières suivantes.*

— $Reg_1 = CV.CH.IP.SC.ED.$

— $Reg_2 = CV.CH.(IP)^*.SC.ED.$

— $Reg_3 = CV.CH.(IP)^*.SC.ED.C.MD^*.C.P$

— $Reg_4 = CV.CH.(IP)^*.SC.ED.C.MD^*.A$

Il est important de signaler que les traces d'exécution sont formalisées par des expressions régulières.

4.3.5 Notion de sous protocole

Souvent, une partie d'un protocole métier constitue une sous-procédure de gestion qui exprime un intérêt de gestion spécifique. En conséquence, l'interrogation des données d'exécution des instances du processus métier doit tenir compte de cet aspect. Cela est aussi utile dans le cas de changement d'une partie de la logique métier, l'intégration d'une fonctionnalité d'un partenaire ou encore lors de la substitution d'une règle de fonctionnement par une autre. Dans ce contexte, la notion de sous-protocole que nous introduisons, ci-après s'avère très importante.

Définition 4.6 ***Spécification d'un sous-protocole***
Soit $\mathcal{P} = (Q, q_0, \mathcal{F}, \mathcal{M}, \mathcal{R})$ un protocole de service, et soit $q \in Q$ un état de $\mathcal{P}$. Le sous-protocole $\mathcal{C}_{SP}(\mathcal{P}, q)$ de $\mathcal{P}$ est le protocole : $\mathcal{C}_{SP}(\mathcal{P}, q) = (Q'_s, q'_{0s}, \mathcal{F}'_S, \mathcal{M}'_S, \mathcal{R}'_S)$, obtenu à partir de $\mathcal{P}$ comme suit :

- *q est l'état initial de $\mathcal{C}_{SP}(\mathcal{P}, q)$,*
- *$Q'_s \subseteq Q$ contient l'ensemble des états de Q accessibles à partir de q en utilisant la relation $\mathcal{R}$ de $\mathcal{P}$,*
- *$\mathcal{F}'_s = Q'_s \bigcap \mathcal{F}$,*
- *$\mathcal{R}'_s = \{(q, q', m) \in \mathcal{R} \ \ c.à.d \ \{q, q'\} \subseteq Q'_s\}$,*
- *$\mathcal{M}'_s \subset \mathcal{M}$ est constitué des messages $\mathcal{M}$ qui apparaissent dans les transitions de $\mathcal{R}'_s$.*

Selon cette spécification, le sous protocole $C_{SP}(P, q)$ permet de capturer toutes les exécutions possibles de P à partir de l'état q. C'est à dire qu'il couvre tous les chemins d'exécutions futurs de P qui commence par l'état q et qui se terminent à l'un des états finaux de P.

Exemple 4.6 *En utilisant toujours le même exemple § 1, et à partir de cet exemple on fait l'extraction du sous protocole illustré ci-dessous. Ce sous protocole décrit une séquence d'activités incluses dans la logique métier du protocole principal.*

4.4 Modélisation de l'approche

D'une manière générale, les données d'une application informatique sont organisées de façon à pouvoir opérer des interrogations afin d'extraire certaines données d'intérêt ou pour faire des raisonnements adéquats sur ces données. Dans une base de données classique (*relationnelle*), les structures de tables, rubriques, lignes ... offrent bien cette possibilité avec le langage **SQL**. Souvent, l'utilisateur dispose d'un langage de commandes standard et normalisé qui lui permet d'interroger la base des données. Néanmoins, l'interrogation des bases de données n'est pas toujours évidente pour certaines données spécifiques, telles que celles des données d'exécution des processus métiers. En effet, il est fréquent que l'utilisateur désire obtenir des informations précises, complexes et par fois des données particulières, répondant à des besoins de gestion spécifiques.

Après avoir vu les concepts liés aux processus métiers, on passe à la modélisation de notre approche, une approche d'interrogation de données.

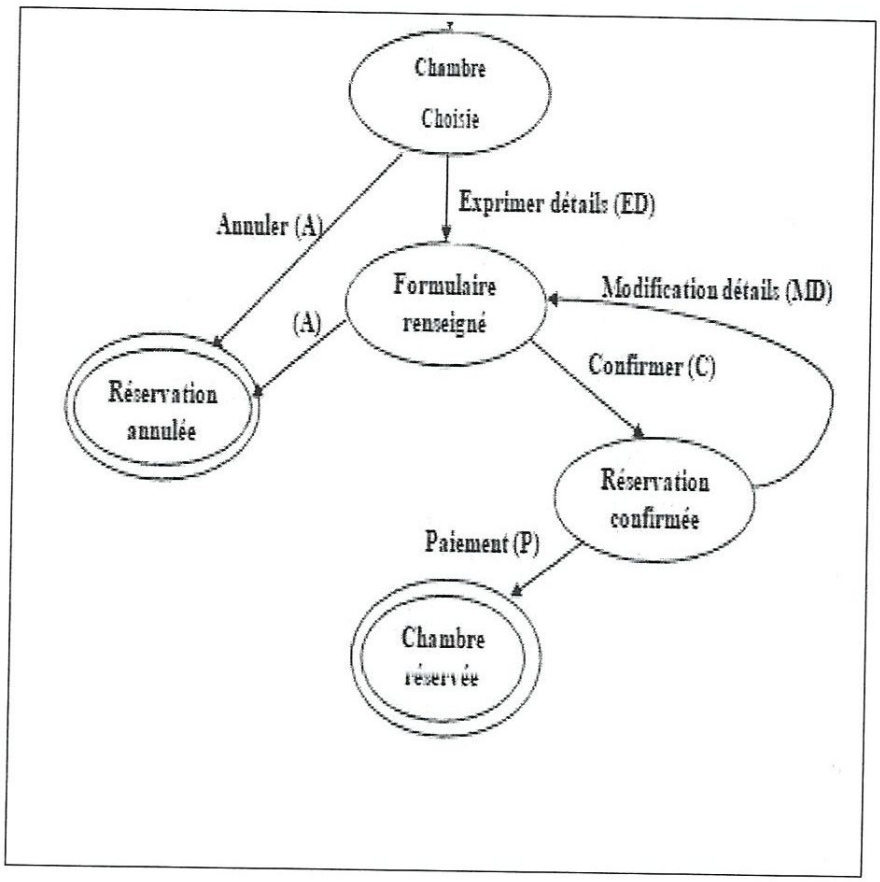

FIGURE 4.2 – Un exemple de sous protocole.

L'objectif est de répondre aux requêtes non conventionnelles des utilisateurs. C'est à dire des requêtes non prises en charge par les langages classiques (*SQL*, *XML*, *Cypher*,...etc.). A titre d'exemple, on peut se demander si on peut formuler des requêtes pour répondre aux préoccupations suivantes.

— **Requête 1 :** Donner la liste des instances, qui ont démarrées avant une telle date et elles sont en cours (leur état est active).

— **Requête 2 :** Quelles sont les traces d'exécution d'un **PM** qui contiennent des boucles sur états ?

Exemple 4.7 *Boucles sur états*

1. ab*c *: On à l'exécution de l'activité* a, *ensuite l'activité* b *ou plusieurs fois l'activité* b *et enfin l'activité* c.
2. a* *: On à l'exécution de l'activité* a *une ou plusieurs fois.*

— **Requête 3 :** Quelles sont les traces d'exécution d'un **PM** qui contiennent des itérations, c'est à dire des boucles sur des sous structures.

Exemple 4.8 *Boucles sur des sous-structures*

1. a(bcd)*e : *On à l'exécution de l'activité* a, *ensuite la séquence d'activités* bcd *une ou plusieurs fois et enfin l'activité* e.
2. (ab)*c : *On à l'exécution de la suite des activités* ab *ou plusieurs fois la suite d'activités* ab, *suivie de l'activité* c.

Pour pouvoir répondre à de telles soucis, nous aurons besoins de structures de données adéquates. Ces dernières peuvent êtres structurées (*tables relationnelles*), semi-structurées ou de type graphes. Une donnée semi-structurée ne doit se conformer à aucun schéma et pourtant il y a en son sein des structures (*souvent écrits avec des balises*) qu'il faut exploiter et qui sont utilisées par les langages de requêtes. Il est donc nécessaire de disposer de représentations (*de modèles ou schémas*) adaptés à ces données. Dans notre approche, nous allons se concentrer uniquement sur les deux derniers modèles de représentation.

1. Une représentation arborescente, qui convient particulièrement à la modélisation des données des processus métiers sous forme d'un document XML.
2. Une représentation par un graphe orienté étiqueté multi-racines. De manière générale ce modèle convient dès que les nœuds représentent des objets simples ou complexes, composant la donnée et les arcs représentent la composition de ces objets.

Dans ce qui suit, nous allons détaillé ces deux familles de représentations avec une spécification du modèle de données associé ainsi que les types de requêtes d'interrogation avec leur illustration par des exemples, et nous allons les procéder en deux étapes.

4.4.1 Fondement de l'interrogation des données semi-structurées

Les données structurées sont organisées selon une structure donnée, qui permet de les organiser et de les manipuler selon diverses combinaisons, afin de mieux exploiter les informations stockées. Par contre, les données semi-structurées sont des données qui n'ont pas été organisées en référentiel spécialisé, mais cela n'empêche pas d'avoir des méthodes, une solution convenable pour leur interrogation. Cela est dû au fait que les données semi-structurées ne conforment pas à un schéma fixe comme c'est le cas dans les bases de données relationnelles et les bases de données orientées objets. PAr conséquent, la représentation et l'interrogation des données semi-structurées sont plus difficile que dans les données structurées, à cause de la nature des données qui sont hétérogènes et à cause du schéma qui est absent, même si ce dernier est présent il ne constitue aucune contrainte sur la base.

4.4.2 Spécification du modèle de données XML pour les PM

La particularité d'un document XML est de contenir en même temps des données et des informations permettant d'identifier la structure et le sens de ces données.

Un document XML est composé de plusieurs parties :

— Entête de document précisant la version et l'encodage.

— Un arbre d'éléments basé sur un élément appelé racine.

— Des règles optionnelles permettant de vérifier si le document est valide.

<Transition Etat-Source="Données en cours" Etat-destination="Chambre choisie" > SC </Transition>

<Transition Etat-Source="Chambre choisie" Etat-destination="Formulaire renseigné" > ED </Transition>

<Transition Etat-Source="Formulaire renseigné" Etat-destination="Réservation confirmée" > C </Transition>

<Transition Etat-Source="Réservation confirmer" Etat-destination="Chambre réservée" > P </Transition>

<Transition Etat-Source="Formulaire renseigné" Etat-destination="Réservation confirmée" > MD </Transition >

<Transition Etat-Source="Formulaire renseigné" Etat-destination="Chambre choisie" > SC </Transition>

<Transition Etat-Source="Formulaire renseigné" Etat-destination="Données en cours" > IP </Transition >

<Transition Etat-Source="Données en cours" Etat-destination="Données en cours" > IP </Transition>

</Transitions>

</Automate>

4.4.3 Spécification des traces d'exécution en XML

1. **Attributs des traces :** Les traces sont caractérisées par plusieurs attributs, tels que :
   - Id trace : numéro attribué chronologiquement par le système.
   - Nom utilisateur : celui qui invoque le PM.
   - Temps de départ (time start) : le temps du lancement du PM.
   - Temps de terminaison (time end) : le temps d'arrêt de l'exécution de l'instance.
   - Séquence des activités exécutées.
   - L'etat de la trace (situation : i.e. ; en cours ou achevée).

   Dans nos travaux, on se limite aux propriétés précédentes considérées comme les plus importantes, bien qu'il existe d'autre propriétés, à savoir :
   - Les ressources manipulées.
   - Le temps de démarrage de chaque activité.
   - le coût de chaque activité.
   - Le coût global de la trace.
   - Le temps de terminaison de chaque activité.

   Échantillon de traces d'exécution (Jeu d'essai)
2. **Modèle XML de traces : Modèle de trace d'exécution en XML.** Comme les traces d'exécution sont perçues comme des fichiers au format XML, on donne ci-dessous la structure générique qui va contenir ces traces. *<trace>*

   <Id> id trace </Id>

   <nomcli> nom client </nomcli>

   <etat> etat instance </etat>

ID	User	Start	End	State	Séquence Activités
01	hamici	10 :00	13 :00	active	CV.CH.IP.IP.IP.SC.
02	client2	10 :00	13 :09	active	CV.CH.IP.IP.SC.
03	client3	10 :00	10 :50	terminer	CV.CH.IP.IP.IP.SC.ED.ED.A.
04	client4	14 :00	14 :19	active	CV.CH.IP.IP.IP.SC.ED.ED.IP.SC.ED.C.
05	client5	19 :00	20 :19	terminée	CV.CH.IP.IP.SC.ED.ED.SC.ED.C.P.
06	client6	19 :00	20 :19	terminée	CV.CH.IP.IP.SC.ED.ED.SC.ED.C.P.
07	client7	10 :00	13 :00	active	CV.CH.IP.IP.IP.SC.
08	client8	13 :00	13 :40	terminée	CV.CH.IP.SC.A.

TABLE 4.1 – Exemple d'ensembles de traces d'exécution.

<timestart> temps départ </timestart>
<timestemp> temps pris </timestemp>
<situation> situation d'instance </situation>
<sequenceactivite>
<activite> activité 1 </activite>
<activite> activité 2 </activite>
<activite> activité 3 </activite>
</sequenceactivite>
</trace>

3. **Exemple d'ensemble de trace en** XML : En tenant compte de la spécification XML précédente, les quatre traces contenues dans la **table 4.1** précédente des traces seront exprimées au format XML suivant.

*// **INSTANCE 1** //*
<latrace>
<id> 01 </id>
<nomcli> hamici </nomcli>
<etat> chambre choisie </etat>
<timestart> 10 :00 </timestart>
<timestemp> 03 :00 </timestemp>
<situation> active </situation>
<etat1>début</etat1>
<etat2>ville selectionnée</etat2>
<etat3>hôtel selectionné</etat3>
<etat4>données en cours</etat4>
<etat5>données en cours</etat5>
<etat6>données en cours</etat6>
<etat7>chambre choisie</etat7>
<sequenceactivite>
<activite>CV.</activite>
<activite>CH.</activite>

<activite>IP.</activite>
<activite>IP.</activite>
<activite>IP.</activite>
<activite>SC.</activite>
</sequenceactivite>
</latrace>
// **INSTANCE 2** //
<latrace>
<id> 02 </id>
<nomcli> client2 </nomcli>
<etat> chambre choisie </etat>
<timestart> 10 :00 </timestart>
<timestemp> 03 :19 </timestemp>
<situation> active </situation>
<etat1>début</etat1>
<etat2>ville selectionnée</etat2>
<etat3>hôtel selectionné</etat3>
<etat4>données en cours</etat4>
<etat5>données en cours</etat5>
<etat6>chambre choisie</ctat6>
<sequenceactivite>
<activite> CV.</activite>
<activite> CV.</activite>
<activite> IP.</activite>
<activite> IP.</activite>
<activite> SC.</activite>
</sequenceactivite>
</latrace>
// **INSTANCE 3** //
<latrace>
<id> 03 </id>
<nomcli> client3 </nomcli>
<etat> reservation annulée</etat>
<timestart> 10 :00 </timestart>
<timestemp> 00 :50 </timestemp>
<situation> terminée </situation>
<etat1>début</etat1>
<etat2>ville selectionnée</etat2>
<etat3>hôtel selectionné</etat3>
<etat4>données en cours</etat4>
<etat5>données en cours</etat5>

<etat6>données en cours</etat6>
<etat7>chambre choisie</etat7>
<etat8>formulaire renseigné</etat8>
<etat9>formulaire renseigné</etat9>
<etat10>réservation annulée</etat10>
<sequenceactivite>
<activite>CV.</activite>
<activite>CH.</activite>
<activite> IP.</activite>
<activite> IP.</activite>
<activite> IP.</activite>
<activite>SC.</activite>
<activite>ED.</activite>
<activite> ED.</activite>
<activite> A.</activite>
</sequenceactivite>
</latrace>

// **INSTANCE 4** //

<latrace>
<id> 04 </id>
<nomcli> client4 </nomcli>
<etat> reservation confirmée </etat>
<timestart> 14 :00 </timestart>
<timestemp> 00 :19 </timestemp>
<situation> active </situation>
<etat1>début</etat1>
<etat2>ville selectionnée</etat2>
<etat3>hôtel selectionné</etat3>
<etat4>données en cours</etat4>
<etat5>données en cours</etat5>
<etat6>données en cours</etat6>
<etat7>chambre choisie</etat7>
<etat8>formulaire renseigné</etat8>
<etat9>formulaire renseigné</etat9>
<etat10>données en cours</etat10>
<etat11>chambre choisie</etat11>
<etat12>formulaire renseigné</etat12>
<etat13>réservation confirmée</etat13>
<sequenceactivite>
<activite> CV.</activite>
<activite> CH.</activite>

<activite> IP.</activite>
<activite> IP.</activite>
<activite> IP.</activite>
<activite> SC.</activite>
<activite> ED.</activite>
<activite> ED.</activite>
<activite> IP.</activite>
<activite> SC.</activite>
<avtivite> ED.</avtivite>
<activite> C.</activite>
</sequenceactivite>
</latrace>
*// **INSTANCE 5** .....//*
*// **INSTANCE 6** .....//*
*// **INSTANCE 7** .....//*
*// **INSTANCE 8** .....//*

4.4.4 Formulation des requêtes d'interrogation avec XQuery

XQuery est un langage non XML, mais un langage de requêtes pour XML, qui peut être vu comme une extension de **XPath**. Il permet de traiter des ressources XML [55].

XQuery est un langage fonctionnel manipulant des séquences d'items. Une requête **XQuery** est donc une expression portant sur des items, chaque expression retourne une valeur et n'a pas d'effet de bord. Une requête **XQuery** est composée de trois parties :

1. Une ligne d'entête commencée par xquery et contenant la version et, éventuellement l'encodage du document.
2. Un ensemble de déclarations :
   - déclarations de variables ou constantes,
   - déclarations de fonctions utilisateur locales,
   - importation de modules (bibliothèques XQuery),
   - détermination du schéma du résultat,
   - détermination des espaces de nom et de leur utilisation,
   - détermination du format du résultat et autres paramètres
3. L'expression **XQuery** à exécuter.

Avant de passer à la formulation des requêtes, il faut tout d'abord créer une base de données en XML. C'est ce qu'on appel une **base-X**.

Ci-dessous, nous présentons la commande de création de la base et le résultat obtenu.

Commande 1 : Création de la base AFD

CREATE DB monAFD C :/Program Files(x86)baseX/etc/monAFD.xml
Result :

Database 'monAFD' created in 387.85ms
Ensuite, on passe à la création de la base des traces qu'on interrogera par la suite.
Commande 2 :Création de la base des traces

CREATE DB monAFD C :/Program Files(x86)baseX/etc/basetrace.xml
Result :

Database 'basetrace' created in 387.85ms

4.4.5 Exemples de requêtes

Après avoir créer la base des traces, on peut passer maintenant à l'interrogation de cette base. Pour cela des requêtes spécialisées sont nécessaires.

L'objectif est de pouvoir extraire des **patterns** ou motifs récurrents qui représentent un intérêt pour le gestionnaire des $\mathcal{PM}$. Chaque pattern est exprimé par une expression régulière impliquant des attributs relatifs aux traces ou aux activités. A titre d'illustration on peut se demander *quelles sont les instances du processus qui ont exécuté la séquences d'activités a.b.c et qui sont encore à l'état actif ?*

On classe les requêtes d'interrogation en deux catégories : **simple** et **complexes**.

1. Requêtes simples : Elle agissent sur les données d'exécution par simple filtrage sur des valeurs d'attributs de la trace.
   - **Requête 1** : Elle nous permet d'avoir la liste des instances dont l'état est **actif**.

     for $x in doc("basetrace.xml")/basedestrace/latrace
     return if($x/situation="active")
     then
     ** data($x/nomcli) **

   - Résultat Requête 1 :
     **
hamici
**
     **
client2
**
     **
client4
**
     **
client7
**

   - **Requête 2** : Elle nous permet d'extraire les instances qui ont atteint un certain état spécifié comme paramètre de la requête.

     for $x in doc("basetrace.xml")/basedestrace/latrace
     where ($x/etat="chambre choisie")
     return
     ** data($x/nomcli) **

— Résultat Requête 2 :
**
hamici
**
**
client2
**
**
client7
**

— **Requête 3 :** Elle nous donne les activités exécutées par une instance caractérisée par son utilisateur.
for $x in doc("basetrace.xml")/basedestrace/latrace
return if($x/nomcli="hamici")
then
** le nom du client :data($x/nomcli)**
**ses activites : data($x/sequenceactivite) **

— Résultat Requête 3 :
** le nom du client :hamici**
**ses activités : CV.CH.IP.IP.IP.SC.
**

— **Requête 4 :** Elle nous retourne la liste des instance qui sont à l'état actif et qui ont débuté à un temps donné.
for $x in doc("basetrace.xml")/basedestrace/latrace
return if(($x/situation="active") and ($x)/timestart= "10 :00")
then
** data($x/nomcli) **

— Résultat Requête 4 :
**
hamici
**
**
client2
**
**
client7
**

— **Requête 5** : Elle permet de donner la liste des instance ayant démarrer après un temps donné en entrée et que sont à l'état actif.
for $x in doc("basetrace.xml")/basedestrace/latrace
return if(($x/situation="active") and ($x)/timestart > "10 :00")
then
** data($x/nomcli) **

— Résultat Requête 5 :
**
client1
**

— **Requête 6** : Elle permet de filtrer toutes les instances qui ont terminé leur exécution (*ayant atteint un état final*).
for $x in doc("basetrace.xml")/basedestrace/latrace
where ($x/situation = "terminée")
return

** data($x/nomcli) **

— Résultat Requête 6 :
**
client3
**
**
client5
**
**
client6
**
**
client8
**

2. Requêtes complexes : Ce type de requête permet de pallier aux insuffisances des langages d'interrogation relationnel, tel que **SQL**. En effet, certains préoccupations des gestionnaires de PM ne peuvent être exprimées en **SQL**, mais cela est possible avec les langages d'interrogation des données semi-structurées, tel que **XQuery**.

— **Requête 7** : Cette requête donne les instance qui ont refait une activité données. Autrement dit, les instance qui ont opéré une boucle sur état particulier contenu dans la spécification de l'automate.
for $x in doc("basetrace.xml")/basedestrace/latrace
return
if(xs :string($x/etat7) eq xs :string($x/etat8))
then data($x/nomcli) : a refait cette étape :
**data($x/etat7) **
else
**
data($x/nomcli) : n'a pas refait cette étape **

— Résultat requête 7 :
**
hamici : n'a pas refait cette étape **
**
client2 : n'a pas refait cette étape **
**
client3 : n'a pas refait cette étape **
**
client4 : n'a pas refait cette étape **
**
client5 : a refait cette étape : formulaire renseigné
**
**
client6 : a refait cette étape : formulaire renseigné
**
**
client7 : n'a pas refait cette étape **

— **Requête 8** : Au lieu de s'intéresser à des itérations sur un simple état, comme dans la requête précédente, on peut généraliser la requête pour extraire toutes les instances qui ont bouclé sur une sous structure de l'automate (*un sous-protocole, Voir section 4.3.5*). Ainsi, l'état initial du sous-protocole sera l'état ou commence la boucle en question.
for $x in doc("basetrace.xml")/basedestrace/latrace
where (xs :string($x/etat6) eq xs :string($x/etat10))

return

** Le client : data($x/nomcli) :a une boucle sur sous structure**
,

commence par data($x/etat6) ;
et voilà ses traces :data($x/sequenceactivite)

— Résultat requête 8 :
 Le client : client4 : a exécuté une boucle sur sous structure
,
commence par données en cours ; et voilà sa trace :
CV.CH.IP.IP.IP.SC.ED.ED.IP.SC.ED.C.

 Le client : client5 : a exécuté une boucle sur sous structure
,
commence par chambre choisie ;
et voilà sa traces :CV.CH.IP.IP.SC.ED.ED.SC.ED.C.P.

 Le client : client6 : a exécuté une boucle sur sous structure
,
commence par chambre choisie ;
et voilà sa traces :CV.CH.IP.IP.SC.ED.ED.SC.ED.C.P.

Pour montrer la consistance de notre approche d'interrogation, nous formulons ci-après quelques requêtes d'agrégation qui fournissent des données statistiques sur les traces d'exécution.

— **Requête 9** : Cette requête complexe permet de calculer le nombre total d'instances existantes dans notre base.
<library count="count(doc("basetrace.xml")/basedestrace/latrace)">
for $x in doc("basetrace.xml")
return <nombreclients>$x/text()</nombreclients> </library>

— Résultat requête 9 :
<library count="8">
<nombreclients/>
</library>

— **Requête 10** : Cette requête nous permet de connaître le nombre d'instances qui ont terminées leur exécutions.
<LesSituations
nombre=" count(doc('basetrace.xml')//situation[text() = 'terminée']) ">
doc("basetrace.xml")//situation[text() = 'terminée']
</LesSituations>

— Résultat requête 10 :
<LesSituations nombre="4">
<situation>terminée</situation>
<situation>terminée</situation>

<situation>terminée</situation>
<situation>terminée</situation>
</LesSituations>

Il est important de signaler que dans toutes les requêtes formulées précédemment, nous avons spécifié uniquement un seul attribut qui est le nom du client, comme sortie de la requête. Néanmoins, on peut étendre la liste des attributs de sortie à d'autres champs en modifiant simplement la balise *
... *.

4.5 Conclusion

Ce chapitre constitue notre contribution par la proposition d'une approche d'interrogation des données d'exécution des **PM**.

Dans un premier temps, nous avons commencé par la présentation du modèle formel supportant le choix de représentation des **PM**, puis nous avons exposé les notions et concepts utilisés dans le cadre de notre approche. Par la suite, une modélisation des **PM** sous forme de fichier XML a été proposée et les traces d'exécution ont été conçues dans le même modèle de représentation. Une fois ces éléments bien cernés, on a exprimé différents types de requêtes d'interrogation en utilisant le langage *XQuery*. Les requêtes formulées ont été illustrées par des exemples inspirés d'un cas réel de **PM** relatif au domaine de la gestion hôtelière (*réservation d'une chambre*).

Le prochain chapitre sera dédié à l'implémentation de notre approche pour la réalisation d'un prototype logiciel.

CHAPITRE 5

Implémentation et expérimentation

5.1 Introduction

Ce chapitre est consacré à l'implémentation et à l'expérimentation de notre approche d'interrogation des données des processus métiers.

On commence tout d'abord par la présentation des outils logiciels utilisés pour la réalisation du prototype. Ensuite, on passe à l'exposé de l'architecture et les fonctionnalités du système réalisé. Enfin, quelques résultats expérimentaux sont présentés et discutés.

5.2 Présentation des outils utilisés

Pour l'implémentation de notre approche, on a fait usage de plusieurs outils logiciels et environnements qui rendent différents services.

Dans un premier temps, on a exploré l'environnement BPM **Bonitafost** pour la gestion du cycle de vie des processus métiers ; de la création jusqu'à la maintenance et la supervision. Son usage fût matérialisé par la modélisation du processus de réservation d'une chambre d'hôtel. Étant donné que notre approche est destinée à interroger des données semi structurées relatives à l'exécution des processus métiers, nous avons aussi fait appel à un gestionnaire de données **NoSQL** et on a utilisé **XQuery** pour l'extraction des données à partir de fichiers XML à l'aide de requêtes d'interrogation. En dernier lieu, on a utilisé l'incontournable environnement Eclipse-Java pour la réalisation des interfaces graphiques, pour la connexion des différents modules ainsi que pour la génération des données de test.

Dans ce qui suit, on donne une brève présentation de ces outils. Pour le lecteur qui veut aborder les aspects techniques approfondis de ces outils, il doit consulter la documentation technique de chacun d'eux.

5.2.1 Bonitasoft

Bonita est une plate-forme applicative open source de gestion de processus métier. Elle nous aide à modéliser nos processus d'entreprise sous forme de diagramme d'activités (*états-transition*). Après la création du schéma du processus, l'utilisateur peut modifier le modèle et générer des instances avec des niveaux d'exécution variés. Ainsi, avec le temps, une base de données des processus est gérée par BonitaSoft. De même les données d'exécution qui lui sont associées sont stockée au niveau du système.

La figure **5.1** montre l'interface principale de Bonitasoft.

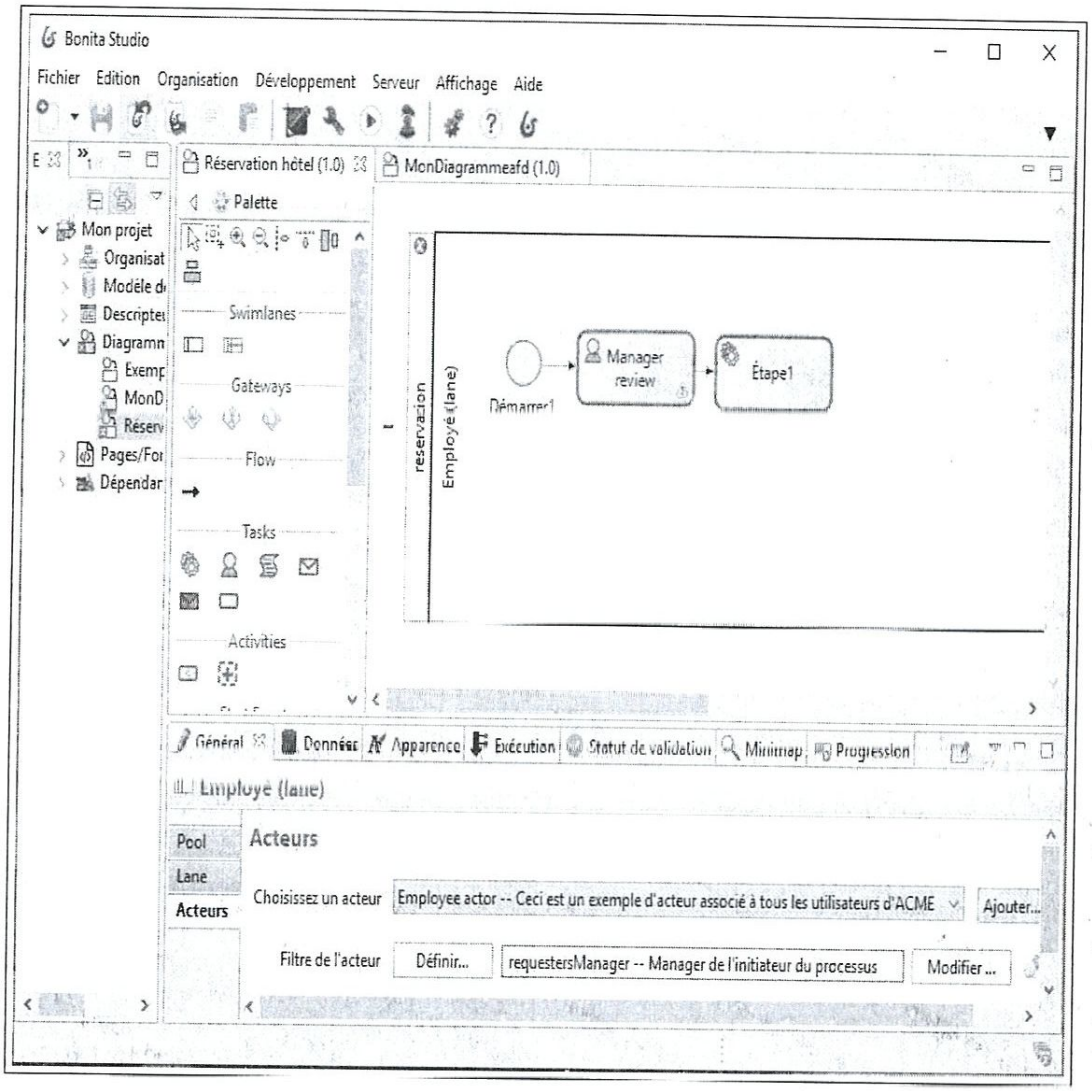

FIGURE 5.1 – Interface de BonitaSoft.

5.2.2 XQuery

XQuery est un langage de requêtes d'interrogation des données semi-structurées. Il nous permet d'extraire des informations d'un document XML avec des requêtes. Ces requêtes ont une syntaxe précise, dite aussi **FLWOR** (*prononcer flower*), dont le nom vient des cinq clauses principales qui la composent (*for, let, where, order by et return*).
La figure **5.2** illustre l'interface de **XQuery**. On remarque que l'interface du **XQuery**

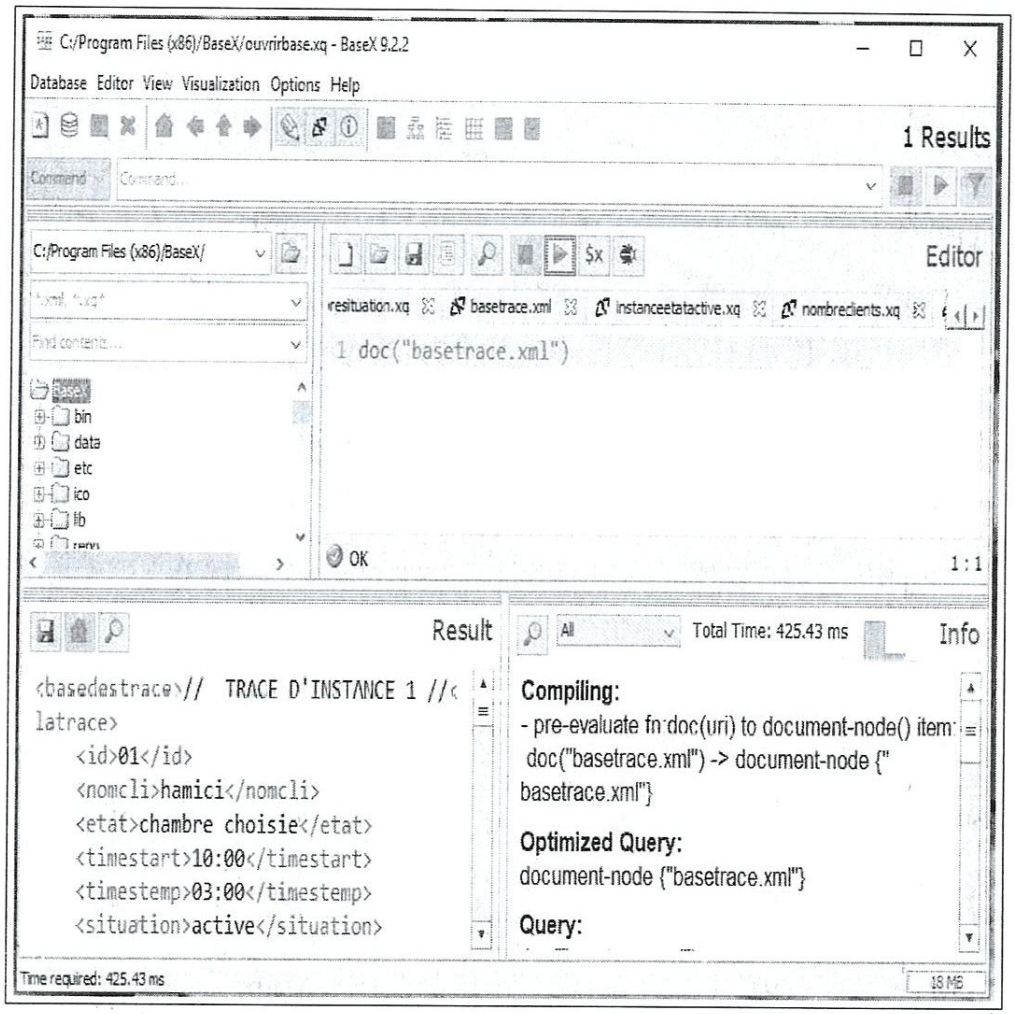

FIGURE 5.2 Interface du XQuery.

se subdivise en quatre parties,

1. Partie 1 : Pour les répertoires.
2. Partie 2 : En dessous la partie 1, pour l'affichage des résultats.
3. Partie 3 : Au coté droit de la partie 2, pour l'affichage des erreurs ainsi que le Query plan.
4. Partie 4 : La partie principale, la partie spécifie pour l'éditeur.

Il existe d'autres parties optionnelles qui sont accessibles à partir de la barre d'outils, ensuite le bouton **visualisation**.

5.2.3 JRE et Eclipse

Eclipse est un environnement de développement intégré (IDE) dont l'objectif est de fournir une plate-forme modulaire permettant de réaliser des développements d'applications informatiques. I.B.M. est à l'origine du développement d'Eclipse qui est, d'ailleurs, toujours le cœur de son outil Websphere Studio Workbench (WSW), lui même à la base de la famille des derniers outils de développement en Java d'I.B.M. Tout le code d'Eclipse a été donné à la communauté par I.B.M afin de poursuivre son développement. Eclipse utilise énormément le concept de modules nommés "plug-ins" dans son architecture. D'ailleurs, hormis le noyau de la plate-forme nommé "Runtime", tout le reste de la plate-forme est développé sous la forme de plug-ins. Ce concept permet de fournir un mécanisme pour l'extension de la plate-forme et ainsi fournir la possiblité à des tiers de développer des fonctionnalités qui ne sont pas fournies en standard par Eclipse. Les principaux modules fournis en standard avec Eclipse concernent Java [56].

Les vues représentent l'élément de base permettant de naviguer dans une hiérarchie d'informations. Elles sont typiquement utilisées pour ouvrir un éditeur de texte, afficher des propriétés sur la fenêtre active ou récupérer des informations spécifiques sur l'application. Par défaut, l'IDE propose un certain nombre de vues, telles que l'explorateur de package, la fenêtre des propriétés, l'explorateur de classes, etc ... [57].

5.3 Architecture de système

Dans ce qui suit, nous décrivons l'architecture du système réalisé et nous expliquons l'interaction entre les différents modules.

Comme il est observé dans la figure 5.3, le système réalisé s'articule autour de trois base de données et cinq modules qui communiquent ensemble afin de répondre aux requêtes d'interrogation des données des processus métiers.

Dans ce qui suit, les différents composants sont expliqués en détails.

1. **Module 1 :Analyseur de requêtes :** ce module permet d'analyser la syntaxe de la requête en se basant sur les mots clés du langage. Donc, en commence par tester le premier mot de la requête qui va donner la nature de la requête introduite par l'utilisateur (Select pour SQL, for pour Xquery, MATCH pour Cypher). Une fois identifiée, la requête sera transmise vers le deuxième module
2. **Module 2 :Module d'orientation** : A ce niveau, et après décision sur le type de requêtes, la base de données associée à la requête est chargée (ouverte). Ainsi, suivant le type de requêtes, la base de données adéquate est prise en charge.
3. **Module 3 :***SQL*. Pour les requêtes permettant l'interrogation des bases des données relationnelles.
4. **Module 4 :***XQuery*. Pour les requêtes permettant l'interrogation des bases des données semi-structurées.
5. **Module 5 :***Chypher*. Pour l'interrogation des bases des données graphes.

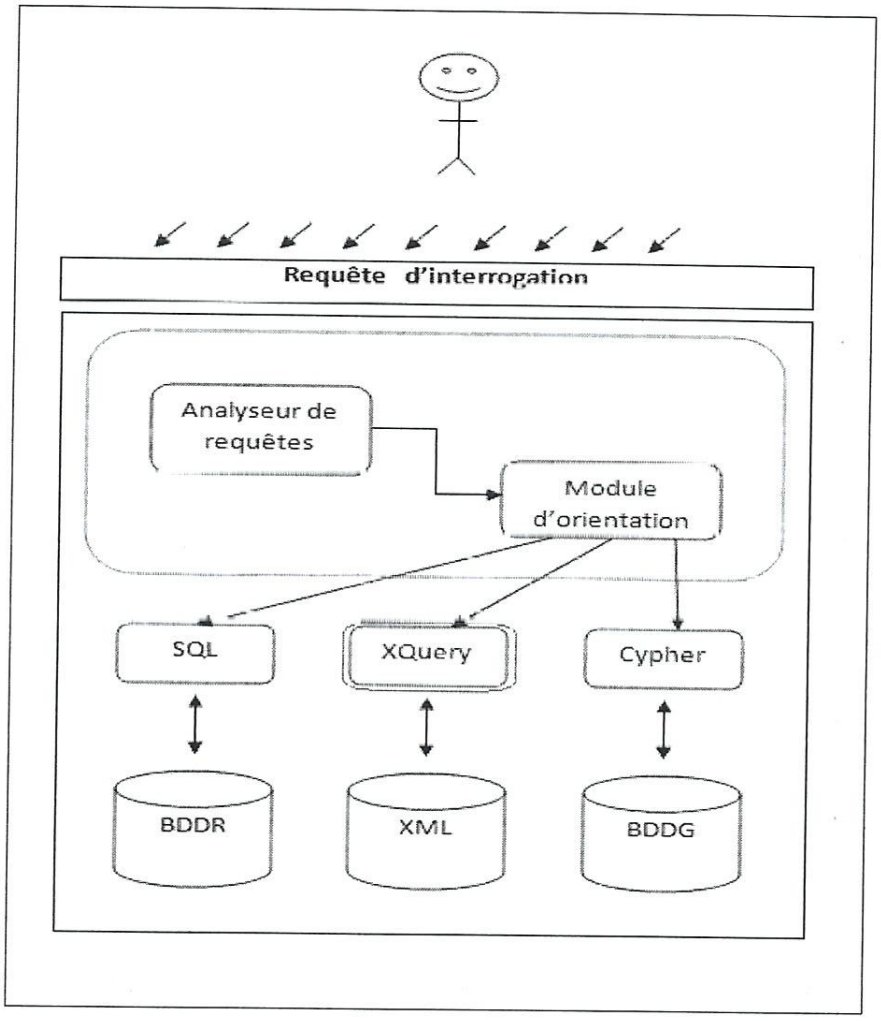

FIGURE 5.3 – Architecture du système.

6. **BDD 1 :** *BDDR.* Dans le cas ou la base des données est relationnelle (*les données sont stockées dans des tables*).
7. **BDD 2 :** *XML.* Dans le cas ou les données sont semi-structurées (*les données sont stockées dans des fichier d'extension XML*).
8. **BDD 3 :** *BDDG.* Si les données sont présentées par des graphes et stockées dans des base de données graphes (*les données sont stockées sous forme de graphe*).

Nous rappelons que la manipulation des bases de données relationnelles et un exercice facile et leur popularité est très répandue, d'autre part la question de l'interrogation des données sous forme de graphes (*BDDG*) a été largement abordée dans le PFE de l'année passée [58]. C'est pourquoi, dans notre implémentation on se focalise uniquement sur l'interrogation des données semi-structurées via **Xquery**.

La figure **5.4** illustre l'interface générale de notre application.

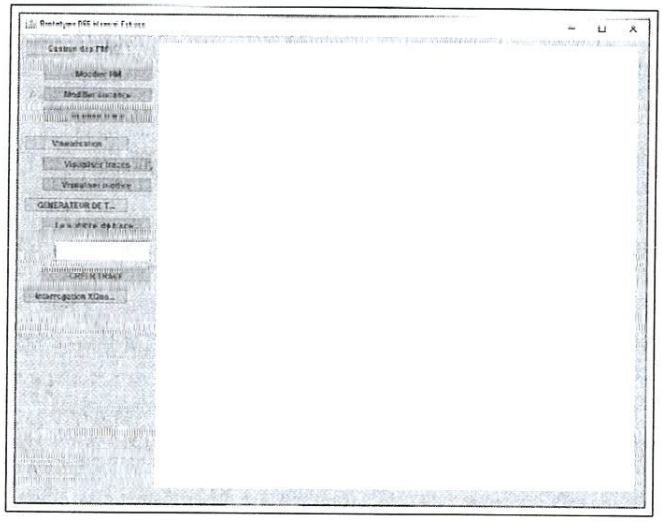

FIGURE 5.4 – L'interface principale de notre prototype.

Quatre modules sont greffés dans le menu général et permettent de réaliser :

— la gestion des processus métiers ;

— la visualisation des traces et des modèles ;

— la génération des traces d'exécution ;

— l'interrogation des données.

5.4 Fonctionnalités du système et scénario d'utilisation

Pour la réalisation de notre application, les trois outils précédents (BonitaSoft, Eclipse et Xquery) ont été utilisés conjointement de façon à répondre aux objectifs attendus. Chaque module répond à une préoccupation particulière.

BonitaSoft a été utilisé pour la création et la maintenance des processus métiers, ainsi que pour la génération des instances et des traces. Pour Eclipse, son rôle été d'assurer la connexion aux différentes BDD, l'implémentation des interfaces, la génération des données de test et l'estimation des temps de réponses aux requêtes. Enfin, l'outil Xquery constitue uniquement le langage d'interrogation des données semi-structurées.

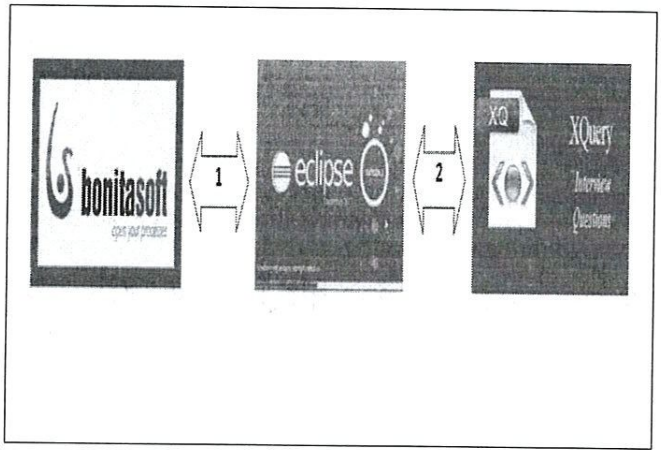

FIGURE 5.5 – Interconnexion des outils utilisés.

La figure **5.5**, ci-dessus, illustre l'interconnexion des trois outils utilisés dans le cadre de notre application.

Nous présentons, ci-après un enchainement des différentes phases illustrant la mise en œuvre de notre approche. Le scénario d'expérimentation est basé sur le processus métier "*Réservation d'une chambre d'hôtel*"

Les captures d'écran suivantes montre la manipulation de l'outil BonitaSoft pour l'accès au système, pour la création du processus métier, la génération des instances et la gestion des différentes tâches du processus.

On commence par les captures d'écran relatives à l'accès et à la connexion au système BonitaSoft. Pour exploiter le système, il faut tout d'abord opéré un log-in pour lancer le processus (*Figure* ***5.6***)

FIGURE 5.6 – Bonita log-in.

Le lancement d'un processus métier se fait simplement par la sélection du nom de processus en question, puis par un clic sur l'icône **Start** (*Figure* ***5.7***)

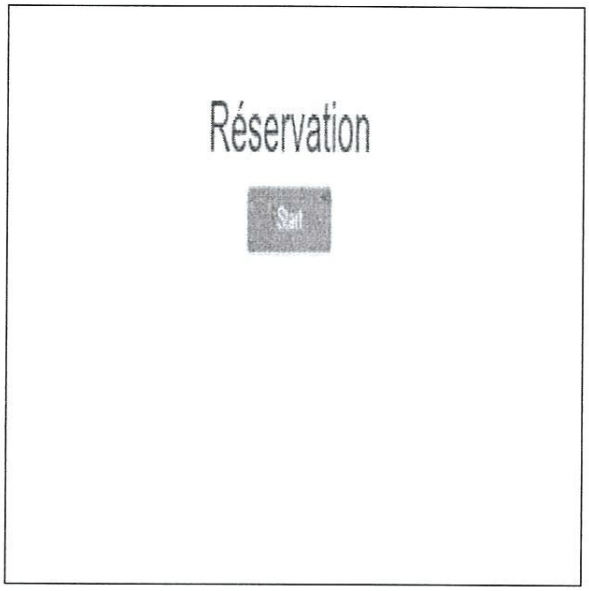

FIGURE 5.7 – Lancement de processus Réservation d'une chambre.

Une fois le processus en cours est actif, on passe maintenant à la création de son schéma (*modèle*), en spécifiant l'enchainement des différentes activités et les conditions de leur ordonnancement. La première étape est le début du processus, puis on crée les différentes activités, les branchements (*boucles, choix, ...*) et on termine par les états finaux du processus. De cette façon différents chemins sont décrits dans la spécification du processus métier (*Figure* ***5.8***)

Après la création du schéma du processus de réservation d'une chambre, on peut à présent passer à la phase de création des instances, en choisissant les tâches réalisées. De cette façon, chaque instance sera dotée de ses données (*ID, Nom utilisateur, temps de démarrage...*) et en même temps les activités réalisées jusqu'à présent.

Les captures d'écran **5.9**, **5.10** et **5.11** montrent la création des instances de processus, la création des tâches associées à des instances ainsi que les tables visualisant les tâches des instances de processus.

A signaler qu'après la génération d'un processus métier quelconque avec ses données d'exécution (*les traces réelles*), l'ensemble des données sont stockées par BonitaSoft dans une base de données relationnelle de type H2. C'est la raison pour laquelle, nous allons développé notre propre module de génération des instances et des traces en Java et par la suite on stockera, plutôt, les données d'exécution dans des bases de données semi-structurées (fichier XML).

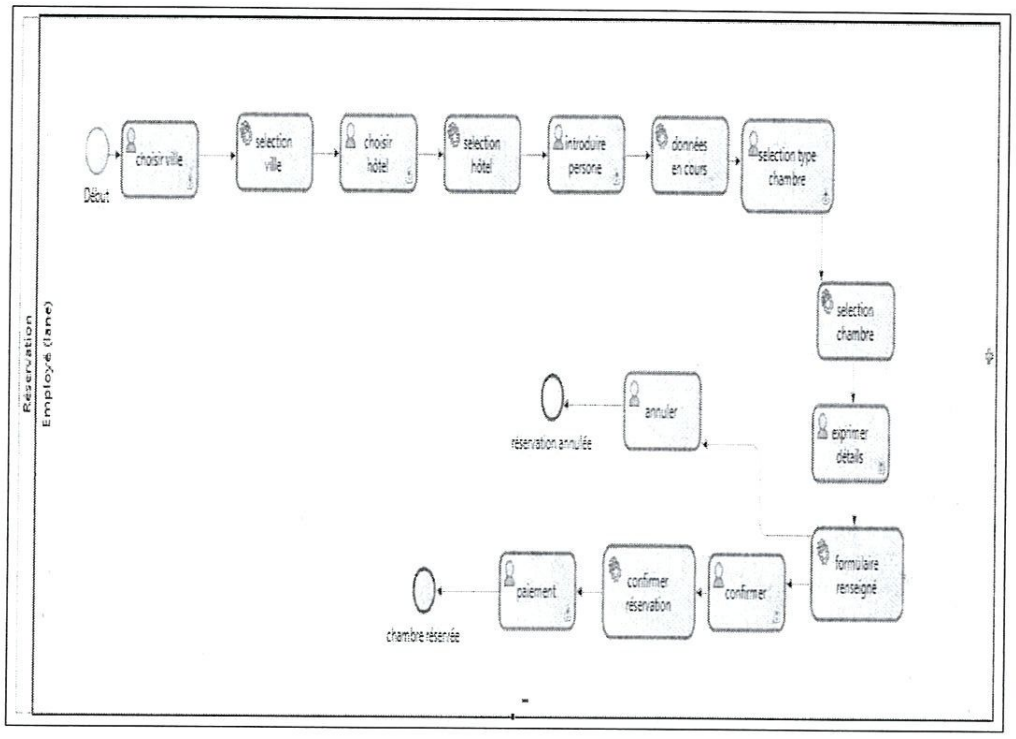

FIGURE 5.8 – Schéma du processus métier Réservation d'une chambre.

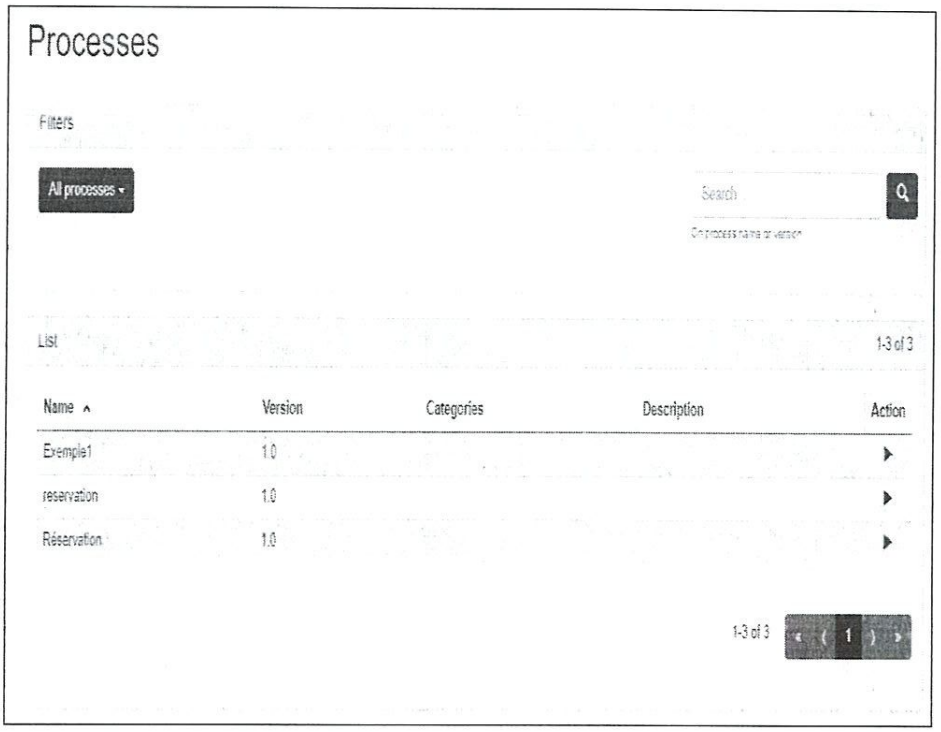

FIGURE 5.9 – Création des instances de processus.

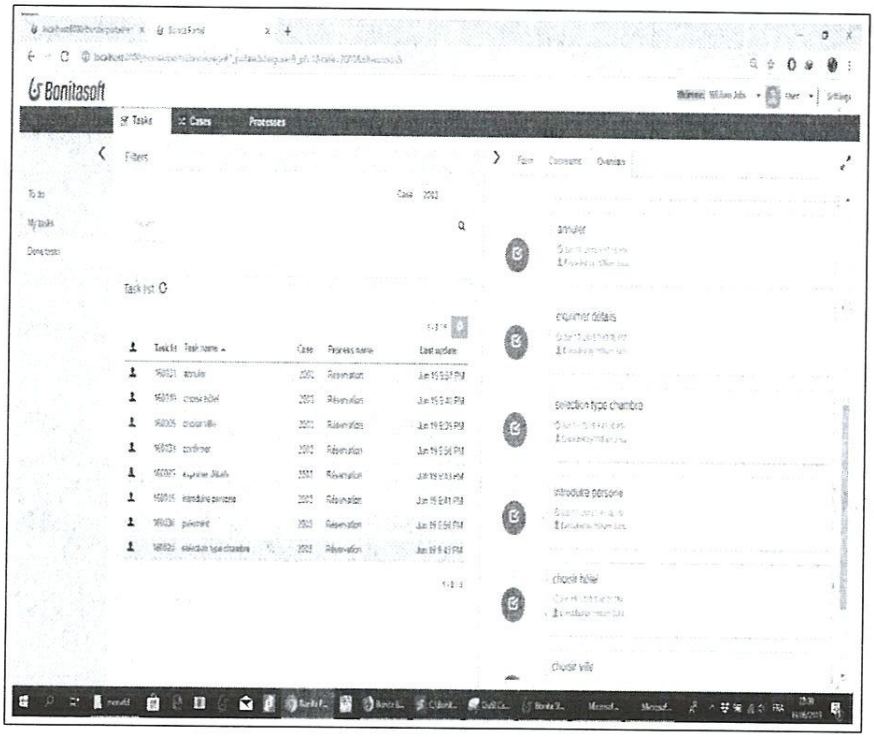

FIGURE 5.10 – Création des tâches associées à des instances de processus.

Task list

1 - 8 / 8

Task Id	Task name	Process name	Last update	Assigned on
160031	annuler	Réservation	Jun 19 9:57 PM	Jun 19 9:43 PM
160010	choisir hôtel	Réservation	Jun 19 9:40 PM	Jun 19 9:39 PM
160005	choisir ville	Réservation	Jun 19 9:39 PM	Jun 19 9:39 PM
160033	confirmer	Réservation	Jun 19 9:58 PM	Jun 19 9:57 PM
160025	exprimer détails	Réservation	Jun 19 9:43 PM	Jun 19 9:43 PM
160015	introduire persone	Réservation	Jun 19 9:41 PM	Jun 19 9:40 PM
160038	paiement	Réservation	Jun 19 9:59 PM	Jun 19 9:58 PM
160020	selection type chambre	Réservation	Jun 19 9:43 PM	Jun 19 9:41 PM

1 - 8 / 8

FIGURE 5.11 – Visualisation des tâches des instances de processus.

On passe maintenant à l'interrogation des données d'exécutions des instances, mais tout d'abord il faut faire la lecture des fichiers XML qui stockent les données d'exécution. Pour cela on a implémenté une fonction en java qui permet le lecture du modèle de processus métier en XML.
La figure **5.12** ci-après montre le fichier XML du processus métier Réservation d'une chambre au format XML. En plus, la figure **5.13** affiche le code Java pour la lecture du fichier XML.

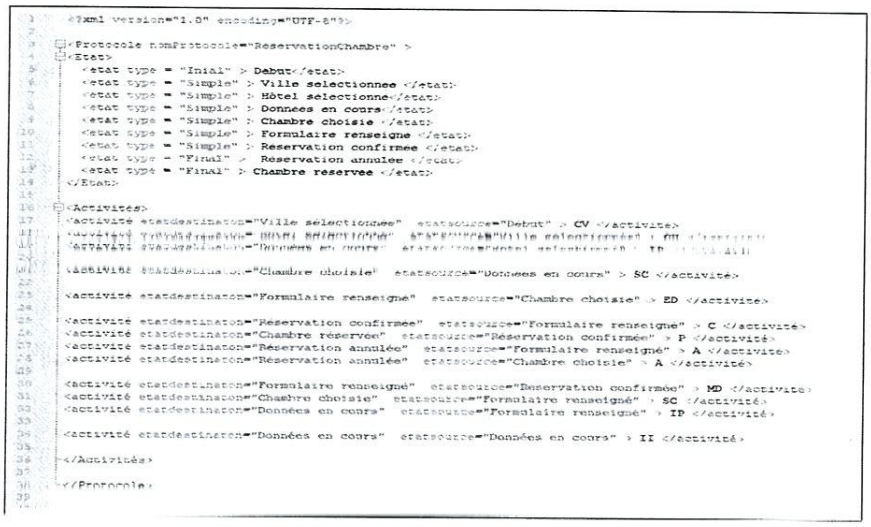

FIGURE 5.12 – Schéma du modèle utilisé en XML.

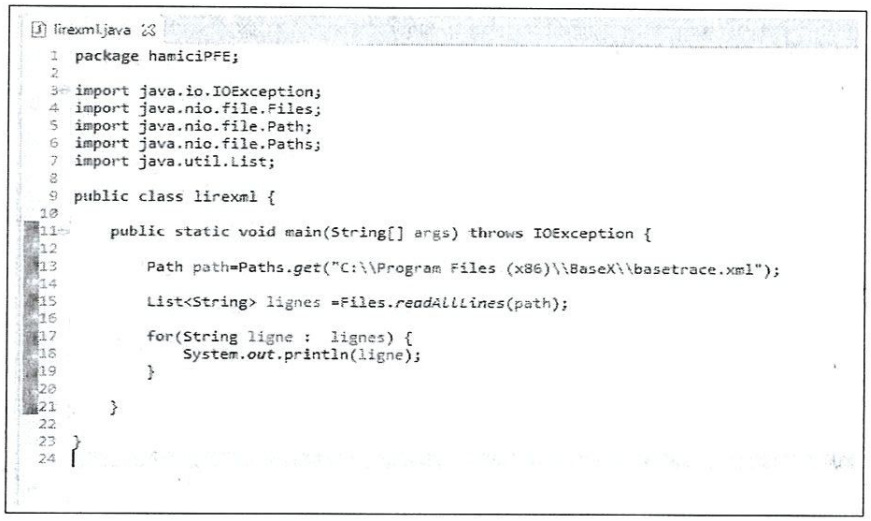

FIGURE 5.13 – Code java de la fonction de lecture du fichier XML.

5.5 Quelques résultats d'expérimentation

Pour l'expérimentation de notre approche, on a évalué le temps de réponse des différentes requêtes XQuery sur les différentes données de test. Pour cela, on a choisi trois ensemble de données d'expérimentation (*Data-sets*) ayant les tailles respectives suivantes : DS1=100 instances, DS2=1000 instances et DS3=10.000 instances. Les mêmes requêtes exprimées en Xquery ont été lancées sur les trois données d'expérimentation et le temps nécessaires à la fourniture des réponses correspondantes a été mesuré.

Le tableau **5.1** ci-dessous affiche les temps de réponse calculés pour chaque requête et chaque ensemble de données.

Requête d'interrogation	Temps de réponse
R1 : Ouverture de la base de 100 traces de 1000 traces de 10000 traces	Le temps pour chaque base Total Time : 232.37 ms Total Time : 606.66 ms Total Time : 4266.1 ms
R2 : Les instances avec boucle sur état Dans la base de 100 traces Dans la base de 1000 traces Dans la base de 10000 traces	Le temps pour chaque base Total Time : 59.31 ms Total Time : 638.47 ms Total Time : 3146.89 ms
R3 : Le nom des clients de la base Dans la base de 100 traces Dans la base de 1000 traces Dans la base de 10000 traces	Le temps pour chaque base Total Time : 42.36 ms Total Time : 487.45 ms Total Time : 2968.82 ms
R4 : Les noms des instance ayant atteint un état particulier Dans la base de 100 traces Dans la base de 1000 traces Dans la base de 10000 traces	Le temps pour chaque base Total Time : 126.14 ms Total Time : 560.99 ms Total Time : 2887.6 ms
R5 : Le nombre des instance qui ont terminées leur exécution Dans la base de 100 traces Dans la base de 1000 traces Dans la base de 10000 traces	Le temps pour chaque base Total Time : 60.44 ms Total Time : 441.16 ms Total Time : 2714.64 ms

TABLE 5.1 – Variation du temps de réponse pour quelques requêtes d'interrogation.

Il est observé que le temps de réponse à chaque type de requête croit linéairement avec le nombre de traces des instances en entrée.

Cependant, il serait très intéressant de comparer ce temps avec le temps de réponse pour les mêmes requêtes, mais dans des bases relationnelles ou des BDD graphes. Cette comparaison n'a pas été réalisée dans le cadre de notre projet de fin d'études faute de temps. En effet, l'achèvement de la partie expérimentation de l'approche exige des données de test qui soient plus volumineux et qui soient plus représentatifs (échantillons extraits de processus métiers plus complexes : e-commerces, réseaux sociaux,...).

5.6 Conclusion

Dans ce chapitre, nous avons présenté un prototype logiciel qui implémente notre approche d'interrogation des données d'exécution des PM. L'architecture et les fonctionnalités du système ont été largement débattues et illustrés. Nous avons terminé le chapitre par la présentation de quelques résultats d'expérimentation.

Conclusion générale

Dans ce projet de fin d'études, nous avons abordé le problème d'interrogation des données d'exécutions des processus métiers, et nous avons proposé une approche permettant l'extraction et l'interrogation de ces données.

L'approche proposée est basée sur les modèles de données semi-structurées. Nous avons utilisé les AFD comme modèle de représentation des processus métiers et nous avons exploité le langage des requêtes XQuery pour l'interrogation des données d'exécution des processus métiers. Par ailleurs, nous avons implémenté un prototype logiciel qui mis en œuvre notre approche en utilisant l'environnement de programmation **Eclipse**.

En conclusion, je peux affirmé que durant la conduite de ce projet de fin d'études de Master, j'ai capitalisé les compétences suivantes :

- Sur le plan **méthodologique** : Je me suis familiarisée avec l'environnement de gestion de processus métiers **BonitaSoft**, ainsi que le langage des requêtes **XQuery**, qui m'ont servi pour réaliser mon travail.
- Sur le plan **scientifique** : J'ai appris à mener un projet et je me suis initié à la démarche et à la méthodologie scientifique qui consistent à comprendre le problème, analyser l'état de l'art, proposer une solution puis l'implémenter.
- Sur le plan **pratique** : J'ai approfondi mes connaissances opérationnelles par la maitrise du domaine des processus métiers en abordant les différentes phases de leur cycle de vie. Ainsi, la spécification du schéma, la création des instances pour ce processus, et enfin la génération des traces de ce processus ont été largement maitrisés.

En perspective à ce travail, il sera opportun de trouver une continuité à cette problématique de recherche par les futures promotions d'étudiants en vue de l'améliorer et de consolider l'approche par les tests comparatifs relatifs aux différents types de requêtes (*Relationnelles, semi-structurées et graphes*). D'autre part, il sera important de considérer d'autres modèles de représentation des processus métiers, tels que les réseaux de pétri et les graphes.

Bibliographie

[1] Jan vom Brocke and Michael Rosemann. *Handbook on Business Process Management 1 : Introduction, Methods, and Information Systems.* Springer Publishing Company, Incorporated, 1st edition, 2010.

[2] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Representing, analysing and managing web service protocols. *Data Knowl. Eng.*, 58(3) :327–357, September 2006.

[3] Data &amp ; knowledge engineering.

[4] Tadao Murata. Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :541–580, April 1989.

[5] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The (2Nd Edition).* Pearson Higher Education, 2004.

[6] G. Roy. *Conception de bases de données avec UML.* Presses de l'Université du Québec, 2007.

[7] Bill Curtis, Marc I. Kellner, and Jim Over. Process modeling. *Commun. ACM*, 35(9) :75–90, September 1992.

[8] Matjaz B. Juric. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2Nd Edition.* Packt Publishing, 2006.

[9] European Association of Business Process Management (EABPM), editor. *Business Process Management Common Body of Knowledge - BPM CBOK · Leitfaden für das Prozessmanagement.* Schmidt, Wettenberg, zweite edition, 2009.

[10] J.N. Gillot. *La gestion des processus métiers.* Jean-Noel Gillot, 2007.

[11] Ute Riemann. Benefits and challenges for business process management in the cloud. *Int. J. Organ. Collect. Intell.*, 5(2) :80–104, April 2015.

[12] Mathias Weske. *Business Process Management : Concepts, Languages, Architectures.* Springer Publishing Company, Incorporated, 1st edition, 2010.

[13] Ali Khebizi, Hassina Seridi-Bouchelaghem, Bouallem Benatallah, and Farouk Toumani. A declarative language to support dynamic evolution of web service business protocols. *Serv. Oriented Comput. Appl.*, 11(2) :163–181, June 2017.

[14] Daniel Deutch and Tova Milo. Type inference and type checking for queries on execution traces. *Proc. VLDB Endow.*, 1(1) :352–363, August 2008.

[15] Vania Marangozova-Martin and Generoso Pagano. Gestion de traces d'exécution pour le systèmes embarqués : contenu et stockage. Research Report RR-LIG-046, 2013.

[16] Richard Hull, Jianwen Su, and Roman Vaculin. Data management perspectives on business process management : Tutorial overview. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 943–948, New York, NY, USA, 2013. ACM.

[17] Michael Havey. *Essential Business Process Modeling.* O'Reilly Media, Inc., 2005.

[18] Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke, editors. *Business Process Management - 16th International Conference, BPM2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings*, volume 11080 of *Lecture Notes in Computer Science*. Springer, 2018.

[19] M Imlau, M Fally, H Coufal, G W Burr, and G T Sincerbox. Holography and data storage. In Frank Träger, editor, *Handbook of Lasers and Optics*, chapter Part D|20, pages 1205–1249. Springer-Verlag, Berlin-Heidelberg, 2007. ISBN-10 : 0-387-95579-8 ; ISBN-13 : 978-0-387-95579-7.

[20] E. F. Codd. Relational database : A practical foundation for productivity. *Commun. ACM*, 25(2) :109–117, February 1982.

[21] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language. *World Wide Web J.*, 2(4) :29–66, November 1997.

[22] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases*. O'Reilly Media, Inc., 2013.

[23] Alexis Martin, Generoso Pagano, Jérôme Correnoz, and Vania Marangozova-Martin. Analyse de systèmes embarqués par structuration de traces d'exécution. In Pascal Felber, Laurent Philippe, Etienne Riviere, and Arnaud Tisserand, editors, *ComPAS 2014 : conférence en parallélisme, architecture et systèmes*, Neuchâtel, Switzerland, April 2014.

[24] Daniel W. Barowy, Sumit Gulwani, Ted Hart, and Benjamin Zorn. Flashrelate : Extracting relational data from semi-structured spreadsheets using examples. *SIGPLAN Not.*, 50(6) :218–228, June 2015.

[25] Michael Benedikt and Christoph Koch. Xpath leashed. *ACM Comput. Surv.*, 41(1) :3 :1–3 :54, January 2009.

[26] Jim Melton. Sql, xquery, and sparql : What's wrong with this picture ? In *XTech 2006 : "Building Web 2.0"*, 2006.

[27] H. V. Jagadish, S. Al-Khalifa, A. Chapman, L. V. S. Lakshmanan, A. Nierman, S. Paparizos, J. M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. Timber : A native xml database. *The VLDB Journal*, 11(4) :274–291, December 2002.

[28] Mohammed Javeed Zaki, Jeffrey Xu Yu, Balaraman Ravindran, and Vikram Pudi, editors. *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II*, volume 6119 of *Lecture Notes in Computer Science*. Springer, 2010.

[29] Andreas Meyer, Sergey Smirnov, and Mathias Weske. Data in business processes. *EMISA Forum*, 31(3) :5–31, 2011.

[30] Jianmin Wang, Tao Jin, Raymond K. Wong, and Lijie Wen. Querying business process model repositories. *World Wide Web*, 17(3) :427–454, May 2014.

[31] Sherif Sakr and Ahmed Awad. A framework for querying graph-based business process models. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1297–1300, New York, NY, USA, 2010. ACM.

[32] Wil Van.Der. Aalst. *Process Mining : Discovery, Conformance and Enhancement of Business Processes*. 1st edition, 2011.

[33] Yutian Sun, Jianwen Su, Budan Wu, and Jian Yang. Modeling data for business processes. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1048–1059, 2014.

[34] Daniel Deutch and Tova Milo. Type inference and type checking for queries on execution traces. *PVLDB*, 1(1) :352–363, 2008.

[35] Asma Hassani and Sonia Ayachi Ghannouchi. Exploring the integration of business process with nosql databases in the context of BPM. In *Intelligent Systems Design and Applications - 17th International Conference on Intelligent Systems Design and Applications (ISDA 2017) Held in Delhi, India, December 14-16, 2017*, pages 771–784, 2017.

[36] Cagdas E. Gerede and Jianwen Su. Specification and verification of artifact behaviors in business process models. In *Service-Oriented Computing - ICSOC 2007, Fifth International Conference, Vienna, Austria, September 17-20, 2007, Proceedings*, pages 181–192, 2007.

[37] Fabio Casati, Malú Castellanos, Umeshwar Dayal, and Norman Salazar. A generic solution for warehousing business process data. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 1128–1137, 2007.

[38] Christian Fritz, Richard Hull, and Jianwen Su. Automatic construction of simple artifact-based business processes. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 225–238, 2009.

[39] Jörg Becker, Nico Clever, Justus Holler, and Maria Shitkova. Icebricks - business process modeling on the basis of semantic standardization. In *Design Science at the Intersection of Physical and Virtual Design - 8th International Conference, DESRIST 2013, Helsinki, Finland, June 11-12, 2013. Proceedings*, pages 394–399, 2013.

[40] Wolfgang Emmerich, Ben Butchart, Liang Chen, Bruno Wassermann, and Sarah L. Price. Grid service orchestration using the business process execution language (BPEL). *J. Grid Comput.*, 3(3-4) :283–304, 2005.

[41] Rajesh K. Thiagarajan, Amit K. Srivastava, Ashis K. Pujari, and Visweswar K. Bulusu. BPML : A process modeling language for dynamic business models. In *Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02), Newport Beach, California, USA, June 26-28, 2002*, pages 239–241, 2002.

[42] Christoph W. Kessler, Lu Li, Aras Atalar, and Alin Dobre. XPDL : extensible platform description language to support energy modeling and optimization. In *44th International Conference on Parallel Processing Workshops, ICPPW2015, Beijing, China, September 1-4, 2015*, pages 51–60, 2015.

[43] XPDL. In *Encyclopedia of Database Systems*, page 3665. 2009.

[44] Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. YAWL : yet another workflow language. *Inf. Syst.*, 30(4) :245–275, 2005.

[45] Elias Bayeh. The websphere application server architecture and programming model. *IBM Systems Journal*, 37(3) :336–348, 1998.

[46] Shili Yang, Ying Sun, Julie Waterhouse, Diana Lau, and Tammam Al-Hamwy. Modeling and implementing a business process using websphere lombardi edition 7.1. In *Proceedings of the 2010 conference of the Centre for Advanced Studies on Collaborative Research, November 1-4, 2010, Toronto, Ontario, Canada*, pages 374–375, 2010.

[47] Dave Thomas, editor. *ECOOP 2006 - Object-Oriented Programming, 20th European Conference, Nantes, France, July 3-7, 2006, Proceedings*, volume 4067 of *Lecture Notes in Computer Science*. Springer, 2006.

[48] OASIS. Web services business process execution language version 2.0. Technical report, 2007.

[49] www.delawareconsulting.be/index.aspx. Sap netweaver. Technical report.

[50] SAP. Mega solman. In *http ://sapinsider.wispubs.com/Assets/Articles/2009/August/*, Model-Your-Business-Processes-With-SAP-Solution-Manager.

[51] Nicolas Chabanoles, Philippe Ozil, and Mickey Farrance. Bonita BPM : an innovative bpm-based application development platform to build engaging, user-oriented business applications. In *Proceedings of the BPM Demo Session 2015 Co-located with the 13th International Conference on Business Process Management (BPM 2015), Innsbruck, Austria, September 2, 2015.*, pages 21–24, 2015.

[52] Jan Mendling, Jan Recker, and Johannes Wolf. Collaboration features in current BPM tools. *EMISA Forum*, 32(1) :48–65, 2012.

[53] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. Introduction to automata theory, languages, and computation, 2nd edition. *SIGACT News*, 32(1) :60–65, 2001.

[54] year = 2013 url = http ://blog.paumard.org/cours/java-api/ José Paumard, title = Java avancé en ligne.

[55] Giorgio Ghelli, Christopher Ré, and Jérôme Siméon. Xquery ! : An XML query language with side effects. In *Current Trends in Database Technology - EDBT 2006, EDBT 2006 Workshops PhD, DataX, IIDB, IIHA, ICSNW, QLQP, PIM, PaRMA, and Reactivity on the Web, Munich, Germany, March 26-31, 2006, Revised Selected Papers*, pages 178–191, 2006.

[56] Xinyu Wang, David Lo, and Emad Shihab, editors. *26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February 24-27, 2019*. IEEE, 2019.

[57] John Businge, Simon Kawuma, Moses Openja, Engineer Bainomugisha, and Alexander Serebrenik. How stable are eclipse application framework internal interfaces ? In *26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February 24-27, 2019*, pages 117–127, 2019.

[58] Douakha I. *Projet de fin d'études de Master, Analyse des données des processus métiers*. Université 8 Mai 1945 - Guelma- Algérie, Promotion 2018.