الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université 8 Mai 1945-Guelma-
Faculté de Mathématiques, d'Informatique et des Sciences de la matière
Département : d'Informatique

Mémoire de Fin d'études Master
Filière : Informatique
Option : Informatique Académique

Thème :

« BPEL Ontology : Une Ontologie pour un processus BPEL »

Encadré Par :
DJAKHDJAKHA Lynda

Présentée par :
Harrat Ali

Juin 2017

«Au nom d'Allah, le Clément, le Miséricordieux»

Remerciements

*Avant tout nous remercions **ALLAH**, le tout puissant, de m'avoir donné la force et la patience nécessaire pour achever ce travail de mémoire.*

Je souhaite exprimer mon sincères remerciements a mon encadreur le «Dr. DJAKHDJAKJA Lynda », quelle s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi que pour l'inspiration, l'aide et le temps qu'elle a bien voulu nous consacrer et sans le quel ce mémoire n'aurait jamais vu le jour.

Toute notre reconnaissance va particulièrement aux membres du jury pour l'intérêt qu'ils ont porté à notre projet en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

J'adresse mes plus sincères remerciements à ma famille : Ma mère, ma sœur, mon frère et tous mes proches et amis, qui m'ont accompagné, aidé, soutenu et encouragé tout au long de la réalisation de ce mémoire.

Je remercier toute personne qui a participé de près ou de loin à l'exécution de ce travail de mémoire.

Enfin, je remercie tous mes enseignants du début à la fin de mes études.

Résumé

De nos jours, la croissance importante de l'utilisation du Web sémantique comme un très fort paradigme pour la gestion des entreprises a conduit à la proposition de nouveaux thèmes de recherches.

Notre objectif est de combiner les recherches sur les processus métier de l'entreprise avec des recherches du web sémantique, en particulier d'enrichir la structure des processus métier avec une sémantique prise à partir des ontologies, d'autres termes, nous visons à donner des descriptions plus complètes et plus formelles aux processus BPEL, et nous présentons les règles de transformation du couple BPEL et son interface WSDL vers une ontologie OWL.

Sommaire

Introduction générale 6

1. Contexte général et problématique 7
2. Contributions et Objectifs 8
.3 Plan du mémoire 8

Chapitre 1/ Langages BPEL et WSDL 9

1. Introduction 10
2. Langage d'exécution du processus métier BPEL 10
   2.1. Historique 10
   2.2. Présentation du langage BPEL 10
   2.3. Caractéristiques du langage BPEL 12
   2.4. Modélisation du langage BPEL 12
   2.4.1. Comportement d'un processus BPEL 12
   2.4.2. Les constructeurs de déclaration 13
   2.4.3. Les constructeurs de définition du traitement 15
3. Le langage de description WSDL 16
   3.1. Présentation 16
   3.2. Les principes fondamentaux du langage WSDL 16
   3.3. Structure du document WSDL : 17
4. Conclusion 19

Chapitre 2/ Ontologies 20

1. Introduction 21
.2 Différentes notions du termeOntologie 21
3. Les composants d'une ontologie 22
4. Différents types d'ontologies 23
.5 Les principes de construction d'une ontologie 24
6. Langages et Outils pour les ontologies 25

6.1. Langages de spécifications ontologiques 25
6.2. Outils pour la manipulation des ontologies 28
7. Raisonnement sur les ontologies 29
8. Conclusion 30
Chapitre 3/ Travaux existants 31
1. Introduction 32
2. Travaux connexes 32
2.1. Le travail de Alexander Osterwalder et Yves 32
2.2. Le travail de Marc Ehrig, Agnes Koschmider et Andreas Oberweis 32
2.3. Le travail d'Oliver Thomas, Michael Fellmann 33
2.4. Le travail de Agata Filipowska, et al. 34
2.5. Le travail de Catriel Beeri Anat Eyal, Simon Kamenkovich 35
2.6. Le travail de Hans-Georg Fill and Patrik Burzynski 36
2.7. Le travail de Khalid Belhajjame, Marco Brambilla 37
2.8. Le travail de Martin Hepp, Dumitru Roman 38
2.9. Les travux de Nitzsche et Al 39
3. Synthèse 39
Chapitre 4/ Transformation du processus BPEL vers BPEL-Ontology 41
1. Introduction 42
2. Structure générale de notre démarche 42
3. Un méta-modèle pour BPEL et WSDL 43
4. Les règles de transformation 45
4.1. Un processus 45
4.2. Les partnerLinks 47
4.3. Les variables 49
4.4. Les faultHandlers 49
4.5. Les eventHandlers 50
4.6. CorrelationSets 50
4.7. Les activités du processus 51
4.8. Message, portType et operation 53
5. Instanciation de BPEL-Ontology 54
6. Conclusion 56
Chapitre 5/ Implémentation 57
1. Introduction 58

2. Présentation de l'environnement et les outils de développement 58
2.1. NetBeanIDE 8.0.2 58
2.2. Le langage JAVA 59
2.3. L'API JDOM 59
2.4. Éditeur Protégé 5.2.0 : 60
3. Architecture et description du système 61
3.1. Architecture du système 61
3.2. Description de notre travail 62
3.3. Espaces de nommage d'ontologie : 63
4. Scénario de transformation de BPEL vers OWL 63
4.1. 1ère Phase : l'implémentation de transformation : 64
4.2. 2ème Phase : Visualisation d'ontologie : 67
5. Évaluation de BPEL-ontologie 69
Conclusion 71
Conclusion Générale 72
Références 73
Annexes 78

Liste Des Figures

Figure 1.1: *BPEL-Businness Process Execution Language.* 11
Figure 1.2 : *Structure d'un processus BPEL.* 13
Figure 1.3 : *Les activités dans un processus BPEL.* 15
Figure 1.5 : *Structure d'un document WSDL.* 18
Figure 2.1 : *les types d' ontologies selon* 24
Figure 2.2 : *Architecture de langage d'ontologie.* 26
Figure 2.3 : *l'évolution des langages ontologiques.* 27
Figure 3.1 : *L' ontologie de modèle e-business* 32
Figure 3.2 : *Représentation de PetriNet en OWL-DL.* 33
Figure 3.3 : *Relation entre les modèles EPC et les ontologies* 34
Figure 3.4 : *Cadre organisationnel* 35
Figure 3.5 : *Modèle et le méta-modèle de l'ontologie.* 36
Figure 3.6 : *Génération de l'ontologie.* 38
Figure 3.7 : *Les sphères proposés* 39
Figure 4.1 : *Vue générale de notre démarche.* 42
Figure 4.2 : *Scénario d'exécution du processus de vente de Amor BenAmor.* 43
Figure 4.3 : *Structure générale d'un processus BPEL.* 44
Figure 4.4 : *Structure générale d'un fichier WSDL.* 44
Figure 4.5 : *Méta-Modèle du BPEL* 45
Figure 4.6 : *Modélisation des attributs d'un processus en protégé 2000* 46
Figure 4.7 : *Structure générale d'un processus BPEL.* 47
Figure 4.8 : *Modélisation des composants d'un processus.* 47
Figure 4.9 : *Représentation de l'élément PartnerLinks et ses relations.* 48
Figure 4.10 : *Exemple d'instanciation de l'ontologie.* 56
Figure 5.1 : *Interface Netbeans IDE.* 59
Figure 5.2 : *Architecture de l'application BPEL Ontologie.* 61
Figure 5.3 : *L'interface générale de l'application.* 64
Figure 5.4 : *Importation du code BPEL.* 65
Figure 5.5 : *Le champ pour visualiser le code BPEL.* 65
Figure 5.6 : *Le champ pour visualiser l'arborescent du code BPEL.* 66
Figure 5.7 : *importation du code WSDL.* 66
Figure 5.8 : *Le champ pour visualisée du code OWL.* 67
Figure 5.9 : *enregistrement du code OWL.* 67
Figure 5.10 : *L'importation d'une ontologie.* 68
Figure 5.11 : *La visualisation du graphe de l'ontologie.* 69
Figure 5.12 : *Sélection du raisonneur.* 70
Figure 5.13 : *Vérification de la classification Consistante et inconsistante.* 70

Liste Des Abréviations et Acronymes

WSDL	Web Services Description Language
BPEL	Business Process Execution Language.
SOA	Service Oriented Archetecture
IBM	International Business Machines Corporation
XLANG	XML Business Process Language
BEA	Bill Coleman, Edward Scott et Alfred Chuang
WSFL	Web Services Flow Language
SAP	Systems, Applications and Products for data processing
OASIS	Open Architecture System Integration Strategy.
SOAP	Simple Object Access Protocol
RDF	Resource Description Framework
RDFS	Resource Description Framework Schéma
KIF	Knowledge Interchange Format
RIF	Rule Interchange Format
DAML	DARPA Agent Markup Language
OIL	Ontology Inference Layer
UML	Unified Modeling Language
WSML	Web Service Modeling Language
SBPM	Semantic Business Process Management
BPMN	Business process model and notation
XML	Extensible Markup Language

W3C	World Wide Web Consortium
UDDI	Universal Description Discovery and Integration
DOM	Document Object Model
JDOM	Java Document Object Model
HTML	HyperText Markup language
OWL	Web Ontology Langage
IDE	Integrated Development Environment

Sommaire

Introduction générale 6

1. Contexte général et problématique 6
2. Contributions et Objectifs 7
.3 Plan du mémoire 7

Chapitre 1/ Langages BPEL et WSDL 9

1. Introduction 10
2. Langage d'exécution du processus métier BPEL 10
   2.1. Historique 10
   2.2. Présentation du langage BPEL 10
   2.3. Caractéristiques du langage BPEL 12
   2.4. Modélisation du langage BPEL 12
   2.4.2. Les constructeurs de déclaration 13
   2.4.3. Les constructeurs de définition du traitement 15
3. Le langage de description WSDL 16
   3.1. Présentation 16
   3.2. Les principes fondamentaux du langage WSDL 16
   3.3. Structure du document WSDL : 17
4. Conclusion 18

Chapitre 2/ Ontologies ..... **Erreur ! Signet non défini.**

1. Introduction 21
.2 Différentes notions du terme Ontologie 21
3. Les composants d'une ontologie 22
4. Différents types d'ontologies 23
.5 Les principes de construction d'une ontologie 24
6. Langages et Outils pour les ontologies 25
   6.1. Langages de spécifications ontologiques 25
   6.2. Outils pour la manipulation des ontologies 28
7. Raisonnement sur les ontologies 29

Chapitre 3/ Travaux existants ..... **Erreur ! Signet non défini.**
1. Introduction 32
2. Travaux connexes 32
2.1. Le travail de Alexander Osterwalder et Yves 32
2.2. Le travail de Marc Ehrig, Agnes Koschmider et Andreas Oberweis 32
2.3. Le travail d'Oliver Thomas, Michael Fellmann 33
2.4. Le travail de Agata Filipowska, et al. 34
2.5. Le travail de Catriel Beeri Anat Eyal, Simon Kamenkovich 35
2.6. Le travail de Hans-Georg Fill and Patrik Burzynski 36
2.7. Le travail de Khalid Belhajjame, Marco Brambilla 37
2.8. Le travail de Martin Hepp, Dumitru Roman 38
2.9. Les travux de Nitzsche et Al 39
3. Synthèse 39
Chapitre 4/ Transformation du processus BPEL vers BPEL-Ontology 41
1. Introduction 42
2. Structure générale de notre démarche 42
3. Un méta-modèle pour BPEL et WSDL 43
..... 45
4. Les règles de transformation 45
4.1. Un processus 45
4.2. Les partnerLinks 47
4.3. Les variables 49
4.4. Les faultHandlers 49
4.5. Les eventHandlers 50
4.6. CorrelationSets 50
4.7. Les activités du processus 51
4.8. Message, portType et operation 53
5. Instanciation de BPEL-Ontology 54
6. Conclusion 56
Chapitre 5/ Implémentation 57
1. Introduction 57
2. Présentation de l'environnement et les outils de développement 58
2.1. NetBeanIDE 8.0.2 58
2.2. Le langage JAVA 59

2.3. L'API JDOM 59
2.4. Éditeur Protégé 5.2.0 : 60
3. Architecture et description du système 61
3.1. Architecture du système 61
3.2. Description de notre travail 62
3.3. Espaces de nommage d'ontologie : 63
4. Scénario de transformation de BPEL vers OWL 63
4.1. 1ère Phase : l'implémentation de transformation : 64
4.2. 2ème Phase : Visualisation d'ontologie : 67
5. Évaluation de BPEL-ontologie 69
Conclusion 72
Références 73
Annexes 78

Tables des figures

Figure 1.1: *BPEL-Businness Process Execution Language.* 11
Figure 1.2 : *Structure d'un processus BPEL.* 13
Figure 1.3 : *Les activités dans un processus BPEL.* 15
Figure 1.5 : *Structure d'un document WSDL.* 18
Figure 2.1 : *les types d' ontologies selon* 24
Figure 2.2 : *Architecture de langage d'ontologie.* 26
Figure 2.3 : *l'évolution des langages ontologiques.* 27
Figure 3.1 : *L' ontologie de modèle e-business* 32
Figure 3.2 : *Représentation de PetriNet en OWL-DL.* 33
Figure 3.3 : *Relation entre les modèles EPC et les ontologies* 34
Figure 3.4 : *Cadre organisationnel* 35
Figure 3.5 : *Modèle et le méta-modèle de l'ontologie.* 36
Figure 3.6 : *Génération de l'ontologie.* 38
Figure 3.7 : *Les sphères proposés* 39
Figure 4.1 : *Vue générale de notre démarche.* 42
Figure 4.2 : *Scénario d'exécution du processus de vente de Amor BenAmor.* 47
Figure 4.3 : *Structure générale d'un processus BPEL.* 48
Figure 4.4 : *Structure générale d'un fichier WSDL.* 42
Figure 4.5 : *Méta-Modèle du BPEL* 49
Figure 4.6 : *Modélisation des attributs d'un processus en protégé 2000* 50
Figure 4.7 : *Structure générale d'un processus BPEL.* 51
Figure 4.8 : *Modélisation des composants d'un processus.* 51
Figure 4.9 : *Représentation de l'élément PartnerLinks et ses relations.* 52
Figure 4.10 : *Exemple d'instanciation de l'ontologie.* 60
Figure 5.1 : *Interface Netbeans IDE.* 61
Figure 5.2 : *Architecture de l'application BPEL Ontologie.* 62
Figure 5.3 : *L'interface générale de l'application.* 64
Figure 5.4 : *Importation du code BPEL.* 65
Figure 5.5 : *Le champ pour visualiser le code BPEL.* 69
Figure 5.6 : *Le champ pour visualiser l'arborescent du code BPEL.* 70
Figure 5.7 : *importation du code WSDL.* 70
Figure 5.8 : *Le champ pour visualisée du code OWL.* 70
Figure 5.9 : *enregistrement du code OWL.* 71
Figure 5.10 : *L'importation d'une ontologie.* 71
Figure 5.11 : *La visualisation du graphe de l'ontologie.* 73
Figure 5.12 : *Sélection du raisonneur.* 74
Figure 5.13 : *Vérification de la classification Consistante et inconsistante.* 74

Introduction Générale

Introduction générale

1. Contexte général et problématique

Ces dernières années ont vu l'émergence d'architectures logicielles fondées sur les services qui visent à mettre en place des processus métier performants ainsi que des systèmes d'information constitués de services applicatifs indépendants et interconnectés. Ces architectures sont connues sous le nom d'Architectures Orientées Services (SOA). Elles facilitent l'exposition, l'interconnexion et la réutilisation d'applications à base de services. L'architecture SOA signifie que les processus métier sont définis comme une collection d'activités servant à invoquer des services et à produire des résultats métier[1]. Dans le cadre de la mise en œuvre des architectures SOA, le BPEL (Business Process Execution Langage) est largement utilisé. Il permet de décrire un processus exécutable et s'appuie sur le langage WSDL (Web Service Description Langage) du fait que chaque processus BPEL est exposé comme un service Web via une interface WSDL[1]. Les premiers objectifs de BPEL étaient de : supprimer les conflits liés à la composition et l'intégration de services dans des processus d'affaires et de définir un langage standard de spécification et de normalisation de ses processus [2].

Depuis l'émergence de l'architecture SOA, avec ses différentes technologies, les communautés industrielles ont mis l'accent sur la fourniture d'outils et la proposition des nouvelles recherches permettant l'interopérabilité et l'intégration flexible des applications des entreprises.

Les ontologies sont considérées toujours comme la solution la plus adéquate pour le problème d'interopérabilité. Elles permettent de spécifier de façon formelle la connaissance partagée entre plusieurs utilisateurs. L'objectif d'une telle modélisation basée sur les ontologies est de représenter la sémantique des connaissances modélisées.

Dans le but d'améliorer la manipulation des connaissances représentées par les ontologies, le consortium W3C propose le langage d'ontologie Web (OWL) afin de permettre aux applications de réaliser des raisonnements plus complexes [3] par des classes et de types de propriétés. Il a l'avantage de donner de sens aux classes et relations définies entre celles-ci, de manière plus expressive. Il apporte une meilleure intégration, une évolution, un partage et une inférence plus facile des ontologies et offre un cadre unificateur et fournit des primitives pour

améliorer la communication entre les personnes, entre les personnes et les processus et entre les processus [4].

2. Contributions et Objectifs

Le présent travail consiste à exploiter la flexibilité apportée par les ontologies. Il sert à introduire la sémantique prise par ces dernières aux processus BPEL et de décrire formellement les connaissances caractérisant ses processus à l'aide des primitives OWL et l'établissement d'une ontologie BPEL-Ontology à partir de codes BPEL et WSDL.

Notre but vise à améliorer la communication, le partage et même l'interopérabilité entre différents systèmes dans différentes entreprises, ainsi de faire plus tard des raisonnements sur les processus métier qui seront redu possibles à travers les ontologies.

Donc, dans ce mémoire :

- Nous proposons un ensemble de règles pour faire la transformation du processus BPEL et son interface WSDL vers une ontologie,
- Nous proposons un algorithme pour la création et l'instanciation de l'ontologie,
- L'implémentation du module de transformation,
- La vérification de l'ontologie résultat à travers des tests de cohérence et de consistance.

3. Plan du mémoire

Dans ce mémoire, nous essayons dans les deux premiers chapitres de dresser un état de l'art sur les différents concepts de base liées à notre travail. Nous terminons cet état par une synthèse sur les travaux existants dans la littérature. Les deux derniers chapitres seront consacrés à notre proposition et sa réalisation.

Le premier chapitre, présente le langage BPEL et WSDL. La structure générale et les différents éléments de chaque spécification seront exposés.

Dans le deuxième chapitre, nous allons présenter la notion d'ontologie par différentes définitions. Ainsi, plusieurs formalismes et langages de spécification d'ontologie seront exposés, avec une description un peu détaillée du langage OWL. Nous terminons ce chapitre par la présentation de quelques outils de développement d'ontologies.

Le troisième chapitre, présente un ensemble de travaux connexes à notre contexte. Le chapitre se terminera avec une synthèse.

Le quatrième chapitre sera consacré entièrement à la présentation de notre contribution. On va présenter les règles de transformation à partir des fichiers BPEL et WSDL vers l'ontologie, et l'illustration des règles proposées par un ensemble d'exemples basés sur des processus réels. Nous proposons ainsi un algorithme pour la création et l'instanciation de l'ontologie.

Le cinquième chapitre est destiné à la réalisation d'une implémentation appliquant les l'ensemble des règles proposées dans le quatrième chapitre. Nous avons développé une application JAVA qui nous permet la transformation du fichier BPEL vers un fichier OWL. Ensuite, nous appliquons le résultat de la transformation pour faire vérifier la consistance et la cohérence de l'ontologie résultat à l'aide de Protégé.

CHAPITRE 1

Langages BPEL et WSDL

1. Introduction

Le processus métier de l'entreprise sera défini comme une collection d'activités servant à invoquer des services et à produire des résultats métier [5]. Cependant, il faut disposer d'un langage qui, étant *spécialisé* et *standardisé,* permet une telle définition. Un des langages possédant ces propriétés est le langage BPEL qui est considéré comme le langage dominant dans sa catégorie vu sa large adoption dans l'industrie. Dans [8], un processus BPEL est considéré comme un ensemble de services Web via une interface WSDL [8]. De ce fait, nous présentons dans ce chapitre un aperçu du processus exécutable BPEL et l'interface WSDL.

2. Langage d'exécution du processus métier BPEL

2.1. Historique

BPEL est largement utilisé dans le cadre de la mise en œuvre d'une architecture orientée services. Il a été créé en 2002 par IBM[1], BEA et Microsoft [6] qui ont fusionné leurs normes existantes WSFL (Web Services Flow Language) [7] et XLANG[2]. Le premier (WSFL) a été conçu par IBM et vise à décrire les compositions de services sous forme de *graphes* d'activités de nature *orientée* et *acyclique,* alors que le deuxième (XLANG) a été conçu par Microsoft et vise à décrire de telles compositions selon une approche *structurée* basée sur des constructeurs représentant des flots séquentiels et parallèles [8][9]. Une édition révisée avec la contribution de SAP[3] et Siebel a été publiée en mars 2003 sous le nom de BPEL 1.1. Cette version a également été soumise à OASIS[4] en avril 2003 pour la normalisation. S'est imposé comme le langage standard OASIS d'orchestration de services Web, et a été défini dans sa version 2.0 par une spécification du consortium OASIS à la fin du mois de mars 2007 [10].

Initialement BPEL était connu sous le nom de BPEL4WS : « Business Process Execution Langage for Web Services ». Comme c'était absolument imprononçable, l'appellation fut modifiée en WS-BPEL.

2.2. Présentation du langage BPEL

Web Services Business Process Exécution Language (WS-BPEL ou BPEL) est un langage standard basée sur le format XML qui est généralement utilisé pour définir une spécification

[1] International Business Machines Corporation, connue sous l'abréviation IBM, est une société multinationale américaine.
[2] XML Business Process Language.
[3] Systems, Applications and Products for data processing.
[4] Open Architecture System Integration Strategy.

(ou une exécution) d'une orchestration du service Web. Il nous permet de définir un ensemble de services Web de deux façons : Abstract ou Exécutable, pour décrire des processus métier complexes qui peuvent interagir de manière synchrone ou asynchrone avec leurs partenaires.

Les processus abstraits **:** Ils représentent une vue ou une abstraction du comportement d'un processus BPEL. Ils permettent de cacher une partie du processus BPEL à déployer. Ils ne sont pas possible de les exécuter.

Les processus exécutable **:** Spécifient les détails d'interactions entre les services qu'ils composent, spécifient les algorithmes exacts contrôlant les activités des compositions à définir, sans oublier de spécifier les messages entrants et sortants échangés par ces activités. La définition de tels processus suit le paradigme de l'orchestration et pourrait être exécutée par un moteur BPEL. Dans la majorité des cas, c'est ce type de processus qui est spécifié par le BPEL [11].

Un processus synchrone : une relation synchrone s'exécute depuis une activité du processus BPEL retournant directement une réponse à l'un de ses partenaires en bloquant l'exécution de ce dernier placé dans l'attente d'une réponse à transmettre. Cette relation est généralement réservée aux processus BPEL exécutés sur des délais de courtes durées [12].

Un processus asynchrone : une relation asynchrone ne bloque pas l'exécution du partenaire car ce dernier n'est pas dans un état d'attente d'une réponse immédiate. Le mécanisme de rappel ou de « calback » en anglais répond à la nécessité de retransmettre directement un message suite à l'invocation simulant une relation synchrone. Cette relation est généralement utilisée avec des processus BPEL exécutés sur des délais important éventuellement interrompus temporairement [12].

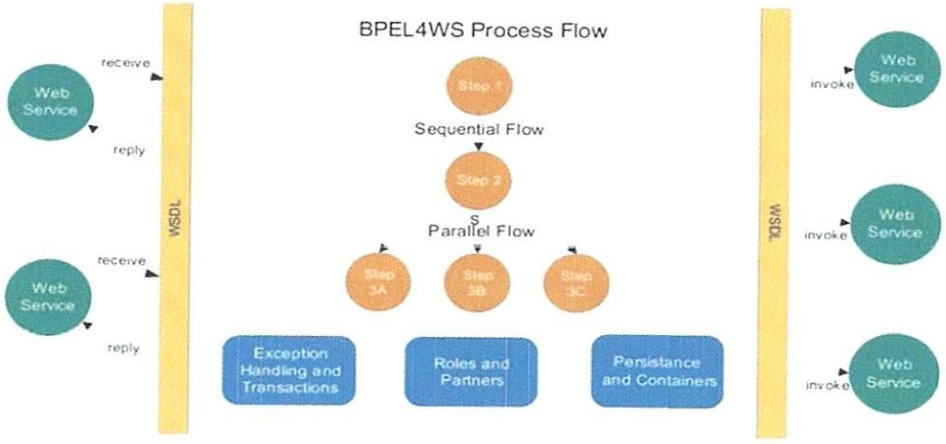


Figure 1.1: *BPEL-Businness Process Execution Language.*[13]

2.3. Caractéristiques du langage BPEL

Un processus BPEL est exécuté directement par un moteur d'orchestration de BPEL[14]. BPEL se caractérise par rapport aux autres langages (d'orchestration) par :

- Sa gestion des exceptions, en particulier, des fautes et des événements ;
- L'exécution parallèle des activités et la synchronisation des flots ;
- La description des transactions contextuelles et de longue durée ;
- Son mécanisme de compensation qui est très utile pour les transactions de longue durée ;
- Sa gestion de la corrélation des messages.

2.4. Modélisation du langage BPEL

2.4.1. Comportement d'un processus BPEL

Un processus BPEL décrit les activités, les interactions et les échanges d'informations entre les services web définis comme « partenaires » du processus BPEL, et est lui-même considéré comme un service web. Chaque processus BPEL est définit en combinant les éléments du langage dans un document XML. Ce document est ensuite soumis à l'interprétation d'un système informatique nommé « moteur BPEL » assurant l'exécution du processus BPEL. Chaque instance de processus est créée en utilisant la définition XML comme modèle [12].

La racine d'un tel document correspond toujours à l'élément <process> qui décrit son comportement de façon opérationnelle. Cet élément pourrait disposer de plusieurs attributs. Parmi ces attributs, on cite l'attribut abstractProcess qui permet de spécifier si le processus est défini comme processus abstrait ou exécutable [15].

Comme le montre la figure suivante une définition d'un processus *BPEL* consiste dans les éléments suivants :

- Les déclarations des éléments qui vont être utilisés par ce processus tels partnerLi
  variables, etc. éléments qui seront manipulés par le processus au niveau de
  interactions avec ses partenaires,
- La description du comportement du processus (workflow)
  activités primaires pour exécuter des opérations de
  exemple, activités séquentielles, concurrentes, etc. p
  base.

- *BPEL* fournit aussi un autre type d'activités pour gérer les globale ou locale. [16]

```
<process name=... >
        ...
      <import ...> importation des déclarations des services Web partenaires<
          <partnerLinks>?
          <partnerLink name=.../>+ déclaration des liens vers les services Web partenaires
          </partnerLinks>
          ...
          <variables>?
          <variable name=.../>+ les variables de l'état interne du processus BPEL
          </variables>
          ...
          <faultHandlers>? déclaration du gestionnaire d'erreurs internes</faultHandlers>
          <eventHandlers>? déclaration du gestionnaire d'événements </eventHandlers>
                Activity  spécification du comportement interne du processus BPEL
</process>
```

Figure 1.2 : *Structure d'un processus BPEL.* [11]

2.4.2. Les constructeurs de déclaration

Ces constructeurs instaurent une déclaration *globale* s'ils sont directement liés au processus. En revanche, si de tels constructeurs sont liés à une des unités logiques composant ce dernier, dans ce cas on dit que ces constructeurs instaurent une déclaration *locale.* Dans les deux cas, ces constructeurs se définissent comme suit :

- **La balise < import >** Cette balise permet d'importer un fichier WSDL [2].
- **La balise des liens partenaires < partnerLinks >**

Les partnerLinks permettent de regrouper les liens partenaires considérés comme intermédiaire entre les activités du processus BPEL et les opérations des services web mis à disposition des partenaires.

Chaque lien partenaire est défini avec l'élément <partnerLink> est typé par un <partnerLinkType> et un rôle. Cet élément assure la transmission correcte des messages en associant les partenaires et le processus BPEL [12].

- **La balise <variables>**

Les variables permettent de garder trace des données représentant l'état interne du processus. Dans la majorité des cas, ces données correspondent aux messages échangés entre

le processus et ses partenaires[15]. Une variable est définie avec un nom, une valeur et un type de donnée. La valeur est affectée d'une valeur durant l'exécution du processus BPEL[12]. Ces variables sont déclarées avec un type qui peut être : - soit un type message de WSDL, - soit un type simple en XML Schéma, - soit un type élément (complexe) en XML Schéma.

Les variables sont affectées soit explicitement par l'élément <assign>, ou par des entrées / sorties par échange de messages. [17]

- **La balise <correlationSets>**

Ils permettent de regrouper l'ensemble de corrélation correspondant à un mécanisme assurant l'échange de données entre toutes les instances de processus BPEL en cours d'exécution et les messages échangés. Le mécanisme d'ensemble de corrélations définit les identifiants communs entre le processus et les messages provenant des partenaires[12]. Pour ce faire, chaque <correlationSet> définit un groupe de propriétés permettant d'identifier, d'une façon unique, chaque instance d'un même processus.

- **Les gestionnaires d'exception < faultHandlers >**

Les gestionnaires de fautes peuvent être associés à l'élément racine <process> d'un processus BPEL, à une activité <scope> ou à une activité <invoke> afin de gérer les fautes d'invocation. Ils permettent d'annuler le travail qui n'a pas pu être achevé dans un processus où une erreur a eu lieu [10]. Elle compose d'activités <catch>, dont chacune permet de capturer une sorte spécifique d'erreurs. On peut également avoir une clause <catchAll> qui peut attraper toutes les erreurs non interceptées par les autres catchs. Une erreur est définie par un nom unique et une variable pour la donnée associée à cette erreur[16].

- **Les gestionnaires d'événements < eventHandlers>**

Ils permettent au processus BPEL ou une activité <scope> d'associé à un ensemble d'événements qui peuvent être invoqués en parallèle avec leur activité principale. Ces gestionnaires d'événements permettent de prendre en charge les événements qui se produisent de manière asynchrone pendant l'exécution du processus BPEL. On distingue deux types d'evenements : <onEvent> (peuvent être des messages entrants qui correspondent à une opération WSDL) et <onAlarm> (déclenchement d'une alarme)[10].

2.4.3. Les constructeurs de définition du traitement

La majorité des structures d'un processus BPEL sont des activités. Les activités, pouvant être effectuées par un processus BPEL, sont classées en deux catégories : activités basiques (ou de base) et activités structurées. Chaque activité comporte cinq attributs qui ont la signification suivante [18]:

- **partnerLink :** Contient le nom du lien partenaire à utiliser. Le moteur d'exécution BPEL utilise ce nom pour identifier la destination de partenaire actuelle.
- **portType :** La valeur donnée indique le type de port WSDL du service cible qui contient l'opération à invoquer.
- **opération :** Indique l'opération WSDL à invoquer.
- **variable d'entrée :** La variable d'entrée contient les données à envoyer au service.
- **variable de sortie :** La variable de sortie est initialisée avec la réponse reçue du partenaire.

Ci-dessous, nous exposons l'ensemble de ces activités.

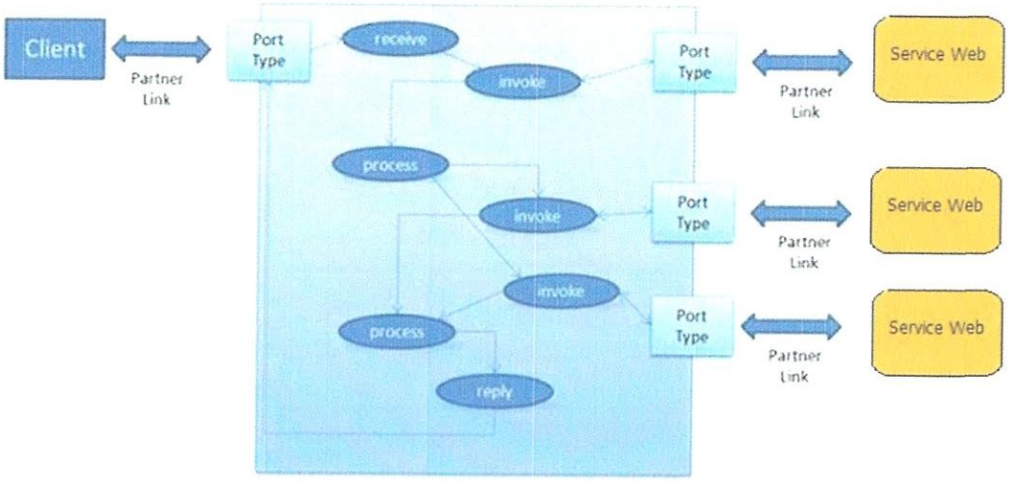


Figure 1.3 : *Les activités dans un processus BPEL.* [1]

- **Les activités de base**

Parmi les activités de base les plus importantes, on trouve celles qui sont liées à la réception et à l'envoi des messages, respectivement, de la part et à destination des partenaires. Les activités sont : <receive>, <reply>, <invoke>, <empty>, <throw>, <exit>, <assign> et <wait>, <rethrow>.

- **Les activités structurées**

Les activités structurées décrivent l'ordre d'exécution de leurs sous-activités. Les activités <sequence>, <if-else >, et l'activité <flow> permet quant à elle une exécution parallèle de ses activités ainsi que leur synchronisation. Enfin, l'activité <pick> définit un choix contrôlé par l'arrivée d'un événement.

- **Activités répétitives**

BPEL offre des activités permettant une exécution répétitive en utilisant des boucles. Depuis la version 2.0 de WS-BPEL, il existe 3 types de boucles qui sont : while, forEach et repeatUntil[17].

3. Le langage de description WSDL

3.1. Présentation

WSDL est le premier langage de description des services web en décrivant les actions offertes par les services, basé sur XML. Depuis 2007, sa version 2.0 est une recommandation officielle du W3C. Il offre une manière standard de construire des descriptions détaillées de service web. Chaque description se présente sous forme de fichier textuel structuré et ayant un encodage XML. Les différentes entreprises peuvent publier des descriptions WSDL pour les services qu'ils fournissent. Des liens vers ces descriptions WSDL sont offerts dans les profils des sociétés présentes dans les registres UDDI.

En utilisant la description WSDL d'un service web, un client peut invoquer n'importe quelle opération fournie par le service alors qu'il n'a aucune connaissance préalable sur ce service[19]. La description WSDL d'un service web est indépendante de toutes plateformes ou technologies particulières, ce qui est un point important pour assurer l'interopérabilité des services.

3.2. Les principes fondamentaux du langage WSDL

WSDL repose sur sept principes fondamentaux [20].

1 - **Extensibilité** WSDL permet la spécification de plusieurs attributs via des extensions, qui peuvent être définies et branchées dans de nombreuses structures centrales de WSDL.

2 - **Prise en charge des systèmes de type multiple** WSDL permet la spécification des données du service web dans des systèmes de type arbitraire.

3 - **Unification messagerie et RPC** sont deux possibilités pour résoudre le problème d'intégration.

- **Séparation de " Quelle " de " Comment " et " Où "** Les documents WSDL sont divisés en deux parties : une partie abstraite, décrivant ce que le service fait en termes généraux et une partie concrète, décrivant où se trouve le service et comment l'appeler exactement. Cette séparation permet d'offrir un service à différents endroits et d'utiliser différentes méthodes d'invocation.
- **Prise en charge de multiples protocoles et transports** Le protocole des messages ou des appels envoyés à un service Web n'est pas dicté par WSDL. Il permet la spécification de formats de fil arbitraires.
- **Pas de commande** WSDL n'autorise pas la spécification d'un protocole d'interaction avec le service.

3.3. Structure du document WSDL :

Un document WSDL, est composé d'une suite d'éléments décrivant un service sous forme d'un ensemble d'opérations exploitables depuis l'extérieur. Il se compose de deux parties indépendantes[20] :

- **La partie concrète (**La partie physique**)** est constituée de la description des protocoles d'accès au service web (information particulière à un service). Ce niveau est seulement utilisé lors de l'invocation des méthodes du Service Web. Les trois éléments concrets XML présents dans un WSDL sont :
  - <wsdl:binding> : protocole et format de données spécifiques pour un type de port précis.
  - <wsdl:service> : une collection des ports d'accès du service.
  - <wsdl:port> : une combinaison d'une adresse réseau et d'un binding qui spécifie une liaison d'un « PortType » à un protocole SOAP.

- **La partie abstraite** (La partie logique) regroupe les informations pouvant être réutilisées. Cette partie est utilisée principalement lors du processus de sélection du service. Les quatre éléments abstraits XML qui peuvent être définis dans un WSDL sont :
  - <wsdl:types> : un conteneur de la définition de données utilisées.
  - <wsdl:message> : description des types de données utilisées lors de l'invocation (et de la réponse) d'une opération.
  - <wsdl:operation> : description d'une des actions supportées par le Service Web.
  - <wsdl:portType> : description de l'ensemble des actions supportées.

L'intérêt d'avoir ces deux parties, est que la partie concrète propose une ou plusieurs réalisations de la partie abstraite. Les documents WSDL ne sont jamais générés par des développeurs, mais le sont grâce à des outils qui automatisent la tâche (par exemple, il existe des outils qui prennent une classe Java et qui créent le WSDL correspondant).

Le schéma suivant illustre les deux partie abstraite et concrète d'un document wsdl :

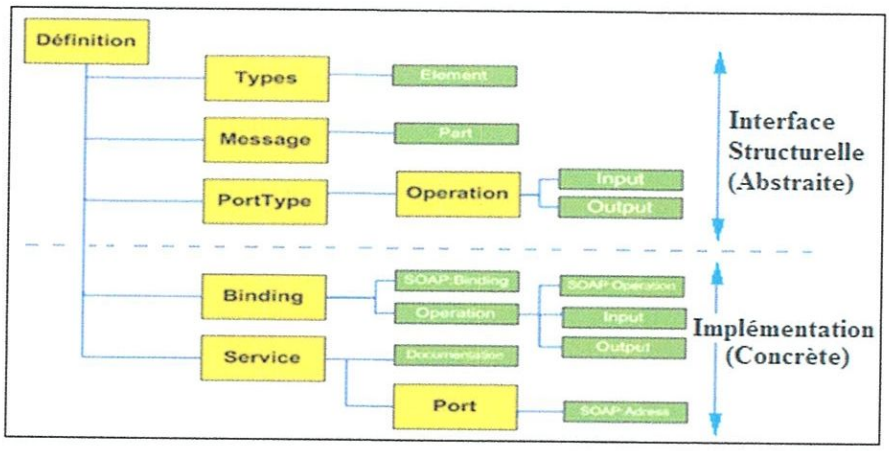


Figure 1.4 : *Structure d'un document WSDL.* [21]

Un document WSDL peut également contenir d'autres éléments, comme les éléments d'extension et un élément de service qui permet de regrouper les définitions de plusieurs services Web dans un seul document WSDL. Il sert à décrire :

- Le protocole de communication (SOAP RPC ou SOAP orienté message)
- Le format de messages requis pour communiquer avec ce service
- Les méthodes que le client peut invoquer ;
- La localisation du service.

4. Conclusion

A partir de ce chapitre, Nous nous sommes retenu que BPEL permet de décrire un processus exécutable décrivant une orchestration de services ou un processus abstrait spécifiant les échanges de messages entre différents partenaires. Il s'appuie sur le langage WSDL puisque chaque processus BPEL est exposé comme un service Web via une interface WSDL. Il utilise les opérations, les données ainsi que les liens des partenaires décrits dans son interface WSDL. Ce dernier décrit aussi tous les éléments nécessaires à un processus BPEL pour interagir avec ses partenaires, à savoir, l'adresse des partenaires, les protocoles de communication et les opérations disponibles.

L'objet du prochain chapitre (deuxième chapitre) est la présentation des ontologies.

CHAPITRE 2

Ontologies

1. Introduction

Depuis plusieurs années, les ontologies occupent une place majeure dans la représentation et la modélisation des connaissances. Elles sont utilisées pour formaliser les connaissances d'un domaine et ainsi ajouter une couche sémantique aux systèmes et aux applications informatiques. Les ontologies permettent de représenter de manière explicite les connaissances d'un domaine au moyen d'un langage formel afin qu'elles puissent être manipulées automatiquement et partagées aisément. [Ref]

Dans ce chapitre nous allons présenter les différentes notions liées aux ontologies. Nous commençons d'abord par donner quelques définitions courantes d'ontologies, avant de décrire leurs différents composants puis de présenter leur typologie. Enfin, définir quelques langages utilisés pour la représentation des ontologies au sein du Web sémantique

2. Différentes notions du terme Ontologie

A l'origine, l'ontologie est une notion philosophique. Les chercheurs du Web sémantique ont adapté ce terme dans leur propre jargon et diverses définitions d'ontologie existent [22], [23], [24], [25] et [26] :

Dans [23], Tom GRUBER définie une ontologie comme « *une spécification explicite d'une conceptualisation* ». Cette définition est considérée par une large partie de la communauté du web sémantique comme étant référentielle, où la notion de conceptualisation est une vision du monde-cible qui se fonde sur un ensemble de concepts universaux et d'autres particuliers au monde-cible.

Une définition plus détaillée a été donné dans [25], où une ontologie est défini comme une description formelle explicite des concepts dans un domaine du discours (classes appelées parfois concepts), des propriétés de chaque concept décrivant des caractéristiques et attributs du concept (attribut appelés parfois rôles ou propriétés) et des restrictions sur les attributs (facettes appelées parfois restrictions de rôles).

Dans [26], Mizoguchi insiste sur la nature *conceptuelle d'une ontologie*, faite pour pouvoir partager des connaissances entre homme et systèmes, et entre systèmes. Il considère *conceptualisation, partage* et *réutilisation* comme les concepts-clés d'une ontologie, ce qui représente les attentes de la communauté d'intelligence artificielle sur les ontologies.

En informatique et en particulier en science de l'information, une ontologie désigne *un ensemble structuré des termes et concepts ayant une sémantique dans un domaine donné*.

Donc une ontologie constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Elle est employée pour raisonner à propos des objets du domaine concerné. Les concepts sont organisés en un graphe dont les relations peuvent être des relations sémantiques ou des relations de subsomption. Ce type de graphe est appelé le schéma d'ontologies [27].

3. Les composants d'une ontologie

Dans la plupart des références, une ontologie est constituée d'un ensemble de primitives représentées par les cinq éléments suivants : Les concepts, les relations, les axiomes, les fonctions et les instances.

- **Les concepts (classes)** Sont des notions (ou objets) permettant la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement dans une ontologie, etc.

Un concept peut être abstrait ou concret, élémentaire ou composé, réel ou fictif. Habituellement, les concepts sont organisés en taxonomie. Une taxonomie est une hiérarchie de concepts (ou d'objets) reliés entre eux en fonction de critères sémantiques particuliers[28].

- **Les relations :** Les relations d'une ontologie désignent les différentes interactions et corrélations entre les concepts de l'ontologie.

Ces relations englobent les associations suivantes : Sous classe de (spécification ou généralisation), partis de (agrégation ou composition), associé a, instance de, est un ... etc.

Un ou plusieurs termes peuvent être associés à une relation en tant qu'étiquettes, tout comme un ensemble de règles spécifiant les propriétés logiques de la relation (la transitivité, la symétrie, la fonctionnalité, etc.)[29].

- **Les fonctions :** soit les cas particuliers de relations dans lesquelles un élément de la relation, le (x-ième), est défini en fonction des (x-1) éléments précédents. Formellement, les fonctions sont définies ainsi, F : C1xC2x … xCn-1, Cn. Comme exemple de fonction binaire, nous avons la fonction « mère de »[30].
- **Les axiomes ou règle d'inférence :** Les axiomes de l'ontologie permettent de définir la sémantique des termes (classes, relations), leurs propriétés et toutes contraintes quant à leur interprétation. Ils sont définis à l'aide de formules bien formées de la logique du premier ordre en utilisant les prédicats de l'ontologie[31].
- **Les instances (modèles) :** C'est une définition extensionnelle de l'ontologie, ils sont utilisées pour représenter des éléments, par exemple les individus « Ahmed » et « Salah » sont des instances du concept «personne». [32].

4. Différents types d'ontologies

Les chercheurs du web sémantique ont classé les ontologies selon plusieurs dimensions : leur niveau de formalisation, leur type de conceptualisation, leur propos, leur niveau de complexité, et selon le nombre de points de vue des concepteurs, etc. Nous retenons dans la suite les catégories des ontologies les plus utilisées dans le web sémantique :

Ontologies de représentation des connaissances (méta ontologie) Ces ontologies sont utilisées pour formaliser un modèle de représentation des connaissances. Les exemples les plus représentatifs sont la « Frame Ontology», qui définit les primitives de représentation des langages à base de frames (classes, instances, propriétés, facettes, relations, etc.) pour décrire les concepts des autres types d'ontologies[33].

Les ontologies de haut niveau Visent à modéliser les concepts de haute abstraction tels que les entités, les évènements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés, etc. [34], indépendants d'un problème ou d'un domaine d'application particulier.

Les ontologies de domaine Décrivent des connaissances d'un domaine spécifique. Leurs concepts sont souvent définis comme une spécialisation des concepts des ontologies de haut niveau. Les ontologies de domaine sont définies de deux manières :

- Des ontologies spécifiques à un domaine particulier qui modélisent les connaissances spécifiques à ce domaine précis.
- Des ontologies de tâches ou d'application qui concernent des tâches réalisées dans un domaine particulier. Elles décrivent un vocabulaire en relation avec une tâche ou une activité d'un domaine. Généralement, ces ontologies ne sont pas réutilisables et possèdent une portée limitée.

Les ontologies de domaine permettent la représentation des connaissances d'un domaine particulier et leur réutilisation par des applications de ce domaine. C'est le type d'ontologies le plus courant en ingénierie ontologique [35].

Les ontologies d'application : elles contiennent les définitions requises pour modéliser la connaissance dans une application particulière. Elle sont spécifiques à un champ précis d'application dans un domaine donné [36].

Les ontologies génériques appelées également méta-ontologies. Elles décrivent des concepts généraux, indépendants d'un domaine ou d'un problème particulier, mais moins abstraits que

ceux décrits dans les ontologies de haut niveau, tels que les concepts de temps, d'espace, de notion mathématiques [28].

Uschold et Gruninger [37] ont distingué quatre types d'ontologies selon le type de langage utilisé pour les implémenter. Selon eux, il existe : - *Ontologie informelle* est exprimée en langage naturel. - Ontologie *semi-informelle* est exprimée dans une forme restreinte et structurée de la langue naturelle. - *Ontologie semi-formelle* est exprimée dans un langage artificiel définit formellement(exemple : Ontolingua [38], OWL [39]); et - *Ontologie formelle* est exprimée dans un langage artificiel disposant d'une sémantique formelle, permettant de prouver des propriétés de cette ontologie. Selon la définition de [40], une ontologie fortement informelle ne serait pas une ontologie puisqu'elle n'est pas compréhensible par la machine.

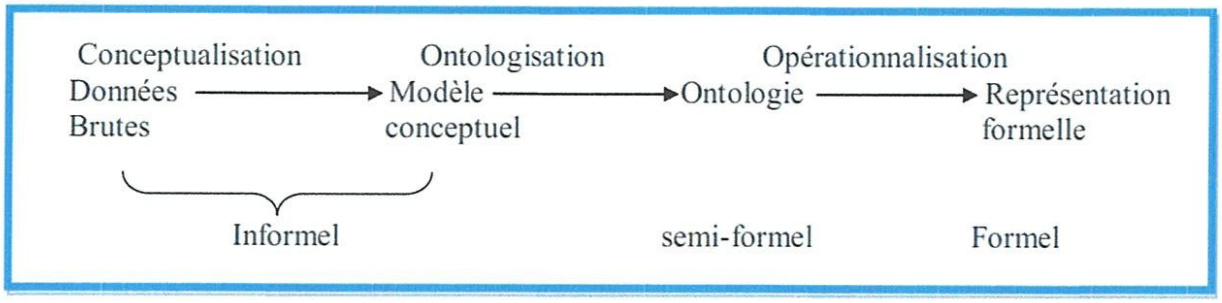


Figure 2.1 : *les types d' ontologies selon* [37]

5. Les principes de construction d'une ontologie

Le concepteur d'ontologie doit respecter certains principes de bases dans le processus de construction d'une ontologie qui permettent d'obtenir une ontologie susceptible répondre aux objectifs visés au départ [23] tel que : - *la clarté,* - *la complétude, la cohérence* de définition de concepts, - *L'extensibilité* par l'ajout d'autre concepts sans avoir à toucher aux fondations de l'ontologie, - *Engagements ontologique minimal* où le but d'une ontologie est de définir un vocabulaire pour décrire un domaine, si possible de manière complète, ni plus, ni moins. - *Principe de distinction ontologique* entre les classes qui devraient être disjointes. Le principe de - *Modularité* qui vise à minimiser les couplages entre les modules. - *Réduire au minimum la distance sémantique* entre les concepts enfants de mêmes parents. Et il est ainsi préférable de - *normaliser* les noms des concepts autant que possible.

6. Langages et Outils pour les ontologies

6.1. Langages de spécifications ontologiques

Plusieurs langages de spécification d'ontologies (ou langage d'ontologies) ont été développés. Dans cette section, nous donnons une liste exhaustive mais n'est pas complète :

- **Extended Markup Language (XML)** fournit une syntaxe pour des documents structurés, mais n'impose aucune contrainte sémantique à la signification des documents[41].
- **Resource Description Framework (RDF)** est un modèle de données pour représenter les objets et les relations entre eux, fournissant une sémantique simple pour ce modèle qui peut être représenté dans une syntaxe XML[42].
- **RDF-Schéma** est un langage de définition de vocabulaire pour la description de propriétés et de classes représentées par des ressources RDF. RDF-S permet de définir des graphes de triplets RDF, avec une sémantique de généralisation/hiérarchisation de ces propriétés et de ces classes[43].
- **La couche d'inférence d'ontologie (OIL)** a été développée dans le projet On-To-Knowledge. Il est basé sur des logiques de description, des langages basés sur des images et des standards Web (par exemple, XML, RDF et RDFS). Il est conçu pour décrire et échanger des ontologies[44].
- **DAML + OIL** est le résultat d'un effort pour combiner DARPA (Defense advanced Research Projects Agency) Agent Markup Language (DAML) et OIL. DAML + OIL est plus efficace qu'OIL en ce qu'il comprend plus de fonctionnalités à partir des logiques de description. Cependant, de nombreuses fonctionnalités basées sur des images ont été supprimées de DAML + OIL, ce qui rend plus difficile l'utilisation de DAML + OIL avec des outils basés sur des images[45].
- **Web Ontology Language (OWL)** est développé comme une extension du vocabulaire de RDF et il est dérivé du langage d'ontologies DAML + OIL[39]. OWL peut être utilisé pour représenter explicitement les sens des termes des vocabulaires et les relations entre ces termes. OWL vise également à rendre les ressources sur le Web aisément accessibles aux processus automatisés, d'une part en les structurant d'une façon compréhensible et standardisée, et d'autre part en leur ajoutant des méta-informations.[23].

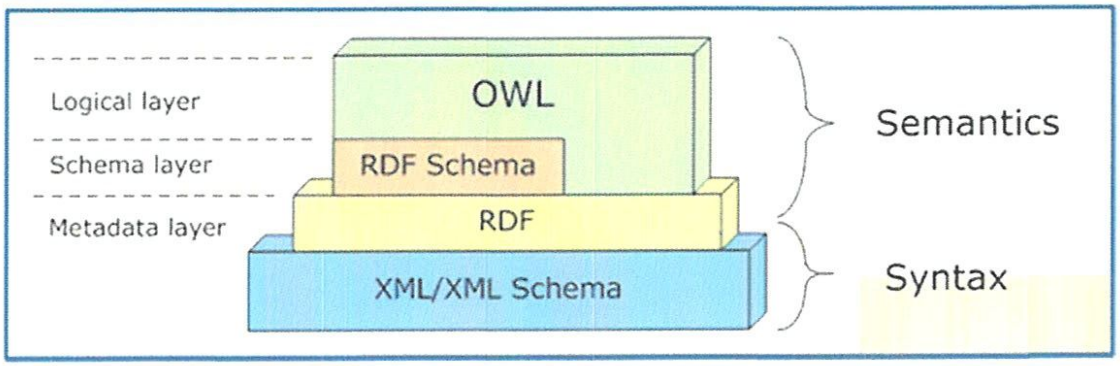


Figure 2.2 : *Architecture de langage d'ontologie*[46].

- ***Les espèces d'OWL :***

Le langage OWL offre trois sous-langages d'expression croissante conçus pour des communautés de développeurs et d'utilisateurs spécifiques :OWL-Lite, OWL-DL et OWL-Full [47].

- **Le langage OWL Lite :** c'est le sous langage le plus simple et le moins expressif. Il permet de représenter une hiérarchie de classes simples et d'exprimer des contraintes simples. Certaines propriétés comme la disjonction et l'union de classes ne sont pas supportées. Il supporte seulement une partie des constructeurs d'OWL.

- **Le langage OWL DL** Il est plus complexe mais aussi plus expressif qu'OWL Lite. Le langage OWL DL supporte tous les constructeurs du langage OWL mais ils sont utilisés avec certaines restrictions. Par exemple, OWL DL ne permet pas la définition de relations transitives. Il se fonde sur la logique de description et supporte le raisonnement automatisé. OWL DL est caractérisé par la complétude de ses raisonnements (toutes les déductions peuvent être calculées) et leur décidabilité (calculs en un temps fini). Mais il n'est pas totalement compatible avec RDF/RDFS.

- **Le langage OWL Full :** C'est la version la plus complexe qui utilise toutes les constructions du langage OWL sans restrictions. C'est aussi le sous-langage qui assure le plus haut niveau d'expressivité. Il existe d'autres langages tels

que : KIF (Knowledge Interchange Format), RIF (Rule Interchange Format) et la F-logique, Gellish, etc.

Entre ces trois sous-langages, il existe une compatibilité ascendante. Autrement dit, toute ontologie OWL Lite valide est également une ontologie OWL DL valide, et toute ontologie OWL DL valide est également une ontologie OWL Full valide. En fonction des besoins de l'utilisateur, un sous langage d'OWL peut être plus approprié qu'un autre.

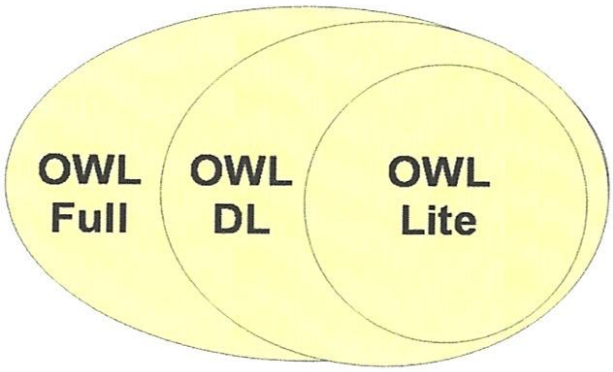


Figure 2.3 : *l'évolution des langages ontologiques*[48].

- ***Les éléments de base du langage OWL :***

OWL se présente sous la forme de classes, de propriétés et d'instances[49].

Une classe comprend une collection de propriétés et décrit un ensemble d'instances. À savoir que toutes les ontologies en OWL ont une "super classe" appelée Thing dont toutes les autres classes sont des sous-classes. Soumises au principe de subsomption, les classes héritent des propriétés des classes qui les dominent.

Une propriété qui permet de définir des faits ou des relations entre ces classes. Il existe en OWL deux types de propriétés : propriété d'objet (owl:ObjectProperty) qui définit une propriété entre deux individus d'une classe ou de plusieurs classes, et une propriété de type de données (owl:DataTypeProperty), c'est à dire une relation entre une valeur ou donnée et un individu d'une classe, l'équivalent d'un champ d'une table dans une base de données relationnelles. Les propriétés peuvent aussi être organisées hiérarchiquement.

Une instance de classe, appelés aussi axiomes sont les membres de ces ensembles. En d'autres termes, ce sont les objets des domaines. Elles sont définies par leur appartenance à une classe et des propriétés. Ces dernières permettent à la fois d'associer les individus

(owl:sameAs permet d'associer Charlie Parker et The Bird) mais également de typer les données auxquelles doivent correspondre les individus (une date d'album correspond à un format particulier de date).

- **Les avantages d'OWL :**

OWL est adéquat pour le web sémantique, il Apporte une meilleure intégration, une évolution, un partage et une inférence plus facile des ontologies grâce à sa sémantique formelle basée sur fondation logique largement étudiée [50] :

- Il ajoute les concepts de classes équivalentes, de propriété équivalente, d'égalité de deux ressources, de leurs différences, du contraire, de symétrie et de cardinalité
- Il permet de définir des associations plus complexes des ressources ainsi que les propriétés de leurs classes respectives
- Il offre une syntaxe définie strictement, une sémantique définie strictement et selon le niveau peut permettre des raisonnements automatisés sur les inférences et conclusions des connaissances

6.2. Outils pour la manipulation des ontologies

Avec l'émergence du marché des technologies du Web Sénanque, on peut noter l'apparition depuis des années des éditeurs d'ontologies gratuits et téléchargeables et d'autres outils logiciels proposés par des éditeurs commerciaux. On peut en citer :

- Protégé est le plus connu et le plus utilisé des éditeurs d'ontologie. Open-source, développé par l'Université de Stanford, a évolué depuis ses premières versions (Protégé-2000) pour intégrer à partir de 2003 les standards du Web sémantique et notamment OWL. Il offre de nombreux composants optionnels : raisonneurs et interfaces graphiques[51].
- SWOOP est un éditeur d'ontologie développé par l'Université du Maryland dans le cadre du projet MINDSWAP. Contrairement à « Protégé », il a été développé de façon native sur les standards RDF et OWL, qu'il prend en charge dans leurs différentes syntaxes (pas seulement XML). C'est une application plus légère que « Protégé », moins évoluée en termes d'interface, mais qui intègre aussi des outils de raisonnement.
- SemanticWorks faisant partie de la suite d'outils XML développée par Altova. Il supporte le langage OWL à travers sa syntaxe XML.

- « TopBraid Composer » est développé par TopQuadrant. Son interface et ses fonctionnalités ressemblent beaucoup à celles de « Protégé » (le développeur principal de TopBraid étant l'ancien développeur des extensions OWL de Protégé).
- Ontology Craft Workbench développé par la société Ontologos-corp suite aux travaux de l'équipe Condillac de l'Université de Savoie. Les ontologies sont disponibles aux formats XML et OWL [46].

7. Raisonnement sur les ontologies

Projection sur votre projet !

Les langages RDF Schema et OWL fournissent un support pour le raisonnement en définissant des hiérarchies de classes ainsi que des relations entre celles-ci et entre des propriétés. Il est par exemple possible de définir une propriété comme étant symétrique par exemple. De telles définitions consistent en des règles d'inférence pouvant être appliquées par des logiciels appelés raisonneurs à la base de connaissance afin d'inférer automatiquement de la connaissance.

- **Vérification de la consistance** - Un raisonneur est capable de s'assurer de la consistance d'une base de connaissance, autrement dit de vérifier qu'il n'y a pas de faits contradictoires dans la base de connaissance.
- **Satisfaisabilité du concept** - Il s'agit de déterminer si une classe peut posséder une instance. Une classe insatisfaisable entraîne une inconsistance si une instance d'une telle classe est définie.
- **Classification** - La classification consiste à établir les liens hiérarchiques entre toutes les classes d'une ontologie, c'est-à-dire identifier les différentes sous-classes de classes. Ceci permet par exemple de fournir toutes les sous-classes d'une classe donnée.

  – **Réalisation** - La réalisation consiste à établir la classe la plus spécifique d'un individu, ce qui nécessite d'avoir réalisé une classification auparavant.

Parmi les raisonneurs existants, citons RacerPro 9, FaCt++ 10, Pellet et HermiT. Ce dernier, écrit dans le langage Java, est un raisonneur OWL.

8. Conclusion

Dans ce chapitre, nous avons présenté le paradigme d'ontologie. Nous avons commencé par présenter la notion d'ontologie, ses composants et son rôle dans le cadre du Web sémantique.

Une partie importante de ce chapitre a été dédiée aux langages de représentation des ontologies tels que le langage OWL, langage que nous avons choisi pour notre contribution dans le cadre de ce mémoire. Ce choix est motivé par les possibilités offertes par ce langage, à savoir un fondement théorique consistant et le support d'une syntaxe utilisable dans le cadre du Web sémantique.

CHAPITRE 3

Travaux existants

1. Introduction

L'idée d'ajouter une sémantique aux processus métiers de l'entreprise n'est pas récente. Dans la littérature, il existe beaucoup de travaux qui ont traité de cette problématique.

Dans ce chapitre, nous présentons certains de ceux que nous avons étudiés.

2. Travaux connexes

2.1. Le travail de Alexander Osterwalder et Yves Pigneur

Dans l'article « **Une ontologie de modèle e-business pour la modélisation de l'e-commerce** » [52], les chercheurs ont présenté une nouvelle ontologie de modèle e_business qui est la base du développement de divers outils utiles pour la gestion des affaires électroniques et IS (Ingénierie des exigences) où ils ont décrit la logique de système d'entreprise et ont souligné les problèmes et les éléments d'affaires électronique pertinents auquel les entreprises doivent penser afin de fonctionner avec succès à l'ère internet. Cette ontologie est composée de quatre piliers principaux qui sont l'innovation de produit, la gestion de l'infrastructure, la relation client et les aspects financières (figure suivante).

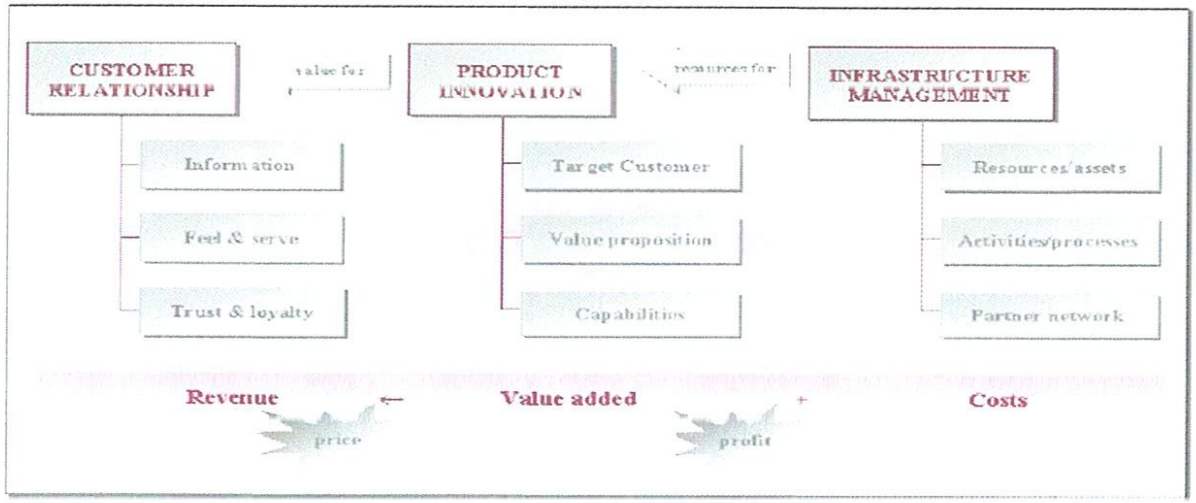


Figure 3.1 : *L' ontologie de modèle e-business*[52]

2.2. Le travail de Marc Ehrig, Agnes Koschmider et Andreas Oberweis

Dans l'article « **Mesurer la similitude entre les modèles de processus métier sémantiques** » [53], les chercheurs ont présenté des solutions pour l'interopérabilité et l'inter

connectivité des processus métier en fonction d'une description OWL-DL basée sur les réseaux de Pétri (figure suivante).

```
<petri:PetriNet rdf:ID="Perform-requests">
   <petri:hasNode rdf:resource="#request"/>
 <petri:Place rdf:ID="request">
  <petri:hasMarking>
   <petri:IndividualDataItem rdf:ID="R_request">
    <petri:hasAttribute rdf:resource="#Name"/>
    <petri:hasAttribute rdf:resource="#Destination"/>
    <petri:hasAttribute rdf:resource="#Date"/>
    <petri:hasAttribute rdf:resource="#Quantity"/>
   </petri:IndividualDataItem>
  </petri:hasMarking>
 </petri:Place>
  ....
</petri:PetriNet>
```

Figure 3.2 : *Représentation de PetriNet en OWL-DL*[53].

Cette solution est une approche pour la détection de synonymes et les homonymes des noms d'éléments de processus afin de supporter l'inter-connectivité et l'interopérabilité du modèle de processus sémantique en mesurant la similarité entre les modèles de processus métier grâce à la traduction des réseaux de Pétri dans OWL. Ils ont démontrés que l'utilisation des trois mesures de similarité : syntaxique, linguistique et structurelle, il est possible de calculer des degrés de similarité entre une paire de noms d'éléments de processus et entre une paire de modèles de processus métier. Afin de calculer le degré de similarité syntaxique, ils ont comparés le nombre de caractères communs dans les noms d'éléments (par exemple, confirmation vs vérification). Le degré de similarité linguistique repose sur un dictionnaire pour déterminer les synonymes. Cependant, la similitude syntaxique et linguistique ne mesure par eux-mêmes le contexte des noms. Ils le font avec des mesures de similarité structurelle, ce qui permet de détecter principalement des homonymes[53][54][54].

2.3. Le travail d'Oliver Thomas, Michael Fellmann

Dans l'article « **EPC[1] sémantique : amélioration de la modélisation des processus en utilisant les langues d'ontologie** » [54], les chercheurs ont décrit une extension sémantique des chaînes de processus axées sur l'événement, avec laquelle il est possible de spécifier la

[1] Event-Driven Process

sémantique des éléments de modèle individuels, car il est indiqué par leur étiquette en langage naturel en utilisant des concepts d'une ontologie formelle. Pour ce faire, ils ont développé une approche multi-niveaux (figure suivante) qui comprend un niveau d'ontologie, un niveau de métadonnées, ainsi qu'un niveau de modèle. Aussi ils ont développé une extension sémantique pour un langage de modélisation de processus, qui représente la sémantique des étiquettes des éléments du modèle de processus avec des concepts d'une ontologie formelle.

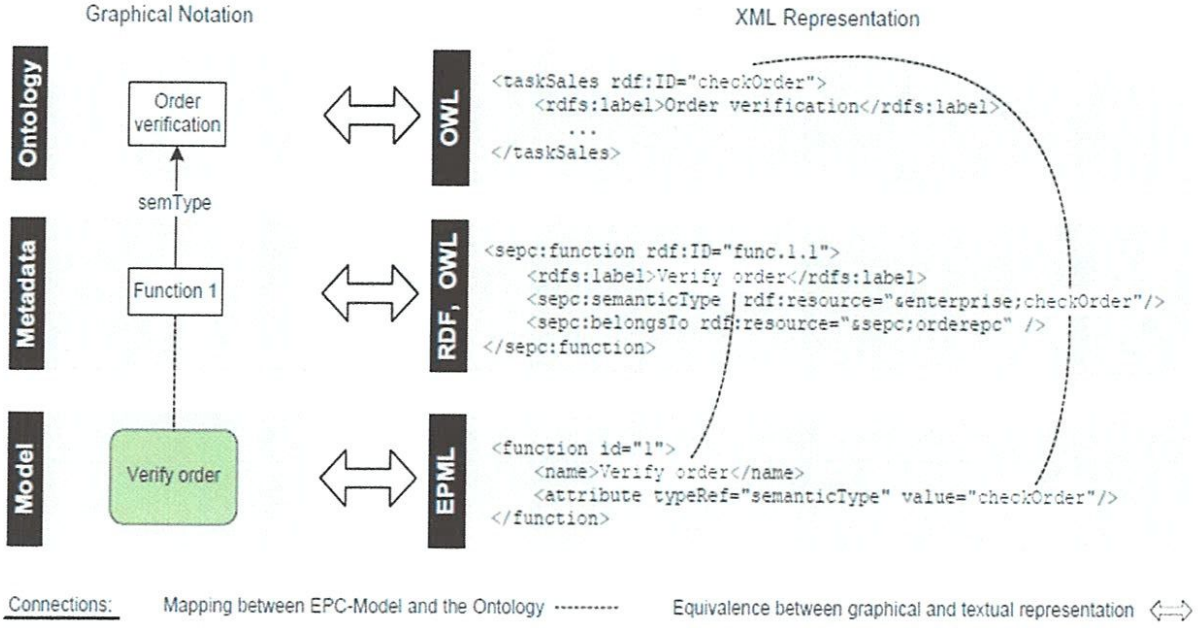


Figure 3.3 : *Relation entre les modèles EPC et les ontologies*[54]

Ils ont sélectionné le langage de modélisation des processus EPC en raison de sa popularité dans la pratique de la modélisation. Cependant, leur approche est principalement transférable à d'autres langages de modélisation semi-formelle, comme par exemple le diagramme d'activité UML[2] ou BPMN[3].

2.4. Le travail de Agata Filipowska, et al.

Dans l'article « **Cadre organisationnel d'ontologie pour la gestion sémantique des processus métiers** » [55], les chercheurs ont décrit un ensemble d'ontologies (figure suivante) pour SBPM (Semantic Business Process Management) qui suivent les exigences suivantes :

- La mise en pratique de la vision SBPM nécessite un réseau cohérent et opérationnel d'ontologies reflétant les différentes sphères des structures et des opérations de l'entreprise.

[2] Unified Modeling Language
[3] Business Process Modeling Notation

- Ontologie Consistante signifie que les ontologies sont basées sur des paradigmes compatibles, ont un degré de détail compatible et comprennent au moins des ensembles partiels de relations d'alignement qui permettent l'interopérabilité des données.
- Opérationnel signifie que les spécifications d'ontologie sont disponibles dans un formalisme d'ontologie unique et actuel pour lequel des référentiels évolutifs, des supports de raisonnement, des API et des outils sont disponibles.

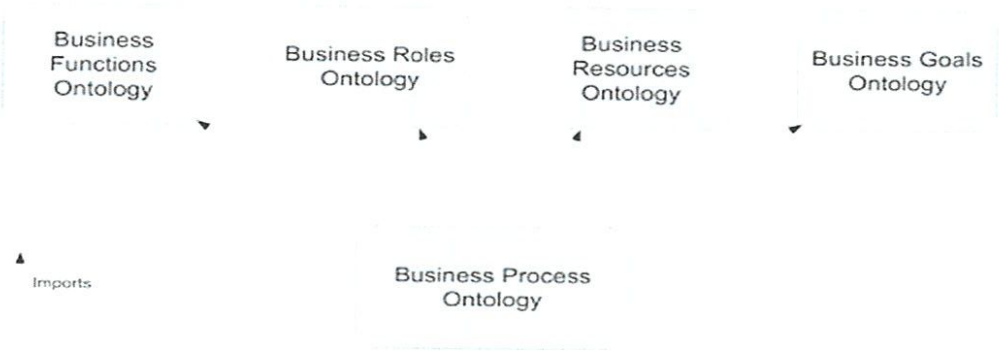


Figure 3.4 : *Cadre organisationnel*[55]

Dans le travail de[56], les chercheurs ont adapté le cadre précédent pour montrer la nécessité d'automatiser la gestion sémantique de la conformité des processus métier. Le défi est de (i) officialiser la prise de décision en matière d'application des politiques, (ii) permettre aux acteurs des entreprises de gérer les connaissances en matière de conformité et (iii) avoir des outils capables d'interpréter ces connaissances. L'application de la conformité trouve sa réalisation potentiellement plus attrayante dans des scénarios hautement collaboratifs où différents partenaires de processus commerciaux interagissent, chacun avec ses propres politiques associées.

2.5. Le travail de Catriel Beeri Anat Eyal, Simon Kamenkovich

Dans ce travail [57], les chercheurs ont présenté BP-QL (Business Process-Qucry Language), un nouveau langage de requête graphique permettant une consultation intuitive des spécifications de processus, en proposant un modèle de données et une interface similaire à celles utilisées pour la spécification BP. Il permet de récupérer des chemins, et offre des facilités pour interroger à différents niveaux de granularité et pour contrôler les requêtes distribuées.

Aussi ils ont présenté un modèle formel pour les systèmes de processus et pour leur langage de requête sur ces systèmes, basé sur des grammaires de graphes. Le modèle permet de

distinguer les fonctions de requêtes qui peuvent être efficacement supportées, et celles qui comportent un coût prohibitif élevé, ou ne peuvent être calculées du tout. À l'aide de ce modèle, ils ont expliqué comment construire une représentation finie et intuitive des réponses (éventuellement infinies) des requêtes dans le polynôme temporel dans la taille des spécifications. La conception de BP-QL a été inspirée par des travaux antérieurs sur les langages de requête visuelle pour XML, tels que XML-GL et XQBE.

2.6. Le travail de Hans-Georg Fill and Patrik Burzynski

Dans l'article « **Integrating Ontology Models and Conceptual Models using a Meta Modeling Approach»** [58], les chercheurs ont présenté trois approches pour intégrer des modèles d'ontologies et des modèles conceptuels de systèmes d'information (figure suivante) :

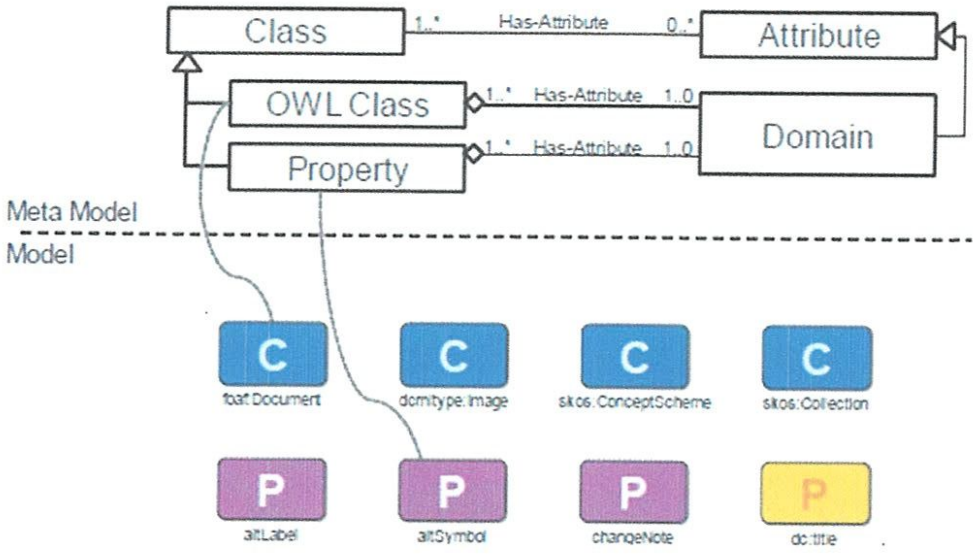


Figure 3.5 : *Modèle et le méta-modèle de l'ontologie*[58].

La première approche intègre le méta-modèle pour un langage de modélisation conceptuel et le langage de modélisation des ontologies OWL. Ainsi, les méta-modèles sont directement liés les uns aux autres. Cela permet de réaliser un environnement de modélisation cohérent pour les modèles conceptuels et les ontologies. Les mécanismes généraux et les algorithmes travaillant sur des méta-modèles arbitraires peuvent être directement réutilisés pour les modèles d'ontologie. Une lacune de cette approche est que l'interaction des utilisateurs avec les modèles visuels de grandes ontologies peut être difficile et que des mécanismes ontologiques spécifiques tels que l'inférence doivent être réintégrés pour les méta-métrages de l'ontologie.

Dans la deuxième approche, les méta-modèles des modèles conceptuels sont étendus avec des attributs aux concepts de référence d'une ontologie OWL. L'ontologie n'est pas tenue dans le même environnement que les modèles conceptuels, mais plutôt dans un référentiel ontologique. Les éléments du modèle conceptuel peuvent donc être annotés en utilisant les identifiants de ressources uniques de l'ontologie.

Avec cette approche, des modifications indépendantes des ontologies et des modèles conceptuels sont possibles et l'utilisation d'un référentiel ontologique permet de fournir des mécanismes spécifiques pour la manipulation des ontologies.

Avec la troisième approche, une combinaison des deux approches précédentes est réalisée. D'une part, les méta-modèles des modèles conceptuels sont intégrés au méta-modèle de l'ontologie. D'autre part, seul un sous-ensemble de l'ontologie est représenté sur le côté des modèles conceptuels. En utilisant des mécanismes d'échange entre les modèles conceptuels et un référentiel d'ontologie, les parties d'une ontologie peuvent être représentées comme des modèles et utilisées pour la liaison avec d'autres éléments de modèle. Dans le même temps, l'ontologie peut être modifiée dans le référentiel otologique sans accorder l'utilisation de concepts ontologiques pour l'annotation de modèles. Comme les parties de l'ontologie utilisées pour la liaison avec les éléments du modèle sont conservées sous forme de modèles, les incohérences possibles entre les modèles d'ontologie et le référentiel d'ontologie peuvent être détectées et résolues si nécessaire.

2.7. Le travail de Khalid Belhajjame, Marco Brambilla

Dans l'article « **Ontology-Based Description and Discovery of Business Processes»** [59], les chercheurs ont exploré une source supplémentaire d'informations codées sous la forme d'annotations qui décrivent sémantiquement les processus métier. Plus précisément, ils ont montré comment les processus métier peuvent être décrits sémantiquement à l'aide des processus d'affaires abstraits par une ontologie et comment cette ontologie peut être construite automatiquement à partir d'une collection de processus d'affaires (concrets), et ils ont illustré comment les experts du domaine peuvent affiner et utilisés dans la découverte des processus métier, dans le but de réutiliser et d'augmenter la productivité de conception. La figure suivante illustre la génération de l'ontologie.

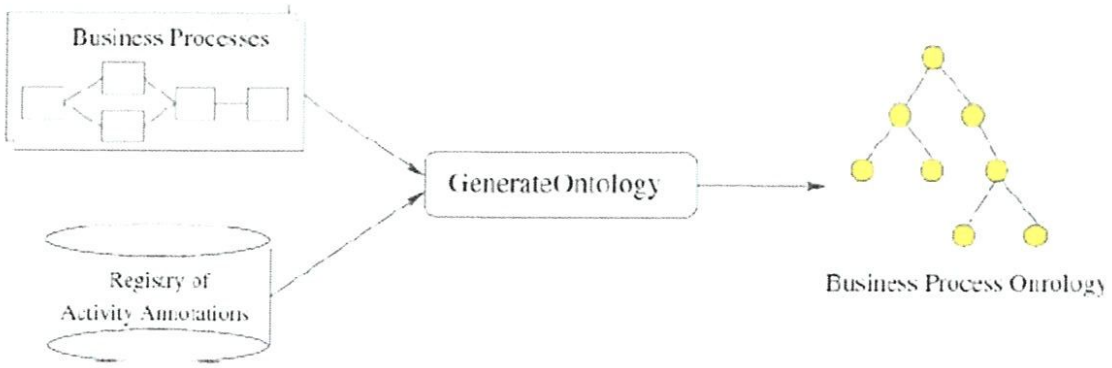


Figure 3.6 : *Génération de l'ontologie*[59].

2.8. Le travail de Martin Hepp, Dumitru Roman

Dans leur travail « **An Ontology Framework for Semantic Business Process Management** » [60], les chercheurs ont décrit les sphères (figure suivante) qui contribuent à l'espace des processus d'une entreprise et qui doivent être représentées dans un cadre SBPM (Semantic Business Process Management) :

- Processus : les chaînes d'activités qui sont réellement exécutées.
- Modèles de processus : les spécifications explicites des processus.
- Organisation : les acteurs qui sont liés par la structure interne de l'entreprise.
- Stratégie d'entreprise : les objectifs stratégiques d'une entreprise influencent également l'espace des processus.
- Contraintes : l'espace de processus est encore influencé par les contraintes liées aux règles légales, réglementaires ou de gestion.
- Fonctions d'affaires : La sphère fonctionnelle d'une entreprise.
- Données transactionnelles et de personnalisation : les données d'entreprise stockées dans les bases de données d'entreprise devraient également être représentées dans un cadre SBPM.

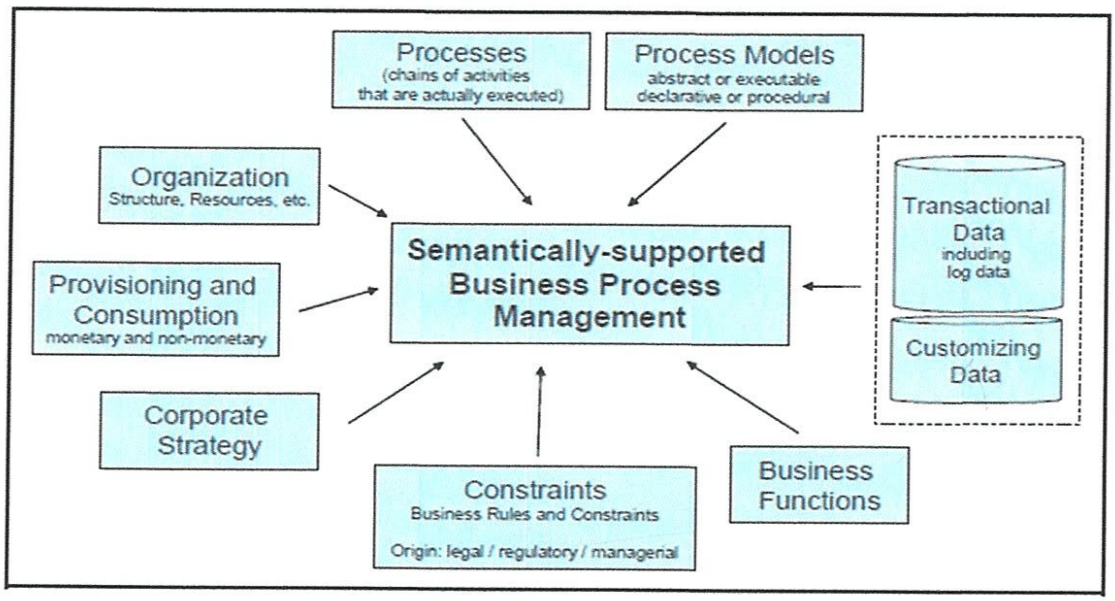


Figure 3.7 :*Les sphères proposés*[60]

Ensuite, les chercheurs ont proposé un ensemble d'ontologies et de formalismes pour ces sphères.

2.9. Les travux de Nitzsche et Al [61] et [62]

Deux travaux très intéressants : le travail de Nitzsche et Al [61] où les auteurs proposent une transformation d'un processus BPEL vers une ontologie décrite en WSML. Et le travail de [62], où les auteurs proposent un mapping entre la spécification BPEL et OWL-S. Les deux travaux sont basés sur la spécification BPEL.

3. Synthèse

Cependant, la plupart des travaux cités, sont classés en deux catégories : la première catégorie des travaux qui sont basés sur la transformation des modèles depuis des processus métier vers des ontologies ou des ensembles d'ontologies selon des exigences proposées par les groupes des chercheurs pour répondre à leurs objectifs, et la deuxième catégories : sont des travaux qui transforment les processus BPEL en des ontologies décrites dans des différents formats (WSML, OWL-S, OWL).

BPEL est lié avec WSDL : les types et les opérations définis en WSDL sont utilisés pour définir les types des variables et les types des messages envoyés et reçus par les activités ‹receive›, ‹reply› et ‹invoke›. De plus, les expéditeurs et destinataires des messages envoyés

sont spécifiés par le biais des liens de communication (‹ partner links ›) qui devront être associés aux services décrits dans la description WSDL [63].

Notre travail dans ce mémoire vient d'étudier le lien entre BPEL et WSDL contenant les déclarations des messages et des variables et de proposer une traduction de ce couple en une ontologie dite BPEL-Ontology.

CHAPITRE 4

Transformation du processus BPEL vers BPEL-Ontology

1. Introduction

Comme nous avons indiqué dans le chapitre précédent, et à notre connaissance, il n'existe aucun travail dans notre contexte qui prend en charge le couple (BPEL, WSDL) dans un processus de transformation vers OWL afin d'ajouter une sémantique aux processus exécutables BPEL.

Le présent chapitre a pour objectif d'exposer les différentes étapes qui seront adoptées pour la transformation d'un processus BPEL et l'interface WSDL en une ontologie. Notre proposition est inspirée des travaux [64], [16], [61] et [65]. Nous basons dans notre travail sur la spécification WS-BPEL 2.0 [66] et la spécification de WSDL 1.1 [67].

Au niveau de ce chapitre, nous avons commencé par la proposition des transformations manuelles pour assurer que nous fournissions une ontologie formelle et valide. Et ensuite, nous proposons un algorithme pour l'instanciation de l'ontologie. La plupart des descriptions des éléments BPEL dans ce chapitre sont retenues de [56].

2. Structure générale de notre démarche

La figure suivante illustre la vue générale de notre démarche. Où nous partons des deux fichiers BPEL et WSDL et nous arrivons à un fichier OWL.

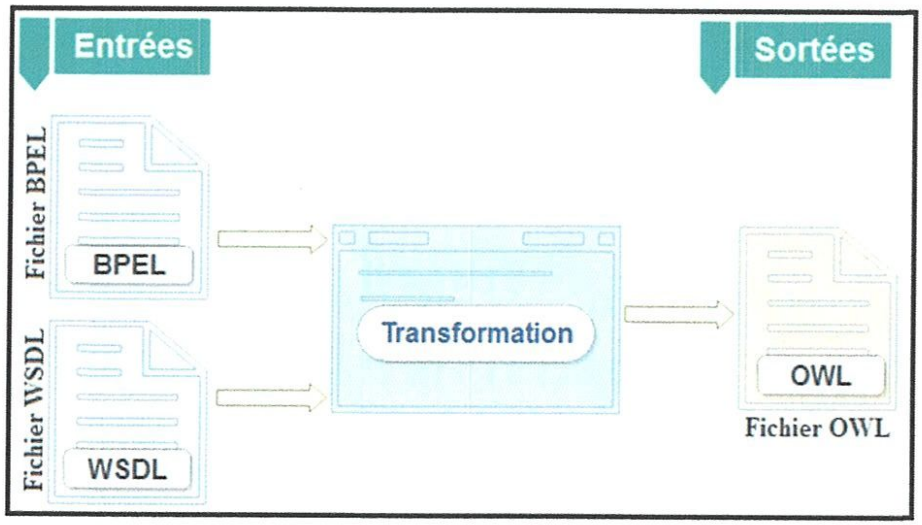


Figure 4.1 : *Vue générale de notre démarche.*

3. Un méta-modèle pour BPEL et WSDL

Un processus BPEL peut interagir avec un ou plusieurs autres processus qui sont présentés dans les partnerLinks. Chaque processus et ses processus partenaires sont modélisés dans la même ontologie.

Pour bien illustrer nos résultats de transformation, nous avons choisi d'appliquer les règles proposées sur des codes réels d'un projet de master[1], le projet a été réalisé pour implémenter l'approche processus dans l'entreprise Amor BenAmor. La mise en œuvre du processus de l'entreprise est basée sur le langage BPEL. Le processus bon de commande joue le rôle du client qui active les processus vente, production, approvisionnement et achat par l'envoi d'une commande. Ce dernier active le processus et puis attend une réponse. Une fois que le processus bon de commande reçoit l'ordre du client, il entre en communication avec les autres partenaires pour accomplir sa tâche et répondre au client (figure suivante).

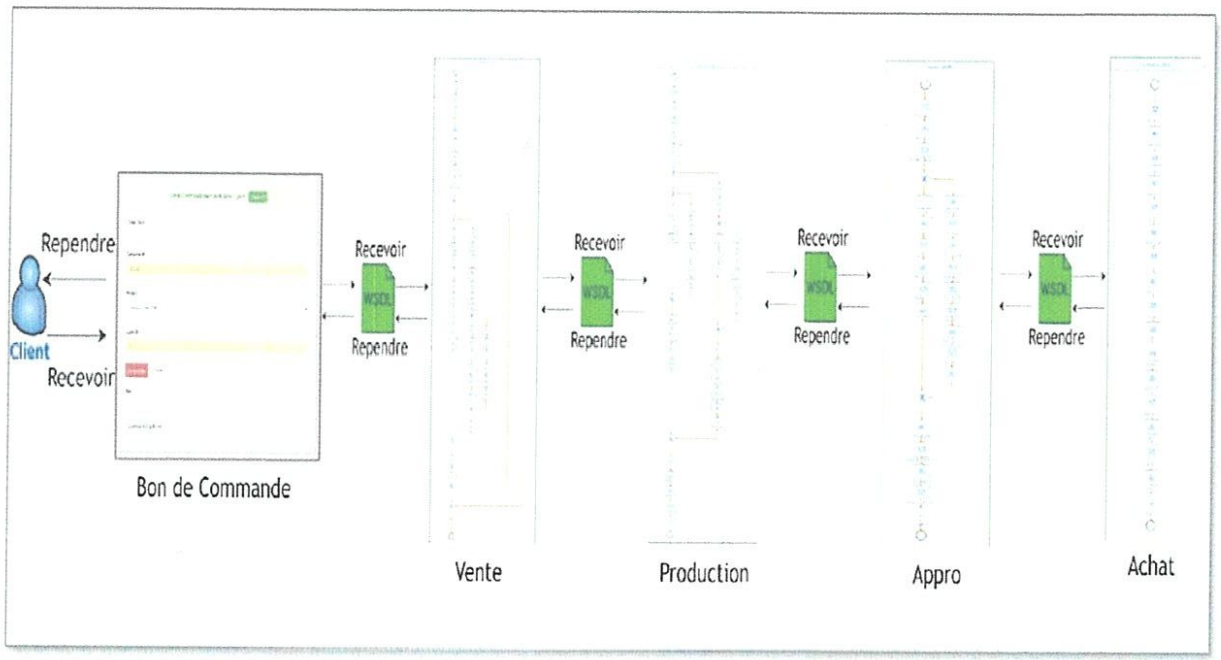


Figure 4.2 : *Scénario d'exécution du processus de vente de Amor BenAmor.*

A partir des structures résumées dans les deux figures suivantes et les définitions vues dans le premier chapitre nous pouvons établir le méta-modèle illustré dans la (figure 4.5) :

1 http://www.mediafire.com/file/p12uwm60pprmeqy/PFE+2016+MAC2+Implementation+de+l%27approche+processus+dans+une+entreprise.rar

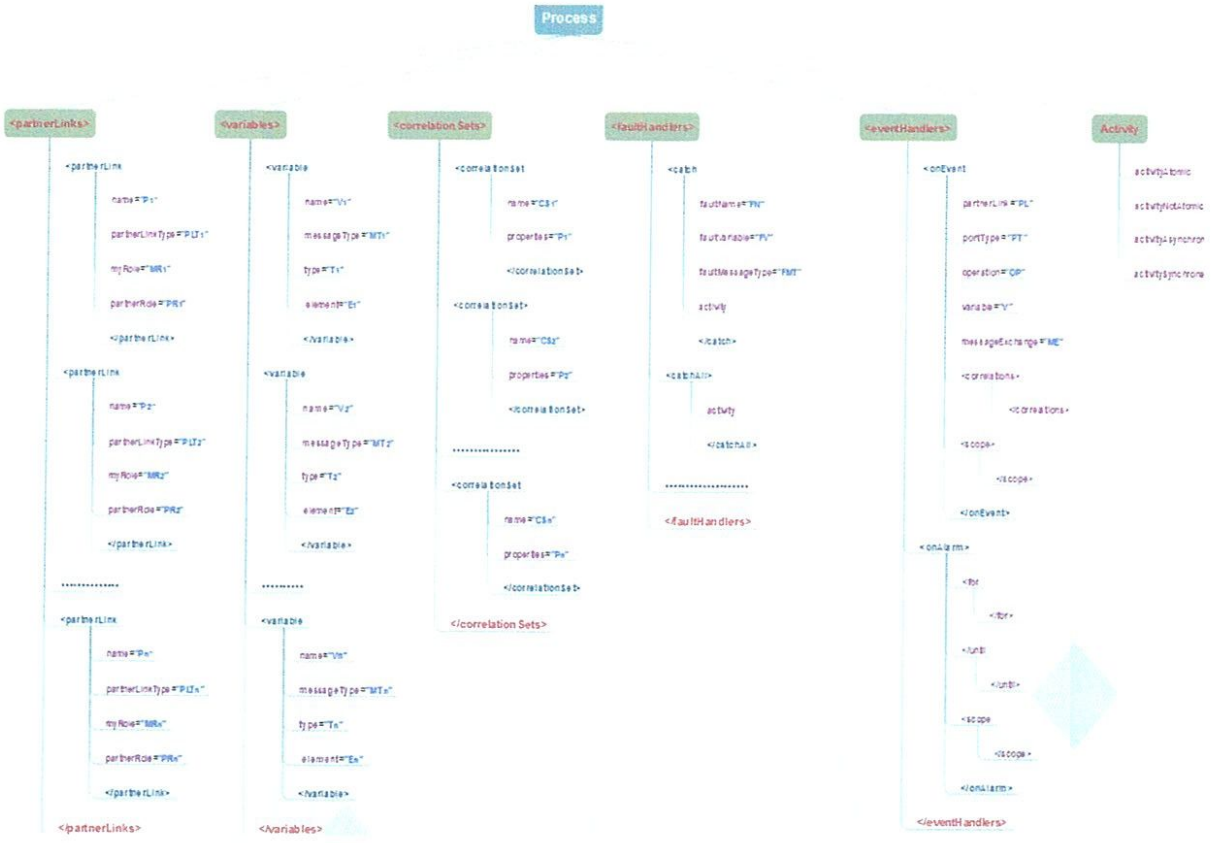


Figure 4.3 : *Structure générale d'un processus BPEL*

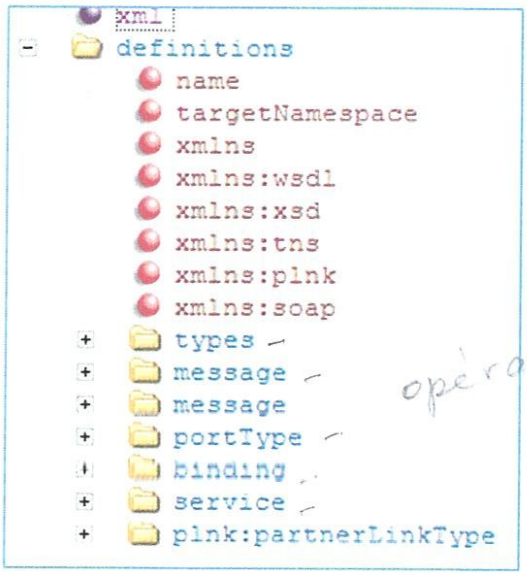
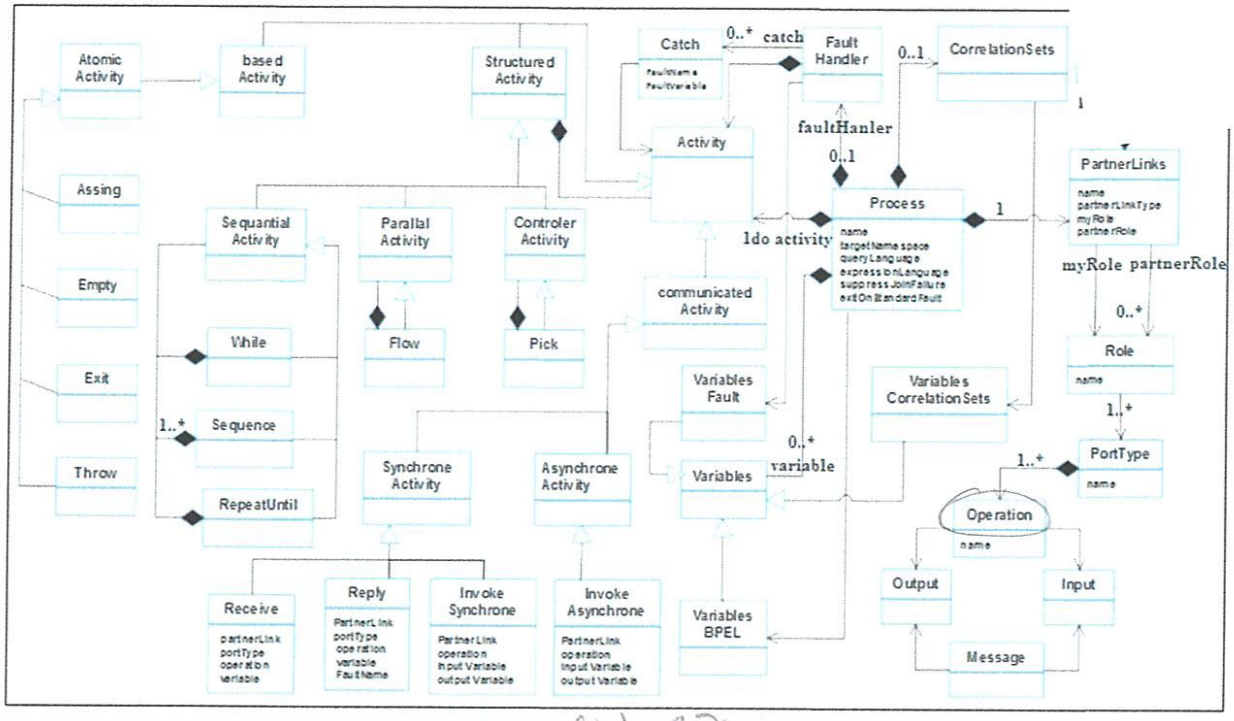


Figure 4.4 : *Structure générale d'un fichier WSDL.*



Slide 37

?? Quel formalisme class UML ?

Figure 4.5 : *Méta-Modèle du BPEL.*

4. Les règles de transformation

4.1. Un processus

Comme règle de départ : le concept processus est représenté comme une classe OWL définit un groupe d'individus possédant des caractéristiques similaires de la façon suivante :

```
<!-- BPEL-Ontologie#Process  -->
 <owl:Class rdf:about="BPEL-Ontologie#Process"/>
```

La classe Process définie dans l'ontologie BPEL-Ontology est une sous-classe de owl :Thing. Un processus est caractérisé par un ensemble d'attributs : name, targetNamespace, queryLanguage, expressionLanguage, suppressionJoinFailure, enableInstanceCompensation et l'attribut abstractProcess dans le cas où le processus est abstrait. Ces attributs sont modélisés comme des propriétés de type de données (dataTypeProperties).

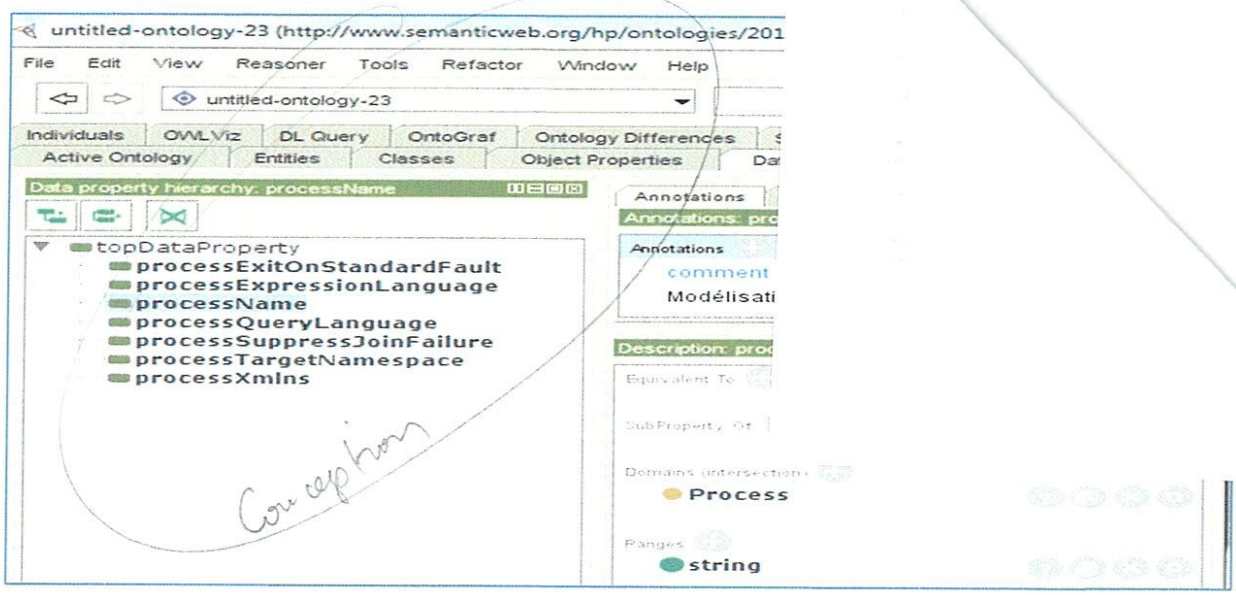


Figure 4.06: *Modélisation des attributs d'un processus en protégé 2000.*

Dans OWL, les propriétés de type de données recouvrent des littéraux RDF ou des types simples définis conformément aux types de données du schéma XML. Les attributs d'un processus sont modélisés par des propriétés de type de données où les type de données sont intégrés du schéma XML plus le type *rdfs :Literal* comme suit :

```
<!-- BPEL-Ontologie#processExitOnStandardFault -->
<owl:DatatypeProperty rdf:about="BPEL-Ontologie#processExitOnStandardFault">
   <rdfs:domain rdf:resource="BPEL-Ontologie#Process"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
 </owl:DatatypeProperty>
<!-- BPEL-Ontologie#processExpressionLanguage -->
<owl:DatatypeProperty rdf:about="BPEL-Ontologie#processExpressionLanguage">
   <rdfs:domain rdf:resource="BPEL-Ontologie#Process"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
 </owl:DatatypeProperty>
<!-- BPEL-Ontologie#processName -->
<owl:DatatypeProperty rdf:about="BPEL-Ontologie#processName">
   <rdfs:domain rdf:resource="BPEL-Ontologie#Process"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
<!-- BPEL-Ontologie#processQueryLanguage -->
<owl:DatatypeProperty rdf:about="BPEL-Ontologie#processQueryLanguage">
   <rdfs:domain rdf:resource="BPEL-Ontologie#Process"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
 </owl:DatatypeProperty>
<!-- BPEL-Ontologie#processSuppressJoinFailure -->
<owl:DatatypeProperty rdf:about="BPEL-Ontologie#processSuppressJoinFailure">
   <rdfs:domain rdf:resource="BPEL-Ontologie#Process"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
 </owl:DatatypeProperty>
```

La structure générale d'un processus en WS-BPEL est donnée par un document XML [66], dont sa racine correspond toujours à l'élément Process (voir la figure suivante) :

```
<process [attributs]>
      <partnerLinks>      ...     </p
      <variables>         ...     </v
      <correlationSets>   ...     </c
      <faultHandlers>     ...     </fa
          <eventHandlers>   ...
      {Activity}
</process>
```

Figure 4.7 : *Structure générale d'un*

Page 13

Un processus est composé des éléments : liens partenaires, variables, ensemble de corrélation, gestionnaires de fautes, gestionnaires d'évènements et ensemble d'activités. Tous ces éléments sont optionnels sauf l'élément ‹Activity›, dans le sens où chaque processus doit avoir une activité principale qui décrit son comportement. Dans notre travail chaque composant est modélisé comme une classe et les liens entre le composé (processus) et les composants sont modélisés par des ObjectProperties (hasVariables, hasPartnerLinks, hasActivity, hasFaultHandlers, hasCorrelationSets, hasEventHandlers).

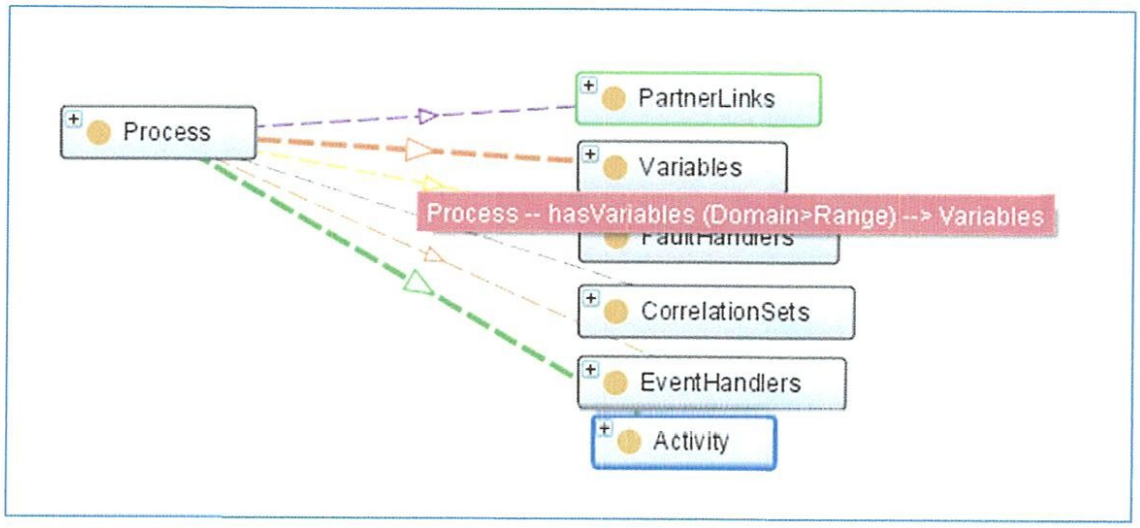


Figure 4.8 : *Modélisation des composants d'un processus.*

4.2. Les partnerLinks

La déclaration des partnerLinks détermine la forme statique des relations que le processus aura avec d'autres processus.

```
<partnerLinks>
    <partnerLink name="VerifClient" partnerLinkType="tns:jdbcpartner" partnerRole="jdbcPortTypeRole"/>
    <partnerLink name="BC" partnerLinkType="tns:javaBonCommandLinkType" myRole="javaBonCommandRole"/>
    <partnerLink name="StockProduit" partnerLinkType="tns:jdbcpartner" partnerRole="jdbcPortTypeRole"/>
    <partnerLink name="Credit" partnerLinkType="tns:jdbcpartner" partnerRole="jdbcPortTypeRole"/>
    <partnerLink name="Production" partnerLinkType="tns:Production_WSDL" partnerRole="Production_WSDLPortTypeRole"/>
    <partnerLink name="BL" partnerLinkType="tns:jdbcpartner" partnerRole="jdbcPortTypeRole"/>
    <partnerLink name="BCStock" partnerLinkType="tns:jdbcpartner" partnerRole="jdbcPortTypeRole"/>
    <partnerLink name="Facture" partnerLinkType="tns:jdbcpartner" partnerRole="jdbcPortTypeRole"/>
  </partnerLinks>
<variables>
```

Chaque partnerLink est caractérisé par un partnerLinkType qui spécifie le lien entre deux services en déterminant les rôles qu'ils jouent, et le porType sur lequel chaque service va recevoir des messages. La définition des partnerLinkTypes est donnée au niveau du fichier WSDL :

```
<plnk:partnerLinkType name="Production_WSDL">
    <plnk:role name="Production_WSDLPortTypeRole" portType="tns:Production_WSDLPortType"/>
  </plnk:partnerLinkType>
```

Nous avons modélisé un *PartnerLinks* comme une classe, qui a comme propriétés de données l'attribut : *namePartnerLink*. Un partnerLink peut avoir un partnerLinkType et un ou plusieurs roles, de ce fait nous proposons l'existence d'autres classes : Role qui a comme attribut nameRole, et PortType, ainsi l'existence d'une relation (ObjectProperty) hasRole entre les classes PartnerLinks et Role.

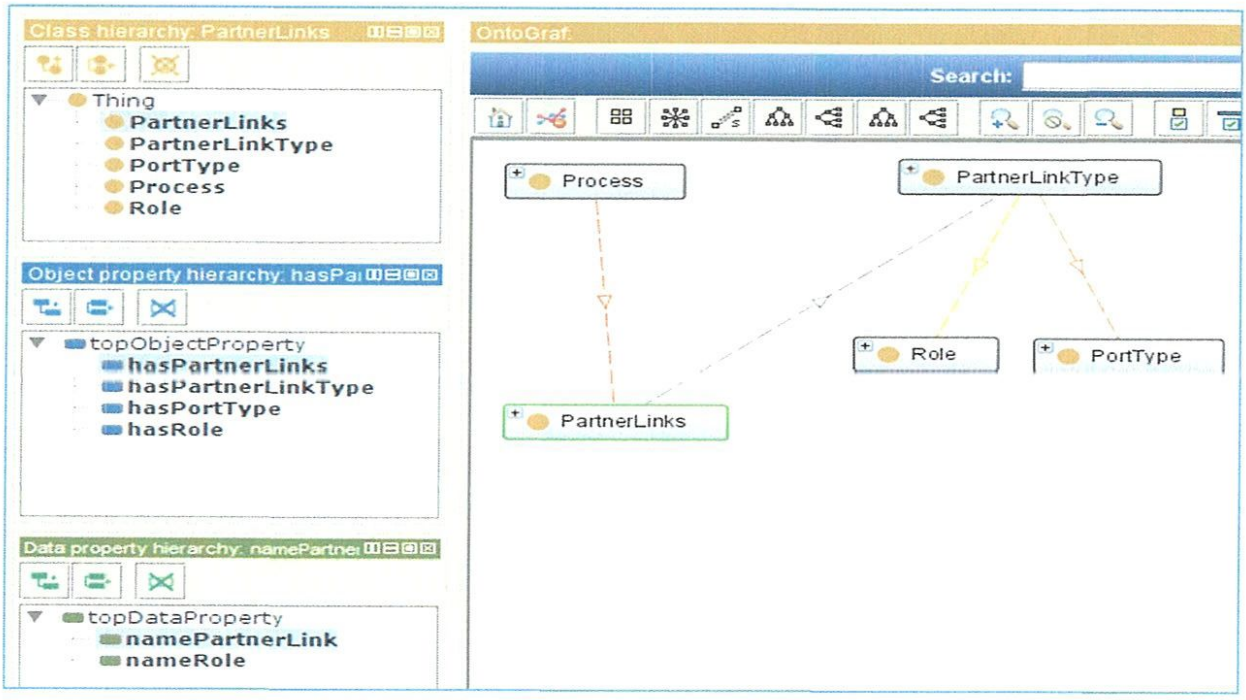


Figure 0.9 : *Représentation de l'élément PartnerLinks et ses relations.*

4.3. Les variables

Dans [65], l'ensemble des variables est divisé en sous-ensembles disjoints : un ensemble de variables définie explicitement par l'élément variables du processus BPEL, un ensemble de variables définie par les éléments partnerLinks et Links du processus BPEL, un ensemble d'horloge, un ensemble de variables de corrélation définies par l'élément correlationSets du processus BPEL, un ensemble fini de variables de fautes, et la variable stopProcess qui est une variable booléenne globale qui initie la terminaison du processus. Nous retenons cette aidée, et nous proposons de décrire la classe Variable comme suit :

```
<!-- BPEL#Variable  -->
<owl:Class rdf:about="BPEL#Variable">
  <owl:disjointUnionOf rdf:parseType="Collection">
     <rdf:Description rdf:about="BPEL#VariablesBPEL"/>
     <rdf:Description rdf:about="BPEL#VariablesCorrelationSetas"/>
     <rdf:Description rdf:about="BPEL#VariablesFault"/>
     <rdf:Description rdf:about="BPEL#VariablesLinks"/>
   </owl:disjointUnionOf>
 </owl:Class>
<!-- BPEL#VariablesBPEL  -->
<owl:Class rdf:about="BPEL#VariablesBPEL">
   <rdfs:subClassOf rdf:resource="BPEL#Variable"/>
 </owl:Class>
<!-- BPEL#VariablesCorrelationSetas  -->
<owl:Class rdf:about="BPEL#VariablesCorrelationSetas">
   <rdfs:subClassOf rdf:resource="BPEL#Variable"/>
 </owl:Class>
<!-- BPEL#VariablesFault  -->
<owl:Class rdf:about="BPEL#VariablesFault">
   <rdfs:subClassOf rdf:resource="BPEL#Variable"/>
 </owl:Class>
<!-- BPEL#VariablesLinks  -->
<owl:Class rdf:about="BPEL#VariablesLinks">
   <rdfs:subClassOf rdf:resource="BPEL#Variable"/>
 </owl:Class>
</rdf:RDF>
```

4.4. Les faultHandlers

Ils permettent de traiter les erreurs qui seraient déclenchées au cours de l'exécution du processus. Ils peuvent être associés à l'élément ‹process›, à une activité ‹scopes› ou à une activité ‹invoke›. L'élément faultHandlers est défini comme classe OWL, et les relations entre cette classe et les classes : ScopeActivity, Invoke activité sont définies comme suit :

```
<!-- BPEL-Ontologie#FaultHandlers  -->
 <owl:Class rdf:about="BPEL-Ontologie#FaultHandlers"/>
```

```
<!-- BPEL-Ontologie#hasFaultHandlers -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#hasFaultHandlers">
   <rdfs:range rdf:resource="BPEL-Ontologie#FaultHandlers"/>
   <rdfs:domain rdf:resource="BPEL-Ontologie#Process"/>
 </owl:ObjectProperty>
<!-- BPEL-Ontologie#invokeFaultHandlers -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#invokeFaultHandlers">
   <rdfs:range rdf:resource="BPEL-Ontologie#FaultHandlers"/>
   <rdfs:domain rdf:resource="BPEL-Ontologie#InvokeActivity"/>
 </owl:ObjectProperty>
<!-- BPEL-Ontologie#scopeFaultHandlers -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#scopeFaultHandlers">
   <rdfs:range rdf:resource="BPEL-Ontologie#FaultHandlers"/>
   <rdfs:domain rdf:resource="BPEL-Ontologie#ScopeActivity"/>
 </owl:ObjectProperty>
```

Les différents gestionnaires de fautes sont représentés comme des sous-classes de la classe FaultHandlers comme suit :

```
<!-- BPEL-Ontologie#Catch -->
<owl:Class rdf:about="BPEL-Ontologie#Catch">
   <rdfs:subClassOf rdf:resource="BPEL-Ontologie#FaultHandlers"/>
   <owl:disjointWith rdf:resource="BPEL-Ontologie#CatchAll"/>
 </owl:Class>
<!-- BPEL-Ontologie#CatchAll -->
<owl:Class rdf:about="BPEL-Ontologie#CatchAll">
   <rdfs:subClassOf rdf:resource="BPEL-Ontologie#FaultHandlers"/>
 </owl:Class>
```

4.5. Les eventHandlers

Ils permettent au processus, ou à une des unités logiques le composant (i.e., un <scope›, de pouvoir traiter les événements *normaux* qui seraient déclenchés au cours de son exécution. Ces événements sont de deux types : ceux qui ont lieu suite à la réception d'un message, et ceux qui se manifestent suite au déclenchement d'une alarme. A partir de cette définition, la relation entre un processus et un gestionnaire d'évènement est représentée comme suit :

```
<!-- BPEL-Ontologie#hasEventHandlers -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#hasEventHandlers">
   <rdfs:range rdf:resource="BPEL-Ontologie#EventHandlers"/>
   <rdfs:domain rdf:resource="BPEL-Ontologie#Process"/>
 </owl:ObjectProperty>
```

4.6. CorrelationSets

Un correlationSet est un groupe de propriétés qui permettent d'identifier une instance unique du processus. Les définitions des CorrelationSets peuvent être trouvées dans les processus d'une manière globale ou dans les scopes du processus d'une manière locale.

Les attributs d'un correlationSet sont : name et properties, modélisés comme des dataTypeProperties.

4.7. Les activités du processus

Les activités d'un processus BPEL sont regroupées sous trois ca.

- ***Les activités de base***

Pour les activités de base, nous distinguons quatre atomiques activités sont qui modélise une activité vide ou une absence d'activité, throw : permet de signale. une faute interne, exit : termine instantanément l'instance d'un processus, assign : est utilisé pour mettre à jour les variables du processus BPEL et et une autre activité c'est l'activité wait : permet de définir une durée d'attente[65], [66]. Les activités atomiques sont représentées comme des sous-classes disjointes.

```
<!-- BPEL-Ontologie#AtomicActivity -->
<owl:Class rdf:about="BPEL-Ontologie#AtomicActivity">
  <rdfs:subClassOf rdf:resource="BPEL-Ontologie#BasedActivity"/>
 <owl:disjointUnionOf rdf:parseType="Collection">
    <rdf:Description rdf:about="BPEL-Ontologie#AssignActivity"/>
    <rdf:Description rdf:about="BPEL-Ontologie#EmptyActivity"/>
    <rdf:Description rdf:about="BPEL-Ontologie#ExitActivity"/>
    <rdf:Description rdf:about="BPEL-Ontologie#ThrowActivity"/>
  </owl:disjointUnionOf>
 </owl:Class>
```

```
<!-- BPEL-Ontologie#NotAtomicActivity -->
<owl:Class rdf:about="BPEL-Ontologie#NotAtomicActivity">
  <rdfs:subClassOf rdf:resource="BPEL-Ontologie#BasedActivity"/>
 </owl:Class>
<!-- BPEL-Ontologie#ThrowActivity -->
<owl:Class rdf:about="BPEL-Ontologie#ThrowActivity">
  <rdfs:subClassOf rdf:resource="BPEL-Ontologie#AtomicActivity"/>
 </owl:Class>
```

- ***Les activités de communication***

Pour les activités de communication [65], nous distinguons des activités synchrone (‹receive› : permet d'attendre une invocation d'un service partenaire et de préparer le processus à l'arrivée d'une réponse, ‹reply› : permet d'envoyer un message de réponse à l'invocation d'un partenaire, et‹ invoke› : permet d'invoquer une opération d'un service partenaire et attend sa réponse) et d'autres activités asynchrones (‹invoke› : permet juste d'invoquer une opération d'un service). Nous proposons de représenter les activités synchrones comme des sous-classes disjointes.

```
<!-- BPEL-Ontologie#InvokeAsynchroneActivity -->
<owl:Class rdf:about="BPEL-Ontologie#InvokeAsynchroneActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontologie#AsynchroneActivity"/>
 </owl:Class>
<!-- BPEL-Ontologie#InvokeSynchroneActivity -->
<owl:Class rdf:about="BPEL-Ontologie#InvokeSynchroneActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontologie#SynchroneActivity"/>
 </owl:Class>
<!-- BPEL-Ontologie#ReceiveActivity -->
<owl:Class rdf:about="BPEL-Ontologie#ReceiveActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontologie#SynchroneActivity"/>
 </owl:Class>
<!-- BPEL-Ontologie#ReplyActivity -->
<owl:Class rdf:about="BPEL-Ontologie#ReplyActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontologie#SynchroneActivity"/>
 </owl:Class>
<!-- BPEL-Ontologie#SynchroneActivity -->
<owl:Class rdf:about="BPEL-Ontologie#SynchroneActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontologie#CommunicateActivity"/>
  <owl:disjointUnionOf rdf:parseType="Collection">
     <rdf:Description rdf:about="BPEL-Ontologie#InvokeSynchroneActivity"/>
     <rdf:Description rdf:about="BPEL-Ontologie#ReceiveActivity"/>
     <rdf:Description rdf:about="BPEL-Ontologie#ReplyActivity"/>
   </owl:disjointUnionOf>
 </owl:Class>
</rdf:RDF>
```

- ***Les activités structurées***

Les activités ‹sequence› : qui permet de définir une collection d'activités pouvant être exécutées séquentiellement, ‹if qui permet de définir une liste ordonnée de choix conditionnels permettant de sélectionner une seule activité à exécuté, ‹while›, ‹repeatUntil› et ‹forEach› qui permettent une exécution répétée de leurs sous-activités, toutes ces activités sont des activités structurées permettent un contrôle séquentiel d'exécution de leurs sous-activités. Deux autres activités structurées : ‹flow› qui permet une exécution parallèle de ses activités ainsi que leur synchronisation. Et enfin l'activité ‹pick› qui permet de définir un choix contrôlé par l'arrivée d'un évènement[65], [66].

```
<!-- BPEL-Ontology#SequantialActivity -->
<owl:Class rdf:about="BPEL-Ontology#SequantialActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontology#StructuredActivity"/>
  <owl:disjointUnionOf rdf:parseType="Collection">
     <rdf:Description rdf:about="BPEL-Ontology#ForEachActivity"/>
     <rdf:Description rdf:about="BPEL-Ontology#RepeatUntilActivity"/>
     <rdf:Description rdf:about="BPEL-Ontology#WhileActivity"/>
   </owl:disjointUnionOf>
 </owl:Class>
```

```
<!-- BPEL-Ontology#Activity -->
 <owl:Class rdf:about="BPEL-Ontology#Activity"/>
<!-- BPEL-Ontology#ControlerActivity -->
<owl:Class rdf:about="BPEL-Ontology#ControlerActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontology#StructuredActivity"/>
 </owl:Class>
<!-- BPEL-Ontology#FlowActivity -->
<owl:Class rdf:about="BPEL-Ontology#FlowActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontology#ParallalActivity"/>
 </owl:Class>
<!-- BPEL-Ontology#ForEachActivity -->
<owl:Class rdf:about="BPEL-Ontology#ForEachActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontology#SequantialActivity"/>
 </owl:Class>
<!-- BPEL-Ontology#ParallalActivity -->
<owl:Class rdf:about="BPEL-Ontology#ParallalActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontology#StructuredActivity"/>
 </owl:Class>
<!-- BPEL-Ontology#PickActivity -->
<owl:Class rdf:about="BPEL-Ontology#PickActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontology#ControlerActivity"/>
 </owl:Class>
<!-- BPEL-Ontology#RepeatUntilActivity -->
<owl:Class rdf:about="BPEL-Ontology#RepeatUntilActivity">
   <rdfs:subClassOf rdf:resource="BPEL-Ontology#SequantialActivity"/>
 </owl:Class>
```

4.8. Message, portType et operation

Une variable BPEL fait toujours référence à un message défini dans une description WSDL du service composé ou de l'un de ses partenaires [65]. Un service Web invoqué par un processus BPEL est modélisé par un ‹portType›, qui est un ensemble d'opérations offertes par le service. Les éléments ‹input› et ‹output› sont représentés comme des classes disjointes associés à une opération par deux ObjectProperties hasInPut et hasOutPut. Un message est définie comme une classe OWL, elle a comme dataTypeProperties (messagePart et messageElement). Les codes suivants représentent les différentes classes et relations proposées :

```
<!-- BPEL-Ontologie#Input -->
<owl:Class rdf:about="BPEL-Ontologie#Input">
   <owl:disjointWith rdf:resource="BPEL-Ontologie#OutPut"/>
 </owl:Class>
<!-- BPEL-Ontologie#Message -->
 <owl:Class rdf:about="BPEL-Ontologie#Message"/>
<!-- BPEL-Ontologie#Operation -->
 <owl:Class rdf:about="BPEL-Ontologie#Operation"/>
<!-- BPEL-Ontologie#OutPut -->
 <owl:Class rdf:about="BPEL-Ontologie#OutPut"/>
<!-- BPEL-Ontologie#PortType -->
 <owl:Class rdf:about="BPEL-Ontologie#PortType"/>
</rdf:RDF>
```

```
<!-- BPEL-Ontologie#hasInPut  -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#hasInPut">
   <rdfs:range rdf:resource="BPEL-Ontologie#Input"/>
   <rdfs:domain rdf:resource="BPEL-Ontologie#Operation"/>
 </owl:ObjectProperty>
<!-- BPEL-Ontologie#hasMessageIn  -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#hasMessageIn">
   <rdfs:domain rdf:resource="BPEL-Ontologie#Input"/>
   <rdfs:range rdf:resource="BPEL-Ontologie#Message"/>
 </owl:ObjectProperty>
<!-- BPEL-Ontologie#hasMessageOut  -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#hasMessageOut">
   <rdfs:range rdf:resource="BPEL-Ontologie#Message"/>
   <rdfs:domain rdf:resource="BPEL-Ontologie#OutPut"/>
 </owl:ObjectProperty>
<!-- BPEL-Ontologie#hasOperation  -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#hasOperation">
   <rdfs:range rdf:resource="BPEL-Ontologie#Operation"/>
   <rdfs:domain rdf:resource="BPEL-Ontologie#PortType"/>
 </owl:ObjectProperty>
<!-- BPEL-Ontologie#hasOutPut  -->
<owl:ObjectProperty rdf:about="BPEL-Ontologie#hasOutPut">
   <rdfs:domain rdf:resource="BPEL-Ontologie#Operation"/>
   <rdfs:range rdf:resource="BPEL-Ontologie#OutPut"/>
 </owl:ObjectProperty>
```

5. Instanciation de BPEL-Ontology

Après la présentation des règles de transformation et la construction de la partie terminologique et les différents axiomes, nous proposons dans cette section un algorithme pour l'instanciation de l'ontologie BPEL-Ontology, l'algorithme proposé a comme entrée les deux fichiers BPEL et WSDL, et comme résultat un fichier OWL.

Algorithme Instanciation-BPEL-Ontology ;

Entrée

Fichier BPEL (P : process, PL :partnerLink , PLT : partnerLinkType, FH :FaultHandler, EH :EventHandler, A :Activity) ;

Fichier WSDL(R :role, PT :portType, MESS :message, OP: opération, InP :InPut, OutP :OutPut);

Sortie

Fichier OWL ;

Début

Créer une instance de classe Process (dataTypeProperty.name="nP")

Pour chaque partnerLink (<partnerLink name="nPL" partnerLinkType="pLT" partnerRole="pLR"/>)

Créer une instance de classe PartnerLinks (dataTypeProperty.name="nPL")

Créer une instance d'objectProperty (hasPartnerLinks(nP, nPL))

Pour chaque partnerLinkType

```
<plnk:partnerLinkType  name="pLT">
      <plnk:role name="nR"  portType="nPT"/>
</plnk:partnerLinkType>
```

Créer une instance de classe partnerLinkType (dataTypeProperty.name="pLT")

Créer une instance de classe Role (dataTypeProperty.name="nR")

Créer une instance de classe PortType (dataTypeProperty.name="nPT")

Créer une instance d'ObjetcProperty (hasRole("pLT","nR"))

Créer une instance d'ObjectProperty (hasPortType("pLT","nPT"))

Pour chaque portType

```
<portType name="nPT">
<operation name="nOP">
<input name="nInP" message="nMESS1"/>
<output name="nOutP" message="nMESS2"/>
</operation>
</portType>
```

Créer une instance de classe operation (dataTypeProperty.name="nOP")

Créer une instance de classe InPut (dataTypeProperty.name="nInP")

Créer une instance de classe OutPut (dataTypeProperty.name="nOutP")

Créer une instance de classe Message (dataTypeProperty.name="nMESS1")

Créer une instance de classe Message (dataTypeProperty.name="nMESS2")

Créer une instance d'ObjetcProperty (hasOperation("nPT","nOP"))

Créer une instance d'ObjetcProperty (hasInPut("nOP","nInP"))

Créer une instance d'ObjetcProperty (hasMessage("nInP","nMESS1"))

Créer une instance d'ObjetcProperty (hasOutPut("nOP","nOutP"))

Créer une instance d'ObjetcProperty (hasMessage("nOutP","nMESS2"))

Pour chaque Variable (<variable name="nV" messageType=" mTV"/>

Créer une instance de classe VariableBPEL
(dataTypeProperty.name="nV", dataTypeProperty.messageType="mTV")) ;

Créer une instance d'ObjectProperty (hasPortType("nP","nV"))

Pour chaque FaultHandler (<FaultHandler name="nFH" />

Créer une instance de classe FaultHandler
(dataTypeProperty.name="nFH") ;

Créer une instance d'ObjectProperty (hasFaultHandler("nP","nFH"))

Pour chaque EventHandler (<EventHandler name="nEH" />

Créer une instance de classe EventHandler
(dataTypeProperty.name="nEH") ;

Créer une instance d'ObjectProperty (hasEventHandler("nP","nEH"))

Pour chaque type d'activité Activity

Créer une instance de classe Type-Activity (dataTypeProperty.name="nA");

Créer une instance d'ObjectProperty (hasActivity("nP","nA"))

Fin.

La figure suivante montre quelques instances de classes, des fichiers présentés dans l'annexe :

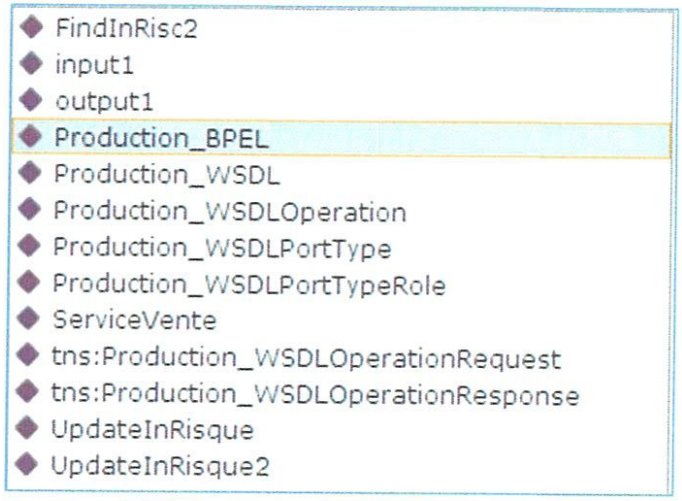


Figure 04.10 : *Exemple d'instanciation de l'ontologie.*

6. Conclusion

L'ontologie BPEL-Ontology proposée définit formellement la plupart des termes employés dans la description des processus BPEL. Au sein de BPEL-Ontology, les constructeurs et les concepts de BPEL et WSDL sont regroupés sous forme de classes sémantiques, ces classes définissent un groupe d'instances possédant des propriétés similaires. Dans la suite, nous allons mettre en œuvre notre application.

l'élément SCOPE ??

1. Introduction

Après avoir terminé notre proposition, maintenant, nous concentrons notre effort sur la mise en œuvre et l'implémentation de notre application. Le prototype logiciel développé est succinctement décrit et quelques aspects pratiques sont montrés.

Dans ce chapitre, nous commençons tout d'abord par présenter l'environnement de travail, les outils utilisées dans cet environnement, l'architecture générale de notre application ainsi que les interfaces principales avec les grandes fonctionnalités pour s'assurer qu'il répond au but de notre thème.

2. Présentation de l'environnement et les outils de développement

Pour le développement de notre application, on a utilisé des outils qui seront présentés dans cette section ainsi que les raisons qui ont motivé leurs choix.

2.1. NetBeanIDE 8.0.2

Nous avons basé dans notre implémentation sur la plate-forme de développement NetBeans 8.0.2, qui est un environnement de développement intégré IDE (Integrated Development Environment) gratuit, open source. Il permet de développer des applications en plusieurs langages, comme Python, C, C++, JavaScript, XML, le Groovy[1], PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi – langage, refactoring, éditeur graphique d'interfaces et de pages web)[68]. NetBeans utilise le principe de plug-ins, ce qui permet d'ajouter les fonctionnalités selon les besoins (l'extensibilité).

En ce qui concerne JEE7[2], NetBeans fournit un environnement riche et complet pour concevoir des applications (visuelles ou non) et les déployer au sein de serveurs tels que GlassFish Server ou Tomcat qui font partie intégrante du kit d'installation. NetBeans est disponible sous différents bundles[3]. Chacun couvrant un ensemble de fonctionnalités [69]. La figure suivante montre l'interface principale de Netbeans :

[1] *Groovy : est le nom d'un langage de programmation orienté objet destiné à la plate-forme Java.*
[2] *JEE : Java Enterprise Edition.*
[3] *Bundles : Paquets par exemple le pack SOA.*

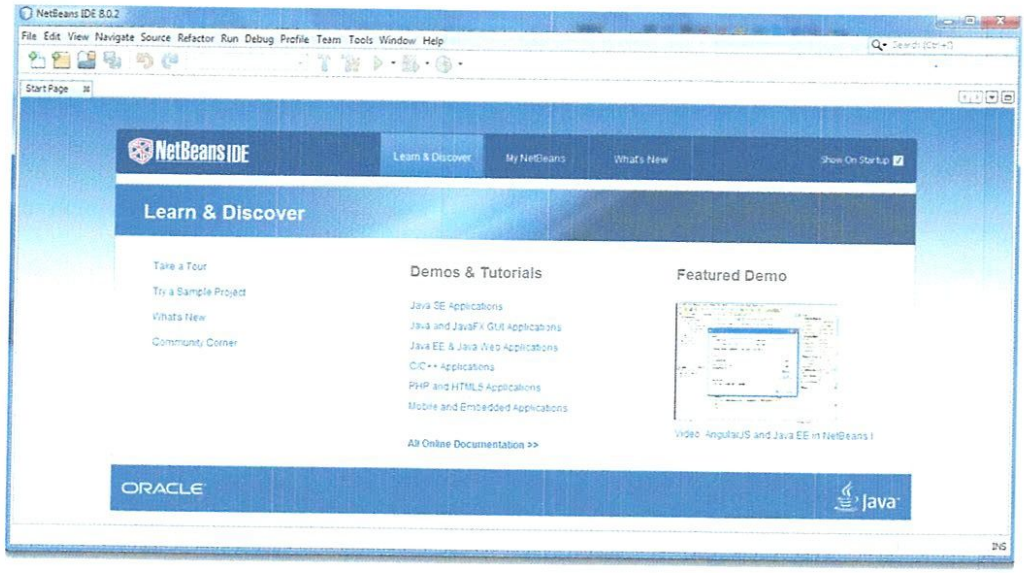


Figure 5.1 : *Interface Netbeans IDE.*

2.2. Le langage JAVA

Le langage Java, née en 1995, est un langage de programmation orienté objet créé par Sun Microsystems, il permet d'écrire de façon simple et claire des programmes portables sur plusieurs systèmes d'exploitation tels que Windows, Mac OS ou Linux. C'est la plateforme qui garantit la portabilité des applications développées en Java[70]. De plus, il bénéficie d'une très grande bibliothèque de classes avec lesquelles l'utilisateur pourra composer des interfaces graphique, créer des applications multithreads, animer une page HTML par des applets ou encore communiquer en réseau. De plus Java assure une totale indépendance des applications vis-à-vis de l'environnement d'exécution : c'est-à-dire que toute machine supportant Java est en mesure d'exécuter un programme sans aucune adaptation (ni recompilation, ni paramétrage de variables d'environnement). Il est à signaler que malgré la grandeur de la bibliothèque native de Java, certaines fonctionnalités, tel que la manipulation des documents XML, l'accès à une base de données et bien d'autres, nécessitent des APIs[4] qui ne sont pas intégrés dans Java par défaut, on les appelle les APIs externes. Donc il faut les intégrer et cela se fait à travers l'IDE NetBeans.

2.3. L'API JDOM

JDOM[5] est une API permettant la manipulation (création, modification, suppression….) de document XML via une structure arborescente. Il présente quelque similitude avec DOM (Document Object Model). Cependant, il s'en distingue parce que

[4] APIs : une interface de programmation applicative.
[5] (www.jdom.org)

JDOM est spécifiquement conçu pour Java et que du point de vue du développeur Java, il s'avère beaucoup plus pratique à utiliser [71].

Les raisons qui nous ont poussé à utiliser l'API JDOM sont [72]:

- ✓ Sa simplicité et la possibilité de sa manipulation avec java.
- ✓ Elle se veut rapide et légère
- ✓ Elle propose d'apporter une solution à ces différents problèmes dans une seule et même API.
- ✓ Elle encapsule un document XML entier dans un arbre d'objets, chaque élément est encapsulé dans une classe dédiée selon son type.
- ✓ Elle veut masquer la complexité de certains aspects de XML tout en respectant ses spécifications.
- ✓ Elle se veut intuitive et productive notamment grâce à des classes dédiées à chaque élément instancié par son constructeur et l'utilisation de getter/setter

2.4. Éditeur Protégé 5.2.0 :

Protégé 5.2.0 est un éditeur pour la création d'ontologies. Est le plus connu et le plus utilise des éditeurs d'ontologie. Il a été développé par le Stanford Medical Informatics de l'Université de Stanford. Protégé est une plate-forme Open Source autonome, qui fournit un environnement graphique permettant l'édition, la visualisation et le contrôle (vérification des contraintes) d'ontologies. De plus, il peut lire et sauvegarder des ontologies dans la plupart des formats d'ontologies : RDF, RDFS, OWL, etc[73].

L'interface très complète ainsi que l'architecture logicielle extensible permettant l'insertion de plusieurs plug-ins offrant de nouvelles fonctionnalités, notamment des plug-ins pour gérer les représentations sous forme graphique, par exemple OWLViz et la prise en charge de nouveaux langages.

Plusieurs raisons ont motivé notre choix d'outil Protégé :

- ✓ Protégé est un éditeur d'ontologies open source et gratuit.
- ✓ Il possède une interface modulaire, ce qui permet son enrichissement par des modules additionnels (plugins).
- ✓ Il permet l'édition et la visualisation graphique d'ontologies.
- ✓ Il permet de contrôler la cohérence de l'ontologie par la vérification des contraintes.

- ✓ Protégé est fourni une API écrite en JAVA, qui permet de développer des applications pouvant accéder aux ontologies de Protégé et de les manipuler.
- ✓ Il permet d'importer et d'exporter des ontologies dans les différents langages de spécification d'ontologies (RDF-Schéma, OWL, DAML, OIL,...etc.).
- ✓ Exécuter des raisonneurs.

Dans notre application, nous l'utilisons pour la visualisation des ontologies BPELOntology.

3. Architecture et description du système

Dans cette partie nous avons présenté l'architecture de notre système et on va expliquer ses fonctionnalités.

1.1. Architecture du système

Notre application transforme un fichier BPEL vers un fichier OWL, puis le résultatsera visualiser avec Protégé pour la vérification de son cohérence et sa consistance. L'architecture de notre application illustrée dans la figure ci-dessous montre les différentes étapes et les interactions entre les divers composants :

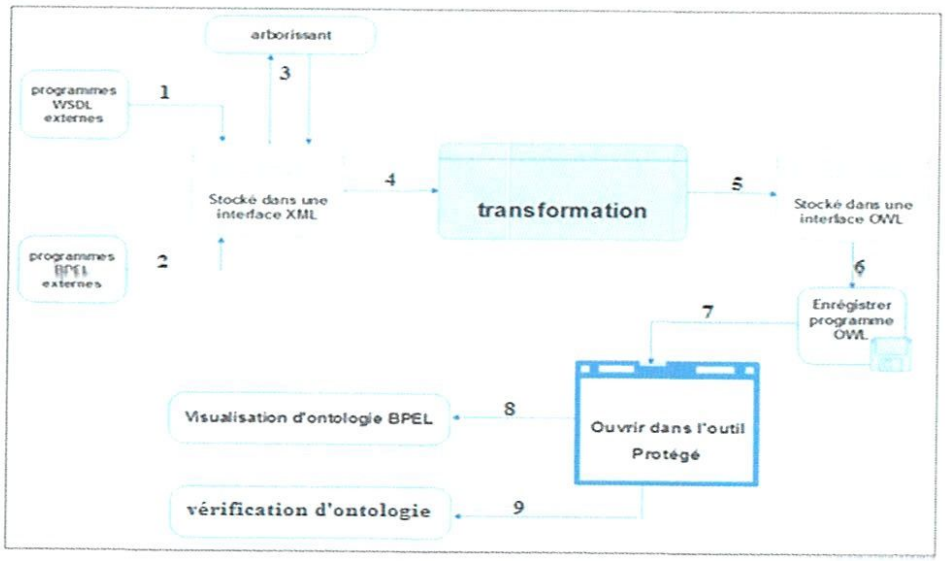


Figure 5.2 : Architecture de l'application BPEL Ontologie.

1. Importer le fichier BPEL.
2. Importer les fichiers WSDL.
3. Analyser de la syntaxe du code BPEL, WSDL pour construire l'arborescente de tous éléments et sous-élément du code puis les stocker dans une liste.
4. La partie essentielle de cette architecture est le module de transformation qui transforme le code BPEL et l'interface WSDL en un code OWL.

5. Le résultat de transformation est un fichier OWL.
6. enregistrement du code résultat avec l'extension .OWL.
7. dans les étapes 7, 8 et 9 importation du fichier OWL avec Protégé pour la visualisation et la vérification.

1.2. Description de notre travail

Dans cette partie nous allons présenter notre implémentation, En générale notre application permet de réaliser les taches suivantes :

- ✓ Le code suivant permet de récupérer les données d'un fichier BPEL (et/ou WSDL) pour les traiter :

```
// Ouverture du fichier BPEL/WSDL
        File XMLFile = new File(pathDuFichierXML) ;
        SAXBuilder builder = new SAXBuilder(true) ;
        Document doc = builder.build(XMLFile) ;
        Element root = doc.getRootElement() ;
```

- ✓ Affichage des codes BPEL et WSDL.
- ✓ Récupération de tous les éléments BPEL et les transformer en un arbre JDOM à l'aide d'une fonction Walker:

```
public static  void recursiveTraverse(TreeWalker walker, String
indent) {
Node parend = walker.getCurrentNode();
String S=txt22.getText().concat(indent + "  -" + ((Element)
parend).getTagName())+"\n";
txt22.setText(S);
int i=0;
for (Node n = walker.firstChild(); n != null; n = walker.nextSibling()) {
i++;
}
recursiveTraverse(walker, indent + ' '+i); }
```

- ✓ Création d'une liste contenant tous les nœuds de l'élément racine Process de notre arborescence,

- ✓ Création pour chaque élément de la liste (élément BPEL source), l'élément correspondant dans le code OWL cible.
  - Un élément process se transforme en une Class : owl.
  - Les éléments composants un processus se transforment en des classes OWL
  - Une relation entre deux éléments composants se transforme en ObjectProperty.
  - Un élément Attribut se transforme en un élément dataTypeProperty.
  - Une valeur de chaque attribut se transforme en une instance (individuals).
- ✓ Affichage du code OWL.

1.3. Espaces de nommage d'ontologie :

Afin de pouvoir employer des termes dans une ontologie, il est nécessaire d'indiquer avec précision de quels vocabulaires ces termes proviennent. C'est la raison pour laquelle, comme tout autre document XML, une ontologie commence par une déclaration d'espace de nom « de nommage » contenue dans une balise rdf: RDF. Le code suivant donne la déclaration d'espace de nom :

```
<rdf:RDF
  xmlns     ="http:// www.w3.org/2002/07/owl#"
  xml:base ="http://www.w3.org/2002/07/owl"
  xmlns:ont ="http://www.co-ode.org/ontologies/ont.owl#"
  xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl ="http://www.w3.org/2002/07/owl#"
  xmlns:xml ="http://www.w3.org/XML/1998/namespace"
  xmlns:xsd ="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#">
```

Toutes les références URI de RDFS sont regroupées dans l'espace de noms :

http://www.w3.org/2000/01/rdf-schema# auquel le préfixe rdfs: est conventionnellement associé.

2. Scénario de transformation dc BPEL vers OWL

Dans le cadre de notre mémoire, cette section serra devisé en deux phases, pour cela on va commencer par la phase de l'implémentation de transformation ainsi que l'organisation des interfaces, par la suite nous décrivons la phase de visualisation d'ontologie.

2.1. 1ère Phase : l'implémentation de transformation :

- **L'interface générale de l'application :**

L'interface générale de l'application est divisée en quatre champs BPEL, WSDL, Arbborescente, OWL.

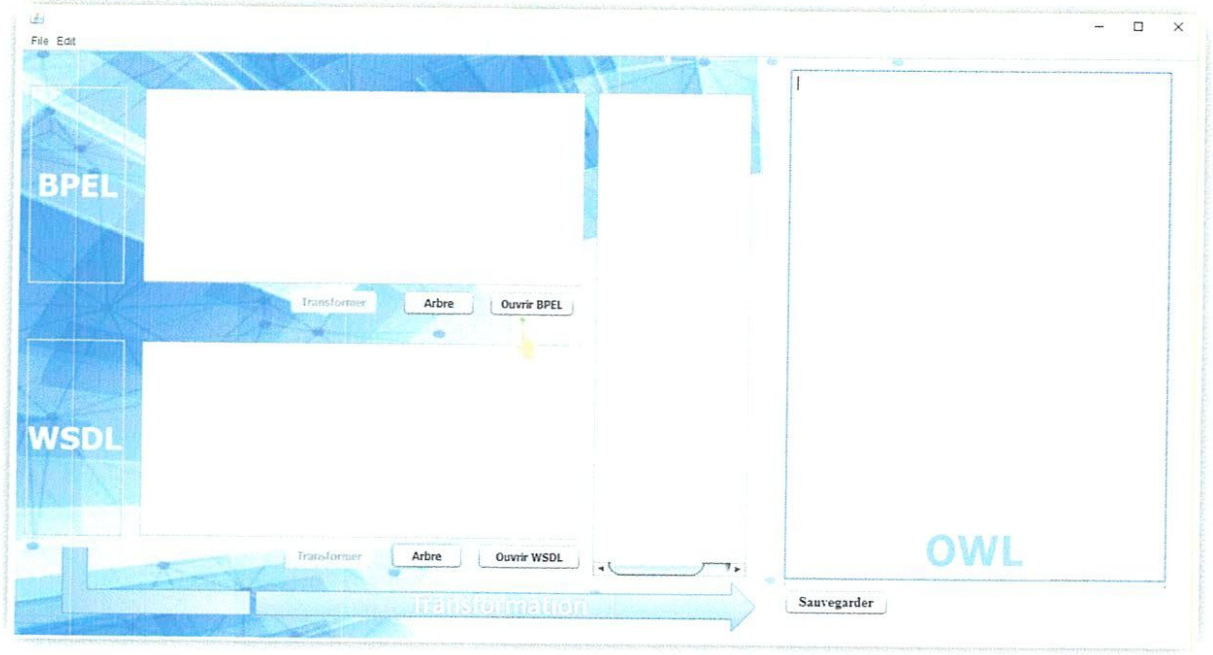


Figure 5.3 : *L'interface générale de l'application.*

- **L'importation du code BPEL :**

Avec notre application BPEL-Ontology nous pouvons sélectionner un fichier BPEL (seule l'extension .BPEL) et l'ouvrir en cliquant sur le Botton ouvrir.

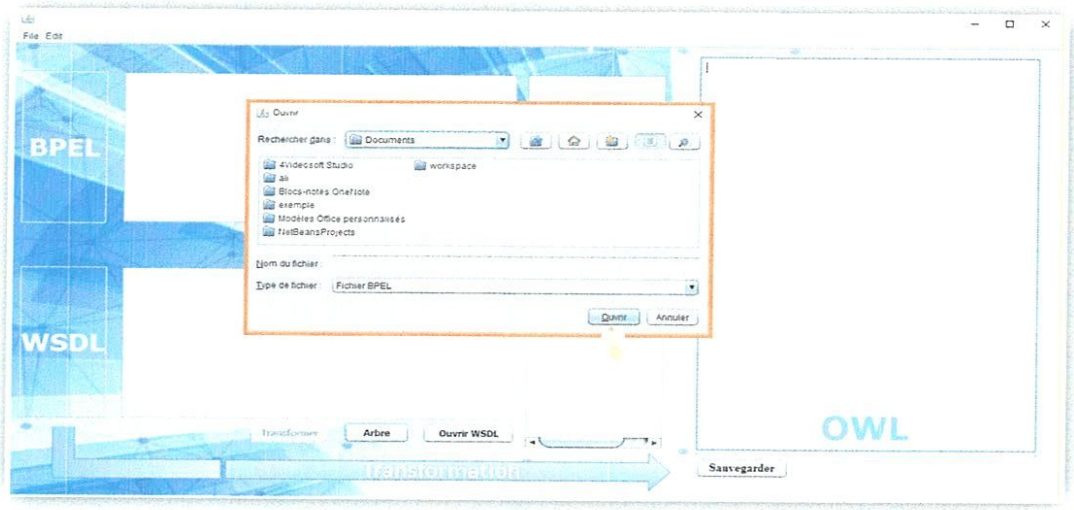


Figure 5.4: *Importation du code BPEL.*

- **La visualisation du code BPEL** : Le champ dans L'interface que nous permet de visualiser le code BPEL.

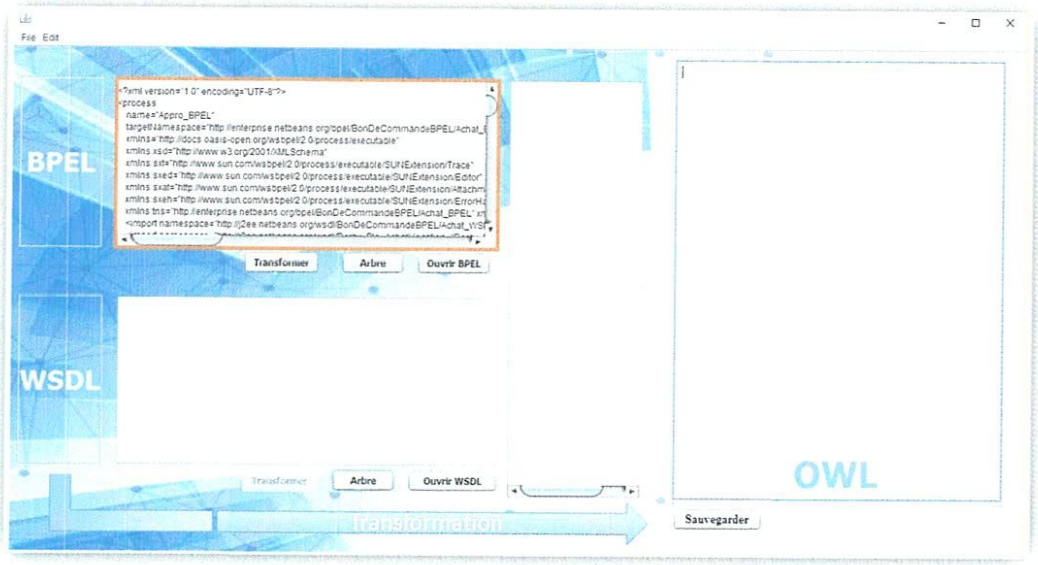


Figure 5.5 : Le champ pour visualisar la code BPEL.

- **L'extraction d'arborescente du code BPEL :**

Construire la structure arborescente qui récupère tous les éléments et les sous-éléments du programme BPEL et les afficher dans le champ spécifique en cliquant sur le Botton arbre.

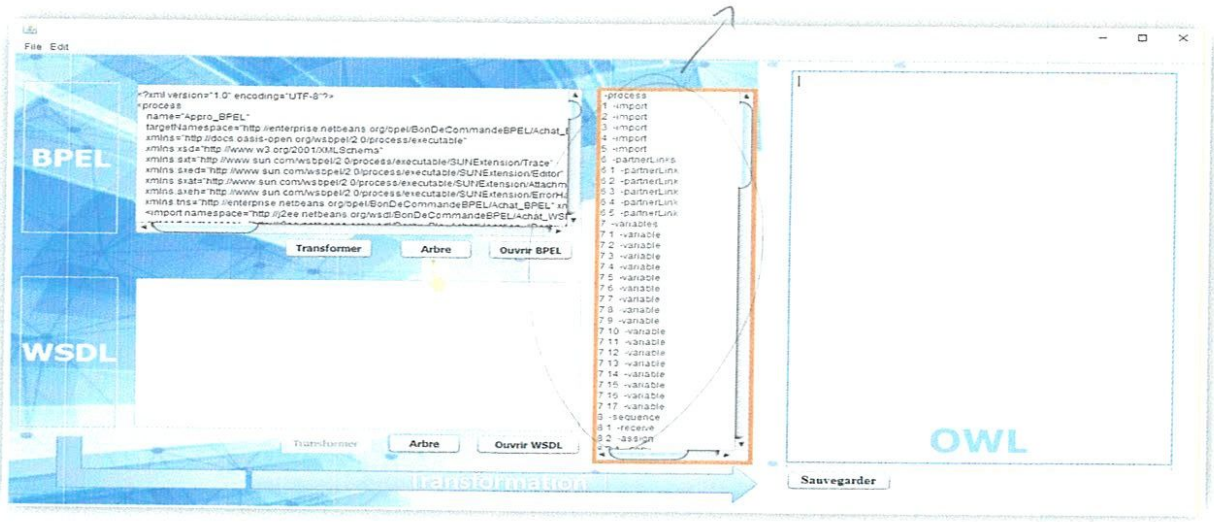


Figure 5.6 : *Le champ pour visualiser l'arborescent du code BPEL.*

- **L'importation du code WSDL :**

Avec notre application BPEL-Ontologie nous pouvons sélectionner un fichier WSDL à partir d'une boite de dialogue et l'ouvrir (seule l'extension .WSDL) en cliquant sur le Botton ouvrir.

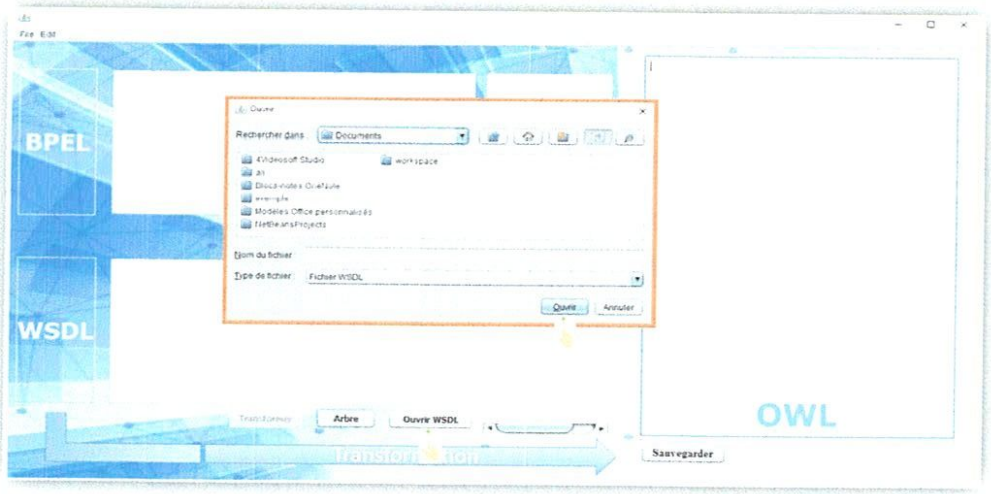


Figure 5.7 : *importation du code WSDL.*

- **Transformation du code BPEL vers OWL** : en cliquant sur le Botton transformer une analyse du programme BPEL est faite, et l'exécution sera commencée pour récupérer les éléments existants pour les transformer en OWL, et afficher le résultat de transformation dans le champ OWL (figure suivante).

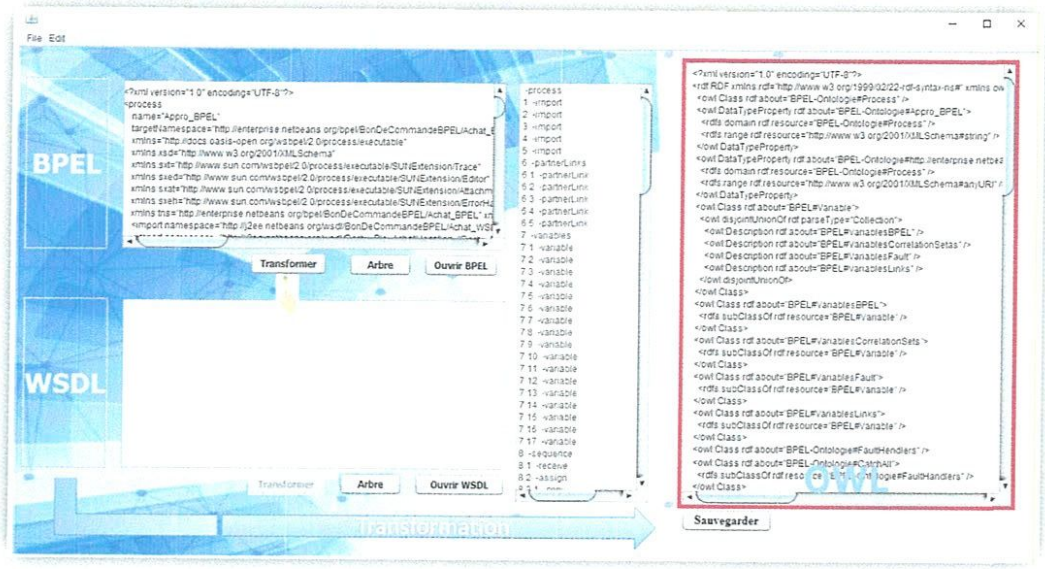


Figure 5.8 : *Le champ pour visualisée du code OWL*

- **Sauvegarder le programme OWL :**

Après la transformation nous pouvons enregistrer le code OWL avec une extension .owl pour l'utiliser dans la deuxième phase de visualisation d'ontologie.

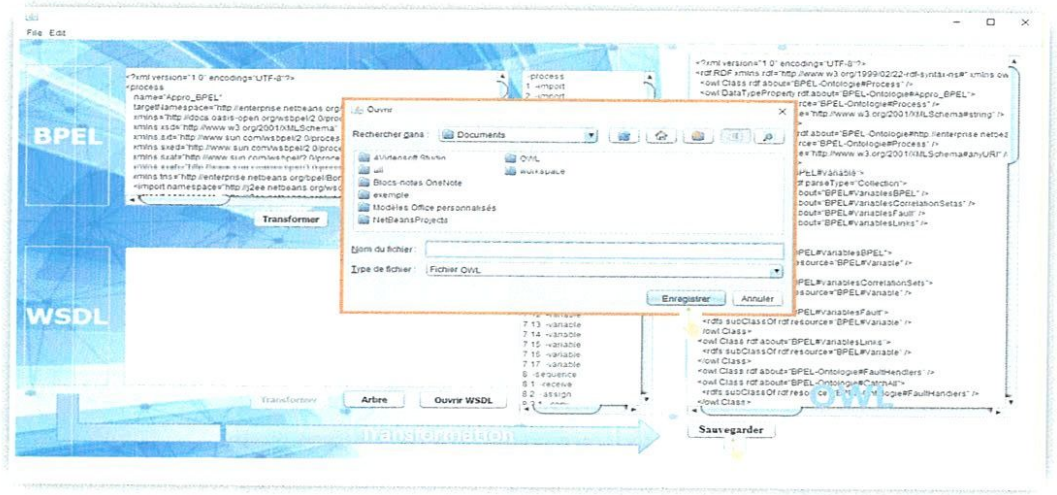


Figure 5.9 : *enregistrement du code OWL.*

2.2. 2ème Phase : Visualisation d'ontologie :

Pour vérifier notre BPEL-Ontology résultat, nous utilisons l'outil protégé, avec ce dernier, nous pouvons parcourir notre ontologie en utilisant les onglets suivants :

- ✓ L'onglet « Classes » nous permet de visualiser les classes.

- ✓ L'onglet «Object Properties » nous permet de visualiser toutes les propriétés.
- ✓ L'onglet «Data Properties » nous permet de visualiser toutes les attributs.
- ✓ L'onglet « Individuals by class » nous permet de visualiser les instances des classes

- **Ouverture de notre ontologie sous Protégé**

Pour ouvrir notre ontologie avec Protégé, il nous faut aller dans le menu système File/Open, open qui nous permet d'ouvrir l'ontologie à partir d'un fichier céer pour le stockage de nos fochier OWL.

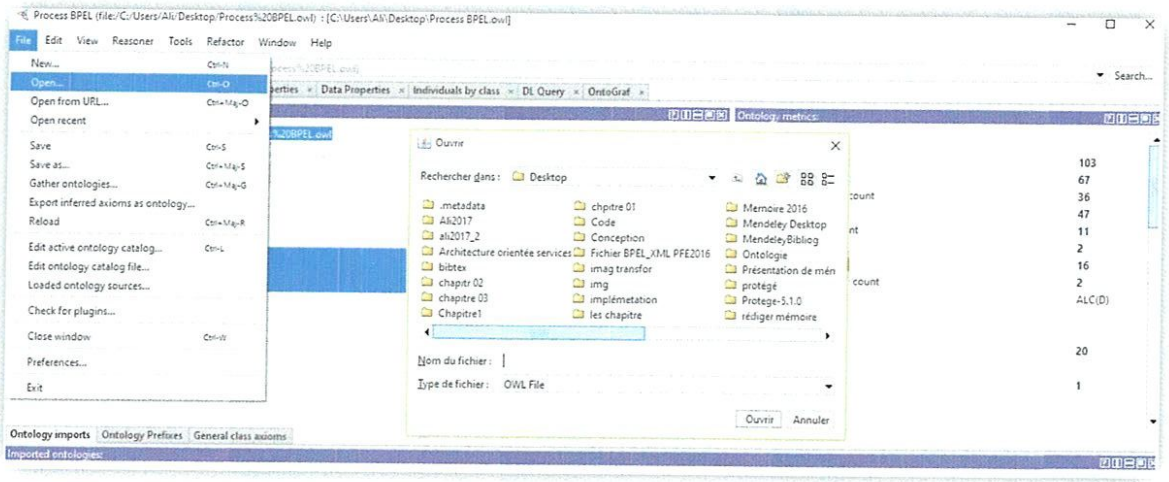


Figure 5.10 : *L'importation d'une ontologie.*

- **La visualisation du graphe de l'ontologie :**

Nous pouvons également afficher notre ontologie en mode graphique, grâce à l'onglet «OntoGraf» qui est un plaging dans Protégé et qui nous permet de visualiser notre BPEL-Ontologie sous forme d'un graphe.

On sélectionne d'abord le concept qu'on souhaite visualiser (par exemple : Process) en tant que racine de la hiérarchie de l'ontologie. Ensuite, soit on utilise la barre « outils » pour choisir un mode de présentation soit on utilise le pop-up menu en cliquant le bouton droit de la souris.

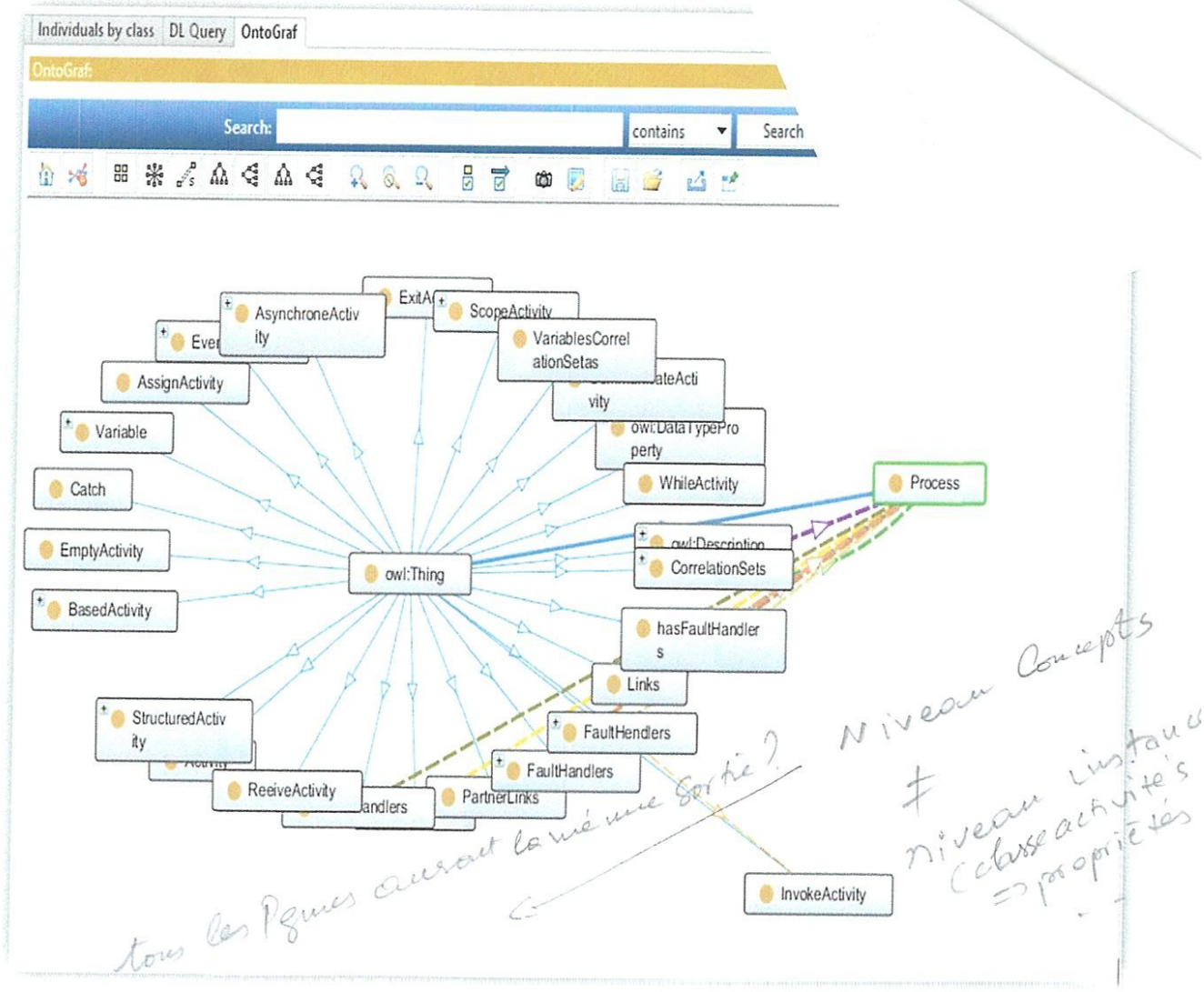


Figure 5.11 : *La visualisation du graphe de l'ontologie.*

3. Évaluation de BPEL-ontologie

Un des grands avantages de l'utilisation de Protégé est la possibilité de faire des raisonnements sur les données présentes dans l'ontologie que l'on a créée grâce aux moteurs d'inférence afin de vérifier la cohérence de l'ontologie[6]et de déduire de nouvelles connaissances ou encore pour faire de la classification.

Dans le menu Reasoner, on va sélectionner HermiT, s'il n'y a pas deja coché dans l'item de menu ; on va sélectionner ensuite et choisis Start Reasoner (figure suivante) :

[6] C'est à dire que l'ontologie créée ne contient pas des définitions contradictoires.

Figure 5.12 : *Sélection du raisonneur.*

- **Test de classification (cohérence) :**

A l'aide du raisonneur HermiT nous pouvons également tester la cohérence de notre ontologie, et nous pouvons vérifier si une classe est une sous classe d'une autre classe ou non parce que la classe inférée est une hiérarchie où les classes sont classifiées selon la relation superclasse/sous-classe. En choisissant l'item de menu «Class hierarchy », « inferred» devrait apparaître les classes de l'ontologie inconsistantes (ne peuvent pas être classées correctement) est affichée en rouge par le système (apparaîtront dans une nouvelle classe Nothing), comme montre la figure suivante :

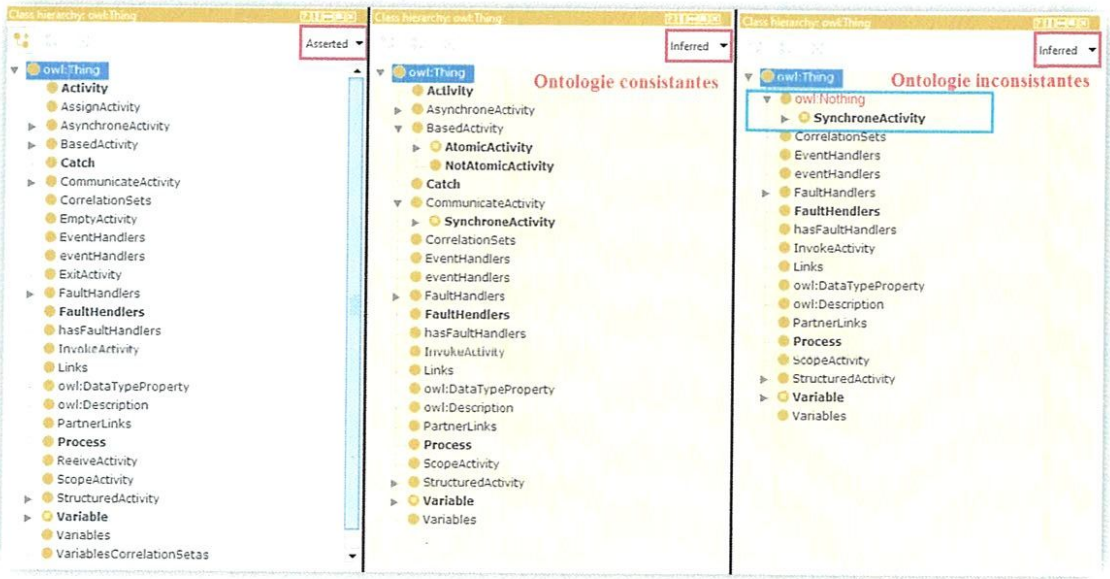


Figure 5.13 : *Vérification de la classification Consistante et inconsistante.*

4. Conclusion :

Dans ce chapitre, nous avons présenté la mise en œuvre de notre application en basant sur un ensemble d'outils pour l'implémentation et ainsi la vérification du résultat.

Le résultat de transformation est un fichier OWL, contient la description formelles des éléments BPEL et WSDL

Conclusion Générale

Conclusion Générale

Notre travail a pour objectif d'apporter une sémantique aux processus métier en termes d'ontologie OWL. Après avoir précisé la spécification BPEL et son interface WSDL, nous avons proposé un ensemble de règles pour construire notre BPEL-Ontology.

BPEL-Ontology est un document OWL qui se compose d'un espace de nommage, d'une entête (owl :Ontology), de la définition de classes, de la définition des propriétés et enfin d'assertion de faits. Les classes définissent des groupes d'individus mis ensemble parce qu'ils partagent certaines propriétés comme : Process, Variables, PartnerLink,... Ces classes sont toutes des enfants de la super-classe OWL :Thing. La définition des instances de classes consiste à énoncer les axiomes des individus. Dans l'ontologie BPEL-Ontology, la plupart des faits concernent la déclaration de l'appartenance aux classes : Process, Variable, PartnerLink,... Les relations entre ces classes sont définies par des propriétés. Les relations entre les individus d'une classe ou de plusieurs classes sont définies par des propriétés d'objets, et les relations entre un individu d'une classe et une valeur ou donnée sont définies par dataTypeProperties.

BPEL-Ontology est une descriptionformelle qui éliminera les ambiguïtés éventuellement existantes et permettra l'utilisation des capacités de raisonnement d'une ontologie basée sur un langage logique pour vérifier la cohérence des processus d'entreprise et de les compléter avec des contraintes d'intégrité et des règles de déduction.

L'une des perspective les plus intéressantes de ce mémoire est d'améliorer le travail engagé et de compléter latransformation de tous les éléments BPEL, et de construire une ontologie de représentation lourde, c'est-à-dire une ontologie intégrant toute la sémantique du domaine, à travers des axiomes et dont le but est d'utiliser cette ontologie pour faire des raisonnements automatiques sur l'ontologie résultat.

Références

[1] S. Chaari, "Interconnexion des processus interentreprises: une approche orient{é}e services." 2008.

[2] D. COULIBALY, "UN LANGAGE ET UN ENVIRONNEMENT DE CONCEPTION ET DE DEVELOPPEMENT DE SERVICES WEB COMPLEXES," université paris dauphine.

[3] A.-S. Feyaerts, "Raisonnement sur les ontologies spatiales," Université libre de Bruxelles, 2010.

[4] A. M. Yang LI, "OWL□: Web Ontology Language," 2006.

[5] P. S. Matjaz Juric, Benny Mathew, *Business Process Execution Language for Web Services 2nd Edition*. 2006.

[6] T. Andrews *et al.*, "BPEL4WS, Business Process Execution Language for Web Services Version 1.1." 2003.

[7] D. F. Leymann, "Web Services Flow Language (WSFL 1.0)."

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

[9] P. Wohed, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede, "Analysis of Web Services Composition Languages The Case of BPEL4WS.," in *ER*, 2003, vol. 2813, pp. 200–215.

[10] OASIS (2007), "Web Services Business Process Execution Language Version 2.0." [Online]. Available: http://bpel.xml.org/.

[11] T. Assia, "Concepts et Outils pour l'Intégration et l'Interopérabilité des Services. Application dans le cadre du E-Government," UNIVERSITE CONSTANTINE 2, 2014.

[12] P. Briol, *Ingenierie Des Processus Metiers, de L'Elaboration A L'Exploitation*. 2008.

[13] W. Fdhila, "D{é}centralisation Optimis{é}e et Synchronisation des Proc{é}d{é}s M{é}tiers Inter-Organisationnels. (Optimized Decentralization and synchronization of Inter-Organizaion Business Processes)," Henri Poincar{é} University, Nancy, France, 2011.

[14] M. Lallali, "Modelisation & Functional Testing of Web Services Orchestration," Institut National des T{é}l{é}communications, 2009.

[15] MOLAY EL□:MEHDI ALAOUI SELSOULI, "TEST UNITAIRE DE PROCESSUS BPEL□: GÉNÉRATION ORIENTÉE CHEMINS DE CAS DE TEST," UNIVERSITÉ DU QUÉBEC À MONTRÉAL.

[16] A. CHAMI, "Vérification de Processus BPEL à L'aide dc Promela-Spin," Université du Québec à Montréal, 2008.

[17] F. ABOUZAID, "Analyse Formelle D'orchestration De Services Web," Université De

Montréal, 2010.

[18] P. Mayer, "Design and Implementation of a Framework for Testing BPEL Compositions," Universität Hannover, 2006.

[19] I. Singh, S. Brydon, G. Murray, V. Ramachandran, T. Violleau, and B. Stearns, *Designing Web Services with the J2EE 1.4 Platform: JAX-RPC, XML Services, and Clients*. Pearson Education, 2004.

[20] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language." 2007.

[21] A. Bakhta, "Méthode De Recherche De Services Web Basé Sur L'Analyse Formelle De Concepts," Université d'oran.

[22] M. et H. B. et Kahoul, "Conception d'un système auteur pour la création et la manipulation d'une base de ressources pédagogiques," Université Badji- Mokhtar Annaba, 2008.

[23] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.

[24] R. Mizoguchi, M. Ikeda, K. Seta, and J. Vanwelkenhuysen, "Ontology for Modeling the World from Problem Solving Perspectives," in *Proc. of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995, pp. 1–12.

[25] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," 2001.

[26] M. R. Yoshinobu Kitamura, "Ontology-based systematization of functional knowledge," 2004.

[27] B. L. Mbaiossoum, "Conception physique des bases de donn{é}es {à} base ontologique□: le cas des vues mat{é}rialis{é}es. (Physicaly Design of Ontology-Based Databases)," {É}cole nationale sup{é}rieure de m{é}canique et d'a{é}rotechnique, Chasseneuil-du-Poitou, Poitiers, France, 2014.

[28] H. H. Mihoubi, *Une approche d{é}clarative de traduction d'ontologies*. Universit{é} Stendhal Grenoble 3, 2000.

[29] A. Budanitsky, "Lexical semantic relatedness and its application in natural language processing," 1999.

[30] D. Bahloul, "Une approche hybride de gestion des connaissances basée sur les ontologies□: application aux incidents informatiques," Institut National des Sciences Appliquées de Lyon, 2006.

[31] S. Staab and A. Mädche, "Axioms are Objects, too - Ontology Engineering beyond the Modeling of Concepts and Relations," in *Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI 2000, Berlin, Germany*, 2000.

[32] BOUARAB née DAHMANI Farida, "Modélisation basée ontologies pour l'apprentissage interactif-Application a l'évolution des connaissances de l'apprenant," Université Mouloud Mammeri de Tizi Ouzou, 2010.

[33] O. Chourabi, "Un cadre ontologique générique de modélisation, de capitalisation et de

partage de Connaissances Métiers Situées en Ingénierie Système. (Ontological framework for System Engineering Knowledge modeling, capitalization and sharing)," Conservatoire national des arts et métiers, Paris, France, 2009.

[34] N. Guarino, "Some Organizing Principles For A Unified Top-Level Ontology," in *AAAI 1997 SPRING SYMPOSIUM ON ONTOLOGICAL ENGINEERING (LADSEB-CNR INT. REP. 02/97, V3.0*, 1997.

[35] K. Drame, "Contribution to ontology building and to semantic information retrieval□: application to medical domain," Universit{é} de Bordeaux, 2014.

[36] A. Maedche and S. Staab, "Ontology Learning for the Semantic Web," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 72–79, 2001.

[37] M. Uschold, "Building ontologies: towards a unified methodology," in *Expert Systems '96*, 1996.

[38] A. Farquhar, R. Fikes, and J. Rice, "The Ontolingua Server: a Tool for Collaborative Ontology Construction," in *International Journal of Human-Computer Studies*, 1996.

[39] M. Dean *et al.*, "OWL Web Ontology Language 1.0 Reference," 2002.

[40] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data Knowl. Eng.*, vol. 25, no. 1, pp. 161–197, 1998.

[41] E. M. Bray Tim, C. M. Sperberg-McQueen, Jean Paoli, "Extensible Markup Language (XML) 1.0 (Second Edition)," *W3C Recommendation*, 2000. [Online]. Available: http://www.w3.org/TR/REC-xml. [Accessed: 05-May-2017].

[42] O. Lassila and R. R. Swick, "Resource {D}escription {F}ramework ({RDF}) {M}odel and {S}yntax {S}pecification," 1999.

[43] R. V. Dan Brickley, Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," 2004.

[44] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein, "{OIL} in a Nutshell," in *Knowledge Engineering and Knowledge Management; Methods, Models and Tools, Proceedings of the 12th International Conference EKAW 2000*, 2000, no. LNCS 1937, pp. 1–16.

[45] F. van Harmelen, P. F. Patel-Schneider, and I. Horrocks, Eds., "Reference description of the DAML+OIL (March 2001) ontology markup language." 2001.

[46] M. Loukil, "Gestion de contexte pour l'optimisation de l'acc{è}s et l'adaptation des services sur des environnements h{é}t{é}rog{è}nes. (Context management for network access optimisation and services adaptation in heterogeneous environments)," Telecom {&} Management SudParis, Évry, Essonne, France, 2012.

[47] M. K. Smith, C. Welty, and D. L. McGuinness, "OWL Web Ontology Language," *Recommandation du W3C*, 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-owl-guide-20040210/. [Accessed: 03-May-2017].

[48] G. Schreiber, "OWL: the Web Ontology Language," *W3C day, Evolve Conference, Brisbane*, 2004. [Online]. Available: http://www.cs.vu.nl/~guus/talks/04-owl-brisbane/all.htm. [Accessed: 04-May-2017].

[49] Francis.Lapique, "Le langage d'ontologie Web OWL," 2006.

[50] L. Abrouk, "Les ontologies," 2006.

[51] PROTÉGÉ, "Protégé," 2016. [Online]. Available: http://protege.stanford.edu/. [Accessed: 04-May-2017].

[52] A. Osterwalder, E. Hec, U. De Lausanne, and Y. Pigneur, "An e-Business Model Ontology for Modeling e-Business," 2002.

[53] M. Ehrig, A. Koschmider, and A. Oberweis, "Measuring Similarity Between Semantic Business Process Models," in *Proceedings of the Fourth Asia-Pacific Conference on Comceptual Modelling - Volume 67*, 2007, pp. 71–80.

[54] O. Thomas and M. Fellmann, "Semantic EPC: Enhancing Process Modeling Using Ontology Languages," in *SBPM*, 2007, vol. 251.

[55] A. F. H. K. Markovic, "Organisational Ontology Framework for Semantic Business Process Management," 2009, pp. 1–12.

[56] M. El Kharbili, S. Stein, I. Markovic, and E. Pulverm□ller, "Towards a Framework for Semantic Business Process Compliance Management." pp. 1–15, 2007.

[57] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo, "Querying Business Processes," in *Proceedings of the 32Nd International Conference on Very Large Data Bases*, 2006, pp. 343–354.

[58] H. Fill and P. Burzynski, "P.: Integrating Ontology Models and Conceptual Models using a Meta Modeling Approach," in *In: 11th International Prote□ge□ Conference*, 2009.

[59] K. Belhajjame and M. Brambilla, "Ontology-Based Description and Discovery of Business Processes," vol. 29, pp. 1–15, 2009.

[60] M. Hepp and D. Roman, "An Ontology Framework for Semantic Business Process Management.," in *Wirtschaftsinformatik (1)*, 2007, pp. 423–440.

[61] J. Nitzsche, D. Wutke, and T. van Lessen, "An Ontology for Executable Business Processes," in *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM 2007) held in conjunction with the 3rd European Semantic Web Conference (ESWC 2007) Innsbruck, Austria, June 7, 2007*, 2007, pp. 52–63.

[62] M. H. Muhammad Ahtisham Aslam, Sören Auer, Jun Shen, "Expressing Business Process Models as OWL-S Ontologies," in *Business Process Management WorkshopsVolume 4103 of the series Lecture Notes in Computer Science*, 2006, pp. 400–415.

[63] M.-C. F. M. Dumas, "Chapitre 4 Les Services Web," in *Intergiciel et Construction d'Applications R'eparties*, 2008.

[64] S. I. (editor) David Martin *et al.*, "OWL-S: Semantic Markup for Web Services," 2004. .

[65] M. LALLALI, "Modélisation et Test Fonctionnel de l'Orchestration de Services Web," l'UNIVERSITE D'EVRY-VAL D'ESSONNE, 2009.

[66] I. Diane Jordan and M. John Evdemon, "Web Services Business Process Execution Language Version 2.0," 2007. .

[67] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1." [Online]. Available: http://www.w3.org/TR/2001/NOTE-wsdl-20010315. [Accessed: 28-Mar-2017].

[68] Wikipedia, "NetBeans," 2017. [Online]. Available: https://fr.wikipedia.org/wiki/NetBeans. [Accessed: 08-May-2017].

[69] NetBeans, "Découvrez Java EE 5 avec NetBeans," 2014. [Online]. Available: https://fr.netbeans.org/edi/55/articles/Java EE 5 avec NetBeans.html. [Accessed: 12-May-2017].

[70] Java, "Qu'est-ce que la technologie Java et pourquoi en ai-je besoin□?" [Online]. Available: https://www.java.com/fr/download/faq/whatis_java.xml. [Accessed: 15-May-2017].

[71] J.-M. DOUDOUX, "Développement en java," 2016. [Online]. Available: http://www.jmdoudoux.fr/java/dej/chap-jdom.htm#jdom-1.

[72] Jean-Michel DOUDOUX, "Les modèles de documents," 2016. [Online]. Available: https://www.jmdoudoux.fr/java/dej/chap-jdom.htm. [Accessed: 10-Jun-2017].

[73] S. C. for B. I. Research, "Protégé 5.2.0," 2016. [Online]. Available: http://protege.stanford.edu/products.php. [Accessed: 05-Jun-2017].

Annexes

A. BPEL

```
<process name="Production_BPEL"
targetNamespace="http://enterprise.netbeans.org/bpel/BonDeCommandeBPEL/Pro
duction_BPEL">
 <import
namespace="http://j2ee.netbeans.org/wsdl/BonDeCommandeBPEL/Production_
WSDL" location="Production_WSDL.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
 <import namespace="http://j2ee.netbeans.org/wsdl/Derby_Ble"
location="Derby_Ble.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
 <import namespace="http://j2ee.netbeans.org/wsdl/Derby_StockProduit"
location="Derby_StockProduit.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
 <import
namespace="http://j2ee.netbeans.org/wsdl/BonDeCommandeBPEL/Achat_WSD
L" location="Achat_WSDL.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
 <import
namespace="http://j2ee.netbeans.org/wsdl/Derby_Production_Historique"
location="Derby_Production_Historique.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
 <import namespace="http://j2ee.netbeans.org/wsdl/DerbyRisque"
location="DerbyRisque.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
<partnerLinks>
 <partnerLink name="EtatRisque" partnerLinkType="tns:jdbcpartner"
partnerRole="jdbcPortTypeRole"/>
 <partnerLink name="BleStock" partnerLinkType="tns:jdbcpartner"
partnerRole="jdbcPortTypeRole"/>
 <partnerLink name="Achat_Client" partnerLinkType="tns:Achat_WSDL"
partnerRole="Achat_WSDLPortTypeRole"/>
 <partnerLink name="ProductionHisto" partnerLinkType="tns:jdbcpartner"
partnerRole="jdbcPortTypeRole"/>
 <partnerLink name="UpdateVente" partnerLinkType="tns:jdbcpartner"
partnerRole="jdbcPortTypeRole"/>
 <partnerLink name="ServiceVente"
partnerLinkType="tns:Production_WSDL"
myRole="Production_WSDLPortTypeRole"/>
 </partnerLinks>
<variables>
 <variable name="UpdateInRisque2" messageType="tns:inputMsg"/>
 <variable name="FindInRisc2" messageType="tns:inputMsg"/>
 <variable name="UpdateInRisque" messageType="tns:inputMsg"/>
 <variable name="FindOutRisque" messageType="tns:outputMsg"/>
 <variable name="FindInRisque" messageType="tns:inputMsg"/>
```

```
 <variable name="InsertOuthistoCas1" messageType="tns:insertRetMsg"/>
 <variable name="InsertInhistoCas1" messageType="tns:inputMsg"/>
 <variable name="InsertOuthisotCas2" messageType="tns:insertRetMsg"/>
 <variable name="InsertInhistoCas2" messageType="tns:inputMsg"/>
 <variable name="UpdateInNewQteBle" messageType="tns:inputMsg"/>
 <variable name="FindOutNewQteBle" messageType="tns:outputMsg"/>
 <variable name="FindInNewQteBle" messageType="tns:inputMsg"/>
 <variable name="Achat_WSDLOperationOut"
messageType="tns:Achat_WSDLOperationResponse"/>
 <variable name="Achat_WSDLOperationIn"
messageType="tns:Achat_WSDLOperationRequest"/>
 <variable name="FindOutProduct" messageType="tns:outputMsg"/>
 <variable name="FindInProduct" messageType="tns:inputMsg"/>
 <variable name="UpdateInSemoule" messageType="tns:inputMsg"/>
 <variable name="UpdateInBle" messageType="tns:inputMsg"/>
 <variable name="FindOut" messageType="tns:outputMsg"/>
 <variable name="FindIn" messageType="tns:inputMsg"/>
 <variable name="Production_WSDLOperationOut"
messageType="tns:Production_WSDLOperationResponse"/>
 <variable name="Production_WSDLOperationIn"
messageType="tns:Production_WSDLOperationRequest"/>
 </variables>
<sequence>
 <receive name="Recevoir" createInstance="yes" partnerLink="ServiceVente"
operation="Production_WSDLOperation"
portType="tns:Production_WSDLPortType"
variable="Production_WSDLOperationIn"/>
<sequence name="Sequence3">
<assign name="Assign4">
<copy>
 <from>'Ble'</from>
 <to>$FindIn.part/ns0:STOCK_BLE_Record/ns0:NOM</to>
 </copy>
 </assign>
 <invoke name="Verification_ressource" partnerLink="BleStock"
operation="find" portType="tns:jdbcPortType" inputVariable="FindIn"
outputVariable="FindOut"/>
<assign name="EnCasDeRisque">
<copy>
 <from>'Retard'</from>
 <to>$FindInRisque.part/ns0:RISQUE_Record/ns0:RISQUE</to>
 </copy>
 </assign>
 <invoke name="VrificationRisque" partnerLink="EtatRisque"
operation="find" portType="tns:jdbcPortType"
inputVariable="FindInRisque" outputVariable="FindOutRisque"/>
<if name="Ifressource">
 <condition>$FindOut.part/ns0:STOCK_BLE_Record/ns0:QTE >
$Production_WSDLOperationIn.part1/Qte_Product</condition>
<sequence name="Sequence1">
```

```
<if name="Ifrisque1">
 <condition>$FindOutRisque.part/ns0:RISQUE_Record/ns0:ET
AT = 'non'</condition>
<sequence name="Pasderisque">
 <empty name="Produire"/>
<wait name="Tps_Prod">
 <for>'P0Y0M0DT0H0M0S'</for>
 </wait>
<assign name="rep1">
<copy>
 <from>$Production_WSDLOperationIn.part1/DateProd
</from>
 <to>$Production_WSDLOperationOut.part1/DateProd</
to>
 </copy>
<copy>
 <from>'Votre commande sera livré apres 1 jours
!'</from>
 <to>$Production_WSDLOperationOut.part1/TYpe_Pod
uct</to>
 </copy>
<copy>
 <from>'Normal'</from>
 <to>$Production_WSDLOperationOut.part1/DateRe</to
>
 </copy>
 </assign>
 </sequence>
<else>
<sequence name="Risque">
<wait name="TpsRes">
 <for>'P0Y0M0DT0H0M0S'</for>
 </wait>
<assign name="Resolution">
<copy>
 <from>'non'</from>
 <to>$UpdateInRisque.part/ns0:RISQUE_Record/ns0:
ETAT</to>
 </copy>
 </assign>
 <invoke name="PrblmResolu" partnerLink="EtatRisque"
operation="update" portType="tns:jdbcPortType"
inputVariable="UpdateInRisque"/>
 <empty name="production"/>
<wait name="tpsprod">
 <for>'P0Y0M0DT0H0M0S'</for>
 </wait>
<assign name="rep2">
<copy>
```

```
 <from>$Production_WSDLOperationIn.part1/Date2</
from>
 <to>$Production_WSDLOperationOut.part1/DatePro
d</to>
 </copy>
<copy>
 <from>'Votre commande sera livré apres 2 jours
!'</from>
 <to>$Production_WSDLOperationOut.part1/TYpe_Po
duct</to>
 </copy>
<copy>
 <from>'Panne'</from>
 <to>$Production_WSDLOperationOut.part1/DateRe<
/to>
 </copy>
 </assign>
 </sequence>
 </else>
 </if>
<assign name="Assign5">
<copy>
 <from>$FindOut.part/ns0:STOCK_BLE_Record/ns0:QTE -
$Production_WSDLOperationIn.part1/Qte_Product</from>
 <to>$UpdateInBle.part/ns0:STOCK_BLE_Record/ns0:QTE<
/to>
 </copy>
 </assign>
 <invoke name="UpdateBle" partnerLink="BleStock"
operation="update" portType="tns:jdbcPortType"
inputVariable="UpdateInBle"/>
<assign name="Assign12">
<copy>
 <from>$Production_WSDLOperationIn.part1/Qte_Product</
from>
 <to>$InsertInhistoCas1.part/ns0:HESTORIQUE_PRODUIR
E_Record/ns0:QTEDEMANDE</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/TYpe_Poduct<
/from>
 <to>$InsertInhistoCas1.part/ns0:HESTORIQUE_PRODUIR
E_Record/ns0:NOMPRODUIT</to>
 </copy>
<copy>
 <from>$FindOut.part/ns0:STOCK_BLE_Record/ns0:QTE</f
rom>
 <to>$InsertInhistoCas1.part/ns0:HESTORIQUE_PRODUIR
E_Record/ns0:QTEAVANT</to>
 </copy>
```

```
<copy>
 <from>$FindOut.part/ns0:STOCK_BLE_Record/ns0:QTE -
$Production_WSDLOperationIn.part1/Qte_Product</from>
 <to>$InsertInhistoCas1.part/ns0:HESTORIQUE_PRODUIR
E_Record/ns0:QTE</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationOut.part1/DateProd</fr
om>
 <to>$InsertInhistoCas1.part/ns0:HESTORIQUE_PRODUIR
E_Record/ns0:DATE_PROD</to>
 </copy>
 </assign>
 <invoke name="HistoriqueCas1" partnerLink="ProductionHisto"
operation="insert" portType="tns:jdbcPortType"
inputVariable="InsertInhistoCas1"
outputVariable="InsertOuthistoCas1"/>
 </sequence>
<else>
<sequence name="Sequence2">
<assign name="Assign8">
<copy>
 <from>$FindOut.part/ns0:STOCK_BLE_Record/ns0:NOM
</from>
 <to>$Achat_WSDLOperationIn.part1/Nom_Prod</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/Qte_Product
-
$FindOut.part/ns0:STOCK_BLE_Record/ns0:QTE</from>
 <to>$Achat_WSDLOperationIn.part1/Qte_Pro</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/BCforn</fro
m>
 <to>$Achat_WSDLOperationIn.part1/BCforn</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/Date3</from
>
 <to>$Achat_WSDLOperationIn.part1/Dateachat</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/Date4</from
>
 <to>$Achat_WSDLOperationIn.part1/DateForn</to>
 </copy>
 </assign>
 <invoke name="Appro" partnerLink="Achat_Client"
operation="Achat_WSDLOperation"
```

```
portType="tns:Achat_WSDLPortType"
inputVariable="Achat_WSDLOperationIn"
outputVariable="Achat_WSDLOperationOut"/>
<if name="Ifrisque2">
<condition>$FindOutRisque.part/ns0:RISQUE_Record/ns0:E
TAT = 'non'</condition>
<sequence name="pasDerisque">
<empty name="Produir"/>
<wait name="TpsProd">
<for>'P0Y0M0DT0H0M0S'</for>
</wait>
<assign name="rep3">
<copy>
<from>$Achat_WSDLOperationOut.part1/Nom_Prod
</from>
<to>$Production_WSDLOperationOut.part1/TYpe_Po
duct</to>
</copy>
<copy>
<from>$Production_WSDLOperationIn.part1/Date2</
from>
<to>$Production_WSDLOperationOut.part1/DatePro
d</to>
</copy>
<copy>
<from>'Normal'</from>
<to>$Production_WSDLOperationOut.part1/DateRe<
/to>
</copy>
</assign>
</sequence>
<else>
<sequence name="AvecRisque">
<wait name="TpsReso">
<for>'P0Y0M0DT0H0M0S'</for>
</wait>
<assign name="ResRisx">
<copy>
<from>'non'</from>
<to>$UpdateInRisque2.part/ns0:RISQUE_Record/n
s0:ETAT</to>
</copy>
</assign>
<invoke name="RiscResolu" partnerLink="EtatRisque"
operation="update" portType="tns:jdbcPortType"
inputVariable="UpdateInRisque2"/>
<empty name="Prod"/>
<wait name="TpsPro">
<for>'P0Y0M0DT0H0M0S'</for>
</wait>
```

```
<assign name="rep4">
<copy>
<from>$Production_WSDLOperationIn.part1/Date3
</from>
<to>$Production_WSDLOperationOut.part1/DateP
rod</to>
</copy>
<copy>
<from>'Panne'</from>
<to>$Production_WSDLOperationOut.part1/DateR
e</to>
</copy>
<copy>
<from>'Dsl cette commande serra retardé ,votre
produit serra livré apres 4 jours '</from>
<to>$Production_WSDLOperationOut.part1/TYpe_
Poduct</to>
</copy>
</assign>
</sequence>
</else>
</if>
<assign name="Assign9">
<copy>
<from>'Ble'</from>
<to>$FindInNewQteBle.part/ns0:STOCK_BLE_Record/ns
0:NOM</to>
</copy>
</assign>
<invoke name="FindNewQteBle" partnerLink="BleStock"
operation="find" portType="tns:jdbcPortType"
inputVariable="FindInNewQteBle"
outputVariable="FindOutNewQteBle"/>
<assign name="Assign10">
<copy>
<from>$FindOutNewQteBle.part/ns0:STOCK_BLE_Recor
d/ns0:QTE -
$Production_WSDLOperationIn.part1/Qte_Product</from>
<to>$UpdateInNewQteBle.part/ns0:STOCK_BLE_Record/
ns0:QTE</to>
</copy>
</assign>
<invoke name="UpdateQteBle" partnerLink="BleStock"
operation="update" portType="tns:jdbcPortType"
inputVariable="UpdateInNewQteBle"/>
<assign name="Assign11">
<copy>
<from>$Production_WSDLOperationIn.part1/Qte_Product
</from>
```

```
 <to>$InsertInhistoCas2.part/ns0:HESTORIQUE_PRODUI
RE_Record/ns0:QTEDEMANDE</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/TYpe_Poduc
t</from>
 <to>$InsertInhistoCas2.part/ns0:HESTORIQUE_PRODUI
RE_Record/ns0:NOMPRODUIT</to>
 </copy>
<copy>
 <from>$FindOut.part/ns0:STOCK_BLE_Record/ns0:QTE
</from>
 <to>$InsertInhistoCas2.part/ns0:HESTORIQUE_PRODUI
RE_Record/ns0:QTEAVANT</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/Qte_Product
-
$FindOut.part/ns0:STOCK_BLE_Record/ns0:QTE</from>
 <to>$InsertInhistoCas2.part/ns0:HESTORIQUE_PRODUI
RE_Record/ns0:QTE</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationOut.part1/DateProd<
/from>
 <to>$InsertInhistoCas2.part/ns0:HESTORIQUE_PRODUI
RE_Record/ns0:DATE_PROD</to>
 </copy>
 </assign>
 <invoke name="HestoriqueCas2" partnerLink="ProductionHisto"
operation="insert" portType="tns:jdbcPortType"
inputVariable="InsertInhistoCas2"
outputVariable="InsertOuthisotCas2"/>
 </sequence>
 </else>
 </if>
 </sequence>
<assign name="Assign7">
<copy>
 <from>$Production_WSDLOperationIn.part1/TYpe_Poduct</from>
 <to>$FindInProduct.part/ns0:STOCK_PRODUIT_Record/ns0:TYP
E_PRODUIT</to>
 </copy>
 </assign>
 <invoke name="ProdType" partnerLink="UpdateVente" operation="find"
portType="tns:jdbcPortType" inputVariable="FindInProduct"
outputVariable="FindOutProduct"/>
<assign name="Assign6">
<copy>
```

```
 <from>$FindOutProduct.part/ns0:STOCK_PRODUIT_Record/ns0:
QTE_PRODUIT +
$Production_WSDLOperationIn.part1/Qte_Product</from>
 <to>$UpdateInSemoule.part/ns0:STOCK_PRODUIT_Record/ns0:Q
TE_PRODUIT</to>
 </copy>
<copy>
 <from>$Production_WSDLOperationIn.part1/TYpe_Poduct</from>
 <to>$UpdateInSemoule.part/ns0:STOCK_PRODUIT_Record/ns0:T
YPE_PRODUIT</to>
 </copy>
 </assign>
 <invoke name="StockSmoule" partnerLink="UpdateVente"
operation="update" portType="tns:jdbcPortType"
inputVariable="UpdateInSemoule"/>
 <reply name="Reply1" partnerLink="ServiceVente"
operation="Production_WSDLOperation"
portType="tns:Production_WSDLPortType"
variable="Production_WSDLOperationOut"/>
 </sequence>
 </process>
```

B. WSDL

```
<definitions name="Production_WSDL"
targetNamespace="http://j2ee.netbeans.org/wsdl/BonDeCommandeBPEL/Productio
n_WSDL">
<types>
<xsd:schema
targetNamespace="http://j2ee.netbeans.org/wsdl/BonDeCommandeBPEL/Prod
uction_WSDL">
<xsd:complexType name="ProductionType">
<xsd:sequence>
 <xsd:element name="TYpe_Poduct" type="xsd:string"/>
 <xsd:element name="Qte_Product" type="xsd:int"/>
 <xsd:element name="DateProd" type="xsd:string"/>
 <xsd:element name="DateRe" type="xsd:string"/>
 <xsd:element name="Date2" type="xsd:string"/>
 <xsd:element name="Date3" type="xsd:string"/>
 <xsd:element name="Date4" type="xsd:string"/>
 <xsd:element name="BCforn" type="xsd:string"/>
 </xsd:sequence>
 </xsd:complexType>
 <xsd:element name="Production" type="tns:ProductionType"/>
 </xsd:schema>
 </types>
<message name="Production_WSDLOperationRequest">
 <part name="part1" element="tns:Production"/>
 </message>
<message name="Production_WSDLOperationResponse">
 <part name="part1" element="tns:Production"/>
```

```
 </message>
<portType name="Production_WSDLPortType">
<operation name="Production_WSDLOperation">
 <input name="input1"
message="tns:Production_WSDLOperationRequest"/>
 <output name="output1"
message="tns:Production_WSDLOperationResponse"/>
 </operation>
 </portType>
<binding name="Production_WSDLBinding"
type="tns:Production_WSDLPortType">
 <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="Production_WSDLOperation">
 <soap:operation/>
<input name="input1">
 <soap:body use="literal"
namespace="http://j2ee.netbeans.org/wsdl/BonDeCommandeBPEL/Produ
ction_WSDL"/>
 </input>
<output name="output1">
 <soap:body use="literal"
namespace="http://j2ee.netbeans.org/wsdl/BonDeCommandeBPEL/Produ
ction_WSDL"/>
 </output>
 </operation>
 </binding>
<service name="Production_WSDLService">
<port name="Production_WSDLPort"
binding="tns:Production_WSDLBinding">
 <soap:address
location="http://localhost:${HttpDefaultPort}/Production_WSDLService/Pr
oduction_WSDLPort"/>
 </port>
 </service>
<plnk:partnerLinkType name="Production_WSDL">
 <plnk:role name="Production_WSDLPortTypeRole"
portType="tns:Production_WSDLPortType"/>
 </plnk:partnerLinkType>
 </definitions>
```

c. L'ontologie OWL

```
?xml version="1.0"?>



<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
```

```
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#"
     xml:base="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37"/>
  <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasActivity --
>


    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#hasActivity">
        <rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#Activity"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#Process"/>
    </owl:ObjectProperty>
     <!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#hasCorrelationSet -->
    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#hasCorrelationSet">
```

<rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#CorrelationSets"/>

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Process"/>

</owl:ObjectProperty>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasEventHandler -->

<owl:ObjectProperty rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasEventHandler">

<rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#EventHandlers"/>

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Process"/>

</owl:ObjectProperty>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasFaultHandler -->

<owl:ObjectProperty rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasFaultHandler">

<rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#FaultHandlers"/>

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Process"/>

</owl:ObjectProperty>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasInPut -->

<owl:ObjectProperty rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasInPut">

<rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#InPut"/>

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Operation"/>

</owl:ObjectProperty>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasMessage -->

<owl:ObjectProperty rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasMessage">

```
<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#hasPortType">

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#PartnerLinkType"/>

<rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#PortType"/>

</owl:ObjectProperty>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasRole -->

<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#hasRole">

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#PartnerLinkType"/>

<rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#Role"/>

</owl:ObjectProperty>


<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#hasVariable -
->


<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#hasVariable">

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#Process"/>

<rdfs:range rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#VariableBPEL"/>

</owl:ObjectProperty>

<!--

///////////////////////////////////////////////////////////////////////////////

//

// Data properties

//

///////////////////////////////////////////////////////////////////////////////

-->
```

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#namePartnerLink -->

<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#namePartnerLink">

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PartnerLinks"/>

<rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#nameRole -->

<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#nameRole">

<rdfs:domain rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Role"/>

</owl:DatatypeProperty>

<!--

//

//

// Classes

//

//

-->

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Activity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Activity"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AssignActivity -->

```
<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AssignActivity">
    <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AtomicActivity"/>
</owl:Class>
<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AsynchroneActivity -->
<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AsynchroneActivity">
    <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#CommunicatedActivity"/>
</owl:Class>
<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AtomicActivity -->
<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AtomicActivity">
    <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#BasedActivity"/>
</owl:Class>
<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#BasedActivity -->
<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#BasedActivity">
    <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Activity"/>
</owl:Class>
<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#CommunicatedActivity -->
<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#CommunicatedActivity">
```

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Activity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ControlerActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ControlerActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#StructuredActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#CorrelationSets -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#CorrelationSets"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#EmptyActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#EmptyActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AtomicActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#EventHandlers -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#EventHandlers"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ExitActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ExitActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AtomicActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#FaultHandlers -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#FaultHandlers"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#FlowActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#FlowActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ParallalActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#InPut -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#InPut"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#InvokeAsynchroneActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#InvokeAsynchroneActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AsynchroneActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#InvokeSynchroneActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#InvokeSynchroneActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SynchroneActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Message -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Message"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Operation -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Operation"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#OutPut -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#OutPut"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ParallalActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ParallalActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#StructuredActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PartnerLinkType -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PartnerLinkType"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PartnerLinks -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PartnerLinks"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PickActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PickActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ControlerActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PortType -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#PortType"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Process -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Process"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ReceiveActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ReceiveActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SynchroneActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#RepeatUntilActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#RepeatUntilActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SequantialActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ReplyActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ReplyActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SynchroneActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Role -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Role"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SequantialActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SequantialActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#StructuredActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SquenceActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SquenceActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SequantialActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#StructuredActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#StructuredActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Activity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SynchroneActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#SynchroneActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#CommunicatedActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ThrowActivity -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ThrowActivity">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#AtomicActivity"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#VariableBPEL -->

<owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#VariableBPEL">

<rdfs:subClassOf rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Variables"/>

</owl:Class>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#VariableCorrelationSets -->

```
    <owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#VariableCorrelationSets">

        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#Variables"/>

    </owl:Class>

     <!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#VariableFault -->

    <owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#VariableFault">

        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#Variables"/>

    </owl:Class>

     <!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Variables --
>

    <owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#Variables"/>

    <!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#WhileActivity -->

    <owl:Class rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-
ontology-37#WhileActivity">

        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#SequantialActivity"/>

    </owl:Class>

   <!--

    ///////////////////////////////////////////////////////////////////////////////

    //

    // Individuals

    //

    ///////////////////////////////////////////////////////////////////////////////

     -->

   <!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#FindInRisc2 -
->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-
37#FindInRisc2"/>
```

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_BPEL -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_BPEL"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDL -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDL"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDLOperation -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDLOperation"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDLPortType -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDLPortType"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDLPortTypeRole -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#Production_WSDLPortTypeRole"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ServiceVente -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#ServiceVente"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#UpdateInRisque -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#UpdateInRisque"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#UpdateInRisque2 -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#UpdateInRisque2"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#input1 -->

<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#input1"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#output1 -->

<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#output1"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#tns:Production_WSDLOperationRequest -->

<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#tns:Production_WSDLOperationRequest"/>

<!-- http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#tns:Production_WSDLOperationResponse -->

<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/hp/ontologies/2017/5/untitled-ontology-37#tns:Production_WSDLOperationResponse"/>

</rdf:RDF>