Une Ontologie des Processus Métier (BBO) pour guider un Agent Virtuel

Amina Annane, Nathalie Aussenac-Gilles, Mouna Kamel

▶ **To cite this version:**

Amina Annane, Nathalie Aussenac-Gilles, Mouna Kamel. Une Ontologie des Processus Métier (BBO) pour guider un Agent Virtuel. 30èmes Journées Francophones d'Ingénierie des Connaissances (IC 2019), AFIA, Jul 2019, Toulouse, France. pp.183-198. hal-02284535

HAL Id: hal-02284535
https://hal.science/hal-02284535

Submitted on 11 Sep 2019

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

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

Une Ontologie des Processus Métier (BBO)

Amina Annane, Nathalie Aussenac-Gilles, Mouna Kamel

IRIT, CNRS – Université de Toulouse, Toulouse, France
prénom.nom@irit.fr

Résumé : Les activités de recherche dans le domaine de la Modélisation de Processus Métier ont donné lieu à différentes propositions de modèles et ontologies, certains étant reconnus comme standards. Mais aucun de ces modèles n'est, à notre connaissance, suffisamment riche pour permettre à un agent virtuel de superviser l'exécution d'un processus métier et d'exploiter des Retours d'EXpériences (REX) en cas d'anomalie. Notre participation au projet AVI-REX, en partenariat avec de grandes entreprises industrielles, a pour objectif de construire une telle ontologie. Le travail présenté dans cet article correspond à la première étape de la construction, à savoir la description de processus métier, avec toutes les connaissances requises pour leur exécution. Pour répondre aux besoins spécifiés, nous nous sommes basés sur plusieurs modèles existants, que nous avons intégrés puis enrichis. L'ontologie obtenue, appelée BBO, a été évaluée selon différents critères, puis mise à disposition de la communauté.

Mots-clés : Modélisation de processus métier, Web sémantique, Ontologie, Intégration de données

1 Introduction

Les processus métier représentent le savoir-faire des entreprises : "*a process is a particular procedure for doing something involving one or more steps or operations. The process may produce a product, a property of a product, or an aspect of a product*" (ISO 10303-49). La Modélisation de Processus Métier (BPM pour Business Process Modeling en anglais) permet d'analyser, améliorer, simuler et automatiser ces processus (Rospocher *et al.*, 2014). A l'heure de l'industrie 4.0 où l'ambition est de rendre la communication toujours plus efficace et performante, d'une part entre les différents systèmes informatiques, et d'autre part, entre les humains et ces systèmes, bien modéliser les processus prend de plus en plus d'importance.

Les technologies du web sémantique constituent des solutions prometteuses pour réaliser cette ambition (Vogel-Heuser & Hess, 2016). Les ontologies notamment permettent de décrire les connaissances de façon formelle (Berners-Lee *et al.*, 2001) et possèdent des capacités de raisonnement pouvant assurer la cohérence des processus métiers (Rospocher *et al.*, 2014; Roy *et al.*, 2018). De plus, représenter les processus métiers à l'aide d'ontologies semble bien répondre à la question de la communication et de l'interopérabilité entre différents systèmes.

Dans ce contexte, le projet AVI-REX vise la réalisation d'un modèle d'agent virtuel pour superviser et guider l'exécution des processus métier, s'inscrivant ainsi dans l'ère de l'industrie 4.0. Ce projet regroupe l'entreprise SimSoft Industry, qui conçoit et implémente l'agent virtuel, l'IRIT [1] et deux entreprises industrielles futures utilisatrices de ces agents : Thales Alenia Space (TAS) et Continental. Le rôle prévu de cet agent virtuel sera le suivant :

1. guider pas à pas l'opérateur et contrôler le bon déroulement de l'exécution des processus métiers ;
2. répondre aux questions que l'opérateur peut poser sur l'exécution du processus ;
3. exploiter les REX (Retours d'EXpérience) et aider les experts métiers à poser un diagnostic en cas d'anomalie survenue lors de l'exécution des processus.

Pour une entreprise donnée, le modèle d'agent est adapté au contexte du poste de travail dans lequel il doit s'intégrer, et cela en lui fournissant des connaissances (sous la forme d'une base de connaissances) propres aux procédures réalisées à ce poste et aux ressources utilisées. Notre contribution au sein d'AVI-REX consiste à construire une ontologie permettant de décrire le plus finement possible les processus métier, ainsi que les traces de leurs

1. https ://www.irit.fr

exécutions pour réaliser la base de connaissances d'un tel agent virtuel. C'est cette ontologie qui sera alors livrée aux clients, qui auront à charge de la spécialiser et de la peupler selon les caractéristiques de leurs processus métiers respectifs.

Dans cet article, nous détaillons le processus qui, après un état de l'art et une spécification des besoins, nous a conduit à construire une ontologie de domaine. Nous présentons ici uniquement la partie de l'ontologie décrivant des processus métier, en détaillant les connaissances nécessaires qui permettront, plus tard, de modéliser les retours d'expériences. Dans la suite, nous désignerons cette partie d'ontologie simplement par "l'ontologie" ou "BBO". La construction de BBO s'appuie sur les quatre étapes majeures (Figure 1) qu'intègrent la plupart des méthodes de construction d'ontologies (Fernández *et al.*, 1997) :

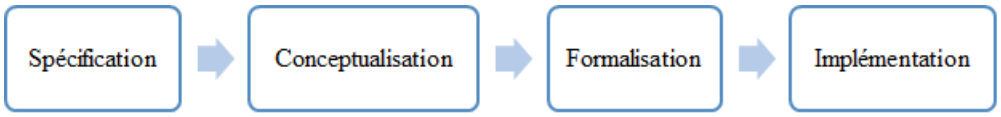

FIGURE *1 – Principales étapes de la construction d'ontologie*

Suivant ce canevas, nous avons :
— spécifié les besoins que doit satisfaire l'ontologie à l'aide de documents techniques et d'experts du domaine, ainsi que d'un ensemble de questions portant sur les compétences techniques (ou "competency questions") ;
— identifié les modèles et ressources existants et réutilisables, puis spécialisé certains concepts de ces modèles ;
— formalisé le contenu de ces modèles ainsi qu'un ensemble de spécifications décrites en Langage Naturel (LN) associées aux modèles réutilisés, ou ayant été exprimées par les experts du domaine ;
— implémenté, documenté et évalué l'ontologie.

Dans la suite de l'article, nous dressons un état de l'art (section 2), puis nous décrivons chacune des étapes de la construction de BBO : la section 3 présente les spécifications, la section 4 la conceptualisation, la section 5 la formalisation et enfin la section 6 l'évaluation. Nous concluons notre article et donnons les suites envisagées pour ce travail en section 7.

2 Travaux connexes

Nous avons alors cherché dans l'état de l'art des ontologies et modèles à intégrer ou aligner pour construire une ontologie des processus métier qui soit à la fois générale (pour toute entreprise) et très précise dans la description des ressources, agents et équipements utilisés.

Le modèle qui semble faire référence dans le domaine est BPMN (OMG, 2011), qui bénéficie de spécifications assez précises et d'une logique d'exécution bien définie. Une partie de ces spécifications est exprimée en UML et au format XML, et une partie en langage naturel. Les spécifications de BPMN 1.0 (datant de 2008) ont donné lieu à la construction semi-automatique d'une ontologie BPMN 1.0, qui a été enrichie par des axiomes et documentée manuellement (Rospocher *et al.*, 2014). Les nouvelles spécifications de 2011, qui intègrent cette fois la sémantique relative à l'exécution des processus, ont également donnée lieu à la construction d'ontologies, comme l'ontologie BPMN 2.0 présentée dans (Natschläger, 2011) qui, à notre connaissance, n'est pas accessible à la communauté. Une autre ontologie a été extraite des spécifications BPMN 2.0 [2], mais sans prise en compte du texte décrivant les classes ni documentation. Pour tous ces travaux, l'idée était de traduire tout le méta-modèle BPMN en une ontologie, ce qui n'est pas notre objectif, certaines parties étant inutiles au regard de nos besoins. En revanche, plusieurs auteurs soulignent que le modèle BPMN ne couvre pas

2. https ://dkm.fbk.eu/bpmn-ontology

toutes les connaissances requises pour représenter des processus métier : il manque des éléments pour décrire des ressources, les sites de production ou les agents réalisant les processus (Awad *et al.*, 2009; Stroppi *et al.*, 2011; Marcinkowski & Kuciapski, 2012; Bocciarelli *et al.*, 2016). Pour cela, nous avons identifié d'autres ontologies pertinentes : une ontologie pour la gestion des projets logiciels (Ruíz *et al.*, 2004), une ontologie des processus logiciels (Falbo & Bertollo, 2009), une ontologie des processus de maintenance industrielle (Karray *et al.*, 2012), une ontologie des processus de fabrication (Chungoora *et al.*, 2013). Enfin, les descriptions des processus métier de certaines ontologies (Uschold *et al.*, 1998; Pedrinaci *et al.*, 2008; Cabral *et al.*, 2009) sont déjà couvertes par le modèle BPMN, avec un grain plus fin.

Nous avons donc retenu BPMN 2.0 comme modèle de référence, et identifié des ontologies pour en combler certaines lacunes. Nous avons décidé de construire l'ontologie BBO à partir de ces sources faute d'ontologie de référence pour représenter précisément les processus métier et les retour d'expériences.

3 Spécifications

Rappelons que BBO devra pouvoir être exploitée tant pour guider un opérateur lors de l'exécution pas à pas d'un processus métier, que pour aider les experts à poser un diagnostic en cas d'anomalie ou de panne lors de l'exécution d'un processus métier. Afin de bien cerner les exigences à satisfaire, nous avons analysé différentes sources de connaissances.

3.1 Documents techniques

Les deux partenaires industriels, TAS et Continental, ont fourni chacun deux corpus :
— un corpus *PM_TAS* et un corpus *PM_Continental* décrivant des processus métier, soit 20 processus métier décrits au total,
— un corpus *REX_TAS* et un corpus *REX_Continental* décrivant des retours d'expérience (REX), soit 28 REX au total.

L'extrait d'un processus métier de chez TAS (que nous avons rendu anonyme) de la table 1 montre un processus composé de 3 activités à exécuter séquentiellement. Une activité peut, à son tour, être composée d'une ou plusieurs actions (e.g., activité 3). Nous appelons *tâche* ou *action* l'élément atomique dans la description d'un processus. Une action (caractérisée par un verbe) peut faire référence à différents types de ressources (*fichier*, *logiciel*, *dispositif*, etc.), à des valeurs de paramètres (*code erreur*), à des instructions conditionnelles (*si l'exécution a généré...*), etc. Le lieu d'exécution du processus est également indiqué (*sur la station MMA*). On note également un cas de polysémie : la station MMA représente à la fois une ressource et un lieu (i.e., un poste de travail). Ce petit exemple montre la finesse de description requise. Ces documents techniques ont été analysés avec l'aide des experts impliqués dans le projet.

```
Sur la station MMA :
1. Exécuter le fichier MUXF.exe
2. En utilisant le logiciel SOFT, vérifier si l'exécution
du script a généré une alarme.
3. Si l'exécution a généré une alarme, exécuter le
processus de résolution d'anomalie N°XY
```

TABLE *1 – Extrait anonyme d'un processus métier TAS*

3.2 Questions basées sur les compétences techniques

Les "competency questions" sont reconnues pour être un bon moyen de spécifier les besoins d'une ontologie (Grüninger & Fox, 1995). Un ensemble de questions de ce type a

émergé suite à notre collaboration avec les experts du projet. Cet ensemble a été enrichi de questions de même type issues de la littérature (Falbo & Bertollo, 2009; Abdalla *et al.*, 2014). Nous avons obtenu 20 questions principales [3], dont nous donnons un extrait dans la Table 2. Ces questions ont permis de mettre en évidence l'importance de certaines entités comme les activités, les ressources, etc.

Quelles ressources sont nécessaires à une activité ?
Quelles ressources sont produites par une activité ?
En quelles sous-activités une activité peut-elle se décomposer ?
Quelle est la nature/le type d'une ressource ?
Quelles activités doit précéder/suivre une activité donnée ?
Qui doit exécuter une activité donnée ?
Où l'activité doit-elle être exécutée ?
...

TABLE *2 – Extrait de l'ensemble des questions basées sur les compétences techniques*

3.3 Synthèse

L'étude de ces sources de connaissance a permis de circonscrire les exigences auxquelles devait répondre l'ontologie, et de caractériser la notion de processus métier. Un processus métier se décompose en un ensemble d'activités (en moyenne une vingtaine), une activité pouvant être un sous-processus ou une tâche ; la tâche est l'élément atomique réalisé par un agent (i.e., opérateur). Une tâche peut nécessiter ou produire une ou plusieurs ressources de natures diverses : matérielles, logicielles, humaines, données, etc. Les activités doivent pouvoir s'exécuter en séquence, en parallèle ou de façon itérative. Elles doivent pouvoir être contrôlées par des évènements (interruption, reprise, etc.). Il est important de connaître la composition et le site de production (entreprise, station de travail, etc.) des produits manufacturés, ainsi que la valeur de certains paramètres qui peuvent conditionner l'exécution des tâches. Enfin, il est important de spécifier le rôle (responsabilité ou autre) de l'agent qui peut/doit réaliser une activité donnée. L'agent peut être une ressource humaine ou une ressource logicielle.

4 Conceptualisation

Suite à l'analyse des spécifications, nous avons identifié cinq entités principales, représentées par des concepts de BBO : *Processus*, *Entrées/Sorties des activités*, *Agent*, *Produit Manufacturé* et *Site de production*. Dans la suite, nous présentons les modèles conceptuels associés à chacun de ces concepts en utilisant des diagrammes de classes UML.

Construire une ontologie "from scratch" est coûteux en termes de temps et d'effort. L'état de l'art présenté en section 2 nous a permis d'identifier les ressources pertinentes pour répondre aux besoins et que nous avons réutilisées : (i) le modèle BPMN 2.0 (OMG, 2011), standard dans le domaine de la représentation des processus métier, qui constituera le noyau de l'ontologie ; (ii) la taxonomie des ressources proposée par (Falbo & Bertollo, 2009; Karray *et al.*, 2012) ; (iii) la taxonomie des sites de production de (Chungoora *et al.*, 2013) et (Fraga *et al.*, 2018) ; (iv) le fragment d'ontologie des agents présent dans (Ruíz *et al.*, 2004).

Pour plus de lisibilité, nous avons scindé la présentation de BBO en cinq sous-sections, chacune correspondant à un concept pivot cité ci-dessus. Par ailleurs, pour différencier les concepts issus du modèle BPMN (modèle ayant servi de base à BBO) des autres sur les diagrammes de classes, nous avons (i) souligné les concepts issus d'ontologies existantes (cf. le concept `UO:unit` de la Figure 4) et (ii) encadré les nouveaux concepts que nous avons ajoutés (cf. le concept `Parameter` de la Figure 4).

3. Liste des 20 questions : `https://github.com/AminaANNANE/BBO_BPMNbasedOntology`

4.1 Processus

Processus est le concept principal de BBO. Le modèle qui nous a semblé le plus abouti pour représenter les processus métier est le modèle BPMN 2.0 (OMG, 2011). En effet, en plus de son expressivité pour représenter un processus, ce modèle intègre la sémantique d'exécution de processus. Il nous a donc paru intéressant de réutiliser une partie de ce modèle, en laissant de côté les concepts liés à la représentation graphique et aux interactions entre processus (i.e., collaboration, conversation, choreography, etc.). Les principaux concepts et relations du fragment réutilisé sont décrits en Figure 2.

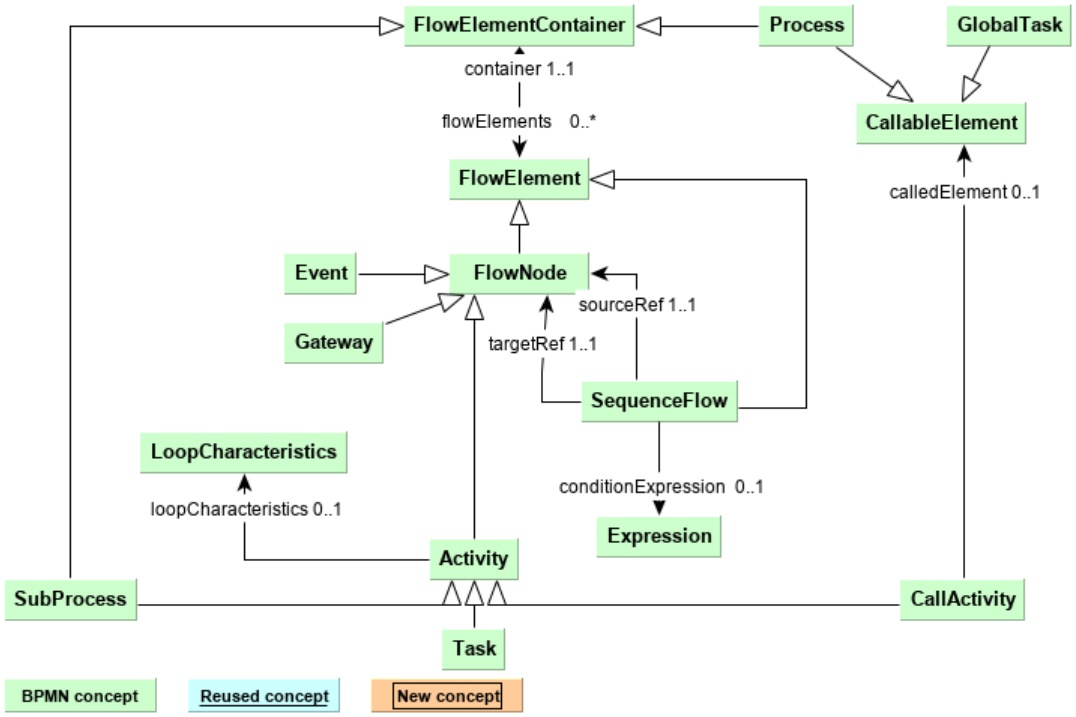

FIGURE 2 – *Diagramme de classes correspondant à la représentation des processus*

Un processus (*Process*) est considéré comme un regroupement (*FlowElementContainer*) d'éléments de flux (*FlowElement*) de deux types :

— *FlowNode* qui désigne une activité (*Activity*), un évènement (*Event*), une passerelle (*Gateway*)
— *SequenceFlow* qui correspond à un connecteur entre deux *FlowNode*. Un connecteur peut être soumis à condition (*Expression*).

Une activité (*Activity*) représente le travail à faire. Elle est spécialisée en trois sous-classes :

— une tâche atomique (*Task*)
— une tâche complexe regroupant plusieurs tâches atomiques (*SubProcess*)
— Une activité d'appel (*CallActivity*) qui appelle un autre processus (*Process*) ou une tâche réutilisable (*GlobalTask*) : les sous classes de *CallableElement*.

La classe *LoopCharacteristics* représente une activité à exécuter de façon itérative.

Un évènement (*Event*) peut survenir au cours de l'exécution d'un processus : le processus peut par exemple être interrompu ou reprendre son exécution s'il avait été interrompu. Un évènement a généralement une cause ou un impact. Différents types d'évènements peuvent être spécifiés : les évènements temporels (*timerEvent*), les évènements conditionnels (*conditionalEvent*), etc.

Une passerelle (*Gateway*) sert à contrôler comment les connecteurs (*SequenceFlow*) interagissent, par exemple s'ils convergent ou s'ils divergent (Figure 3).

Par ailleurs, nous avons enrichi ce modèle selon les spécifications de BPMN 2.0 exprimées en langage naturel. En effet, dans ces spécifications, BPMN décrit les propriétés de certains concepts à l'aide d'attributs valués. Par exemple, l'entité *Gateway* (passerelle) possède un attribut *GatewayDirection* dont la valeur ("convergent", "divergent", "mixte", ou "inconnu") détermine le type. Chaque type de passerelle a alors ses propriétés avec des contraintes propres : les passerelles convergentes doivent être la cible d'au moins deux instances de connecteurs (*SequenceFlow*) et la source d'un seul connecteur, les passerelles divergentes doivent être la cible d'une seule instance de connecteur et la source d'au moins deux connecteurs, etc. (Figure 3). La simple affectation d'une valeur à un attribut n'assure pas la cohérence avec la définition. Pour ces cas-là, nous avons appliqué la règle qui consiste à créer, pour tout attribut *att* de la classe *C*, autant de sous-classes de *C* que de valeurs possibles pour *att*. Pour le concept *Gateway* par exemple, nous avons créé 4 sous-classes correspondant aux quatre valeurs d'attribut.

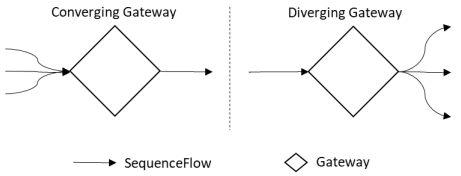

FIGURE *3 – Passerelle convergente vs. passerelle divergente*

Ce même principe a été appliqué à d'autres concepts présentant les mêmes caractéristiques que *Gateway*. La table 3 montre quelques exemples d'enrichissements effectués à partir des spécifications BPMN en langage naturel.

Classe	Sous-Classes
SequenceFlow	ConditionalSequenceFlow, DefaultSequenceFlow, NormalSequenceFlow
SubProcess	EventBasedSubProcess
Gateway	ConvergingGateway, DivergingGateway, MixedGateway, UnspecifiedGateway
EventBasedGateway	ExclusiveEventBasedGateway, ParallelEventBasedGateway
Event	CancelEvent, CompensationEvent, ConditionalEvent, ErrorEvent, EscalationEvent, LinkEvent, MultipleEvent, NoneEvent, ParallelMultipleEvent, SignalEvent, TerminateEvent, TimerEvent

TABLE *3 – Spécialisation des classes BPMN*

Bien que BPMN 2.0 permette de décrire de façon tout à fait satisfaisante les processus métiers, il ne permet pas de représenter certaines connaissances comme les différents types de ressources, les produits manufacturés et les sites des activités, qui, selon les spécifications, sont nécessaires. Par exemple, il n'est pas possible de représenter le fait que la tâche (2) de l'exemple Figure 1 nécessite la ressource "SOFT" de type Logiciel (Software), de donner une valeur attendue à un paramètre d'une ressource donnée, ou encore d'exprimer que les trois tâches doivent être exécutées sur la "station MMA". Il est donc nécessaire d'enrichir ce fragment d'ontologie avec de nouveaux concepts et de nouvelles relations.

4.2 Les Entrées/Sorties des activités

Une activité nécessite des entrées et produit des sorties, sachant que ces entrées/sorties peuvent être de types différents et peuvent être caractérisées par plusieurs valeurs de paramètres. Or le modèle BPMN ne permet de spécifier que des données comme entrées ou sorties d'une activité (*DataInput* et *DataOutput*). Nous avons donc rajouté deux relations "has_resourceInput" et "has_resourceOutput" entre les concepts *InputOutputSpecification* et *Resource* (voir Figure 4).

Par ailleurs, pour représenter les paramètres, leurs valeurs et leurs unités de mesure, nous avons rajouté le fragment d'ontologie décrit dans la Figure 4 (les concepts avec des labels encadrés). Les concepts *Unit* et *Prefix* proviennent de l'ontologie des unités de mesure (UO) [4].

4. http ://purl.obolibrary.org/obo/uo.owl, visited in 2019/03

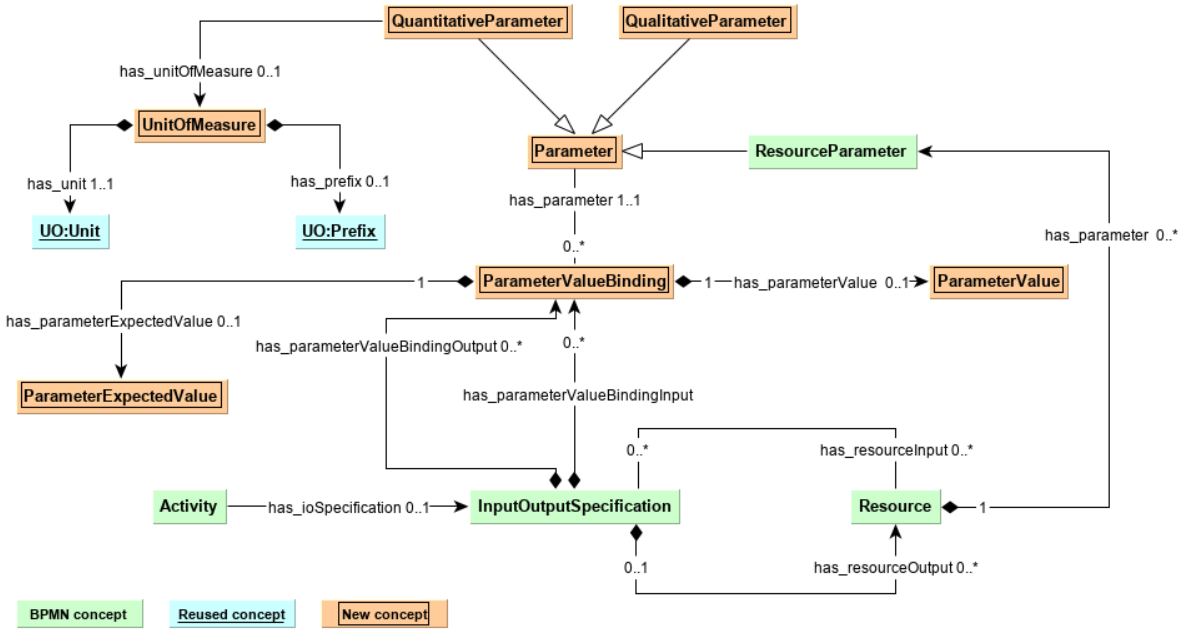

FIGURE *4 – Diagramme de classes correspondant aux entrées/sorties d'activité*

Le concept *Resource* existe dans les spécifications du modèle BPMN 2.0, mais sa sémantique est ambiguë. En effet, la classe *Resource* est supposée couvrir tout type de ressource (p. 95) : "*The Resource class is used to specify resources that can be referenced by Activities. These Resources can be Human Resources as well as any other resource assigned to Activities during Process execution time.*" Mais les relations qui lient les ressources aux processus (p. 148) ou aux activités (p.152), limitent les ressources aux agents responsables de l'exécution du processus : "*. . . defines the resource that will perform or will be responsible for the Process. The resource, e.g., a performer, can be specified in the form of a specific individual, a group, an organization role or position, or an organization.*" Cette définition semble être la plus adoptée. En effet, dans la plupart des travaux qui font référence à BPMN, la notion de ressource est équivalente à celle d'agent (Awad *et al.*, 2009; Stroppi *et al.*, 2011).

Dans BBO, nous adoptons la première définition du concept *Resource*, qui englobe tout type de ressources afin de spécifier les différents types d'entrées/sorties. En effet, nous inspirant des travaux de (Falbo & Bertollo, 2009; Karray *et al.*, 2012), nous avons défini une taxonomie des ressources que nous présentons dans la Figure 5.

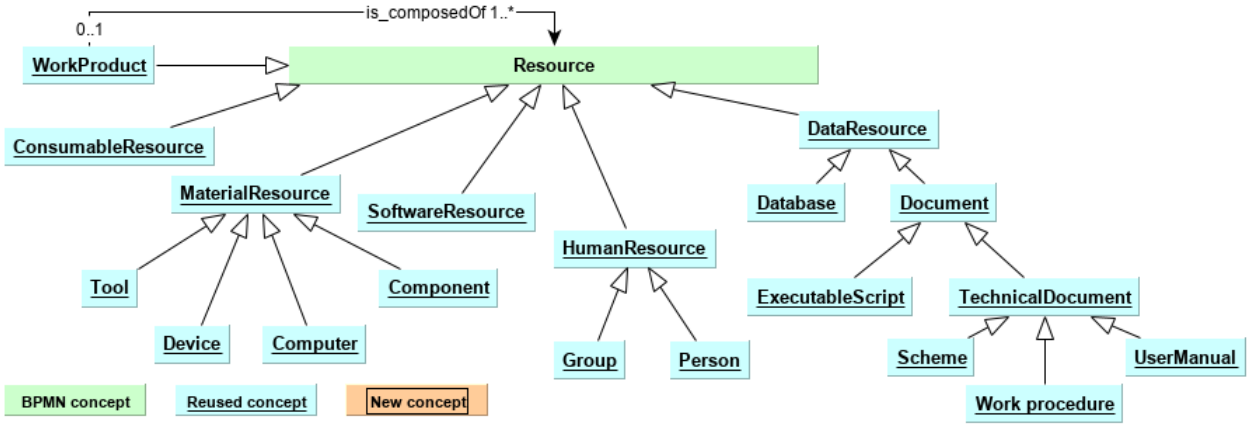

FIGURE *5 – Diagramme de classes correspondant à la taxonomie des ressources*

4.3 Site de production

Nous avons réutilisé la taxonomie proposée par (Chungoora *et al.*, 2013; Fraga *et al.*, 2018) pour décrire le site où une tâche doit être exécutée. Le concept racine *ManufacturingFacility* est spécialisé en cinq sous-classes liées par une relation de méronymie (voir Figure 6), du lieu le plus précis (la station de travail) au lieu le plus global (l'entreprise). Ainsi les sites de production peuvent être associés aux tâches, aux activités, etc.

Le niveau de description le plus fin demande à spécifier le site de production pour chaque tâche (un site par tâche élémentaire) grâce à la relation "takesPlaceAt". Or le modèle permet aussi de spécifier un site pour une activité ou un processus, pour les cas où toutes les tâches de l'activité ou du processus doivent être exécutées sur le même site.

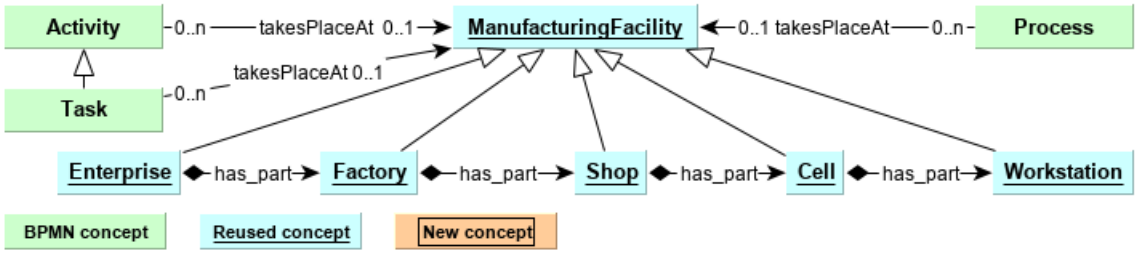

FIGURE *6 – Diagramme UML du concept Site de production*

4.4 Produit manufacturé

Dans les normes ISO 10303-1 et 15531-32, le produit est défini par : "*Thing or substance produced by a natural or artificial process.*". La norme ISO 10303-239, quant à elle, définit la ressource comme suit : "*Resource is the result of a process.*" Ces définitions nous permettent de considérer le produit (*WorkProduct*) résultant de l'exécution d'un processus comme un groupe de ressources, à leur tour utilisables pour en produire d'autres. Cela est représenté à l'aide de la relation *is_composedOf* entre *WorkProduct* et *Resource* (Figure 5).

4.5 Agent

Le diagramme de classe de la Figure 7 est inspiré des travaux de (Ruíz *et al.*, 2004). Un agent peut être une ressource humaine ou logicielle. Pour une activité donnée, il est possible d'affecter un agent particulier (affectation directe), ou de spécifier son rôle (affectation indirecte).

Par ailleurs, les deux relations "subordinated" et "superior" liées au concept *Job* représentent le modèle organisationnel de l'entreprise, qui n'est pas décrit dans BPMN. La représentation de ce modèle est intéressante pour identifier les responsabilités hiérarchiques, par exemple lors d'un Workflow de validation ou de l'envoi de notifications en cas d'anomalie durant le déroulement d'un processus.

Nous avons différencié le concept *Job* du concept *Role* pour offrir plus de flexibilité. En effet, deux personnes ayant le même poste (Job) n'ont pas systématiquement les mêmes autorisations pour exécuter des activités.

5 Formalisation et implémentation en OWL

La formalisation de BBO s'est faite en deux temps : nous avons tout d'abord appliqué un ensemble de règles pour traduire les diagrammes de classes UML créés à la phase précédente en des concepts et relations OWL ; puis nous avons transcrit un ensemble de spécifications énoncées en langage naturel en OWL.

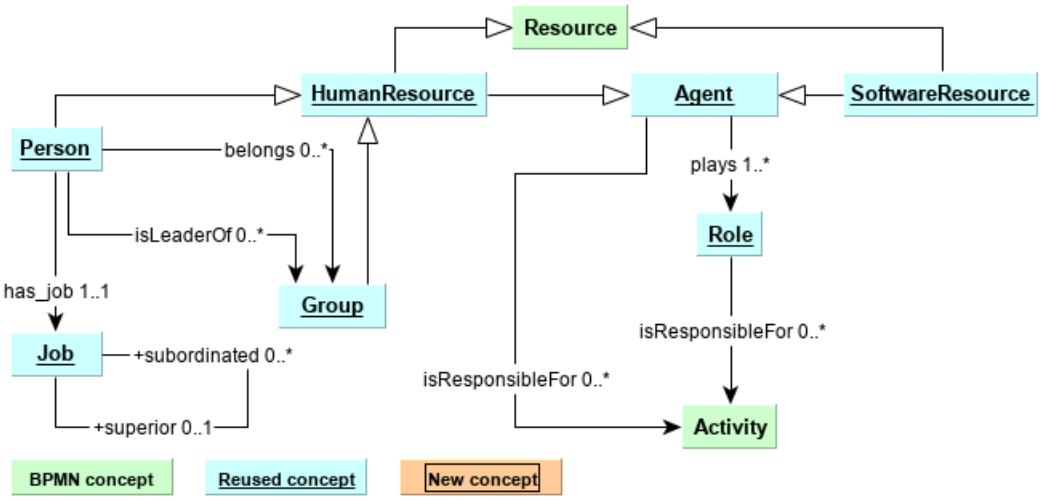

FIGURE 7 – *Diagramme UML du concept Agent*

5.1 Traduction des diagrammes UML en OWL

Nous avons appliqué les règles suivantes :

1. Chaque classe UML donne lieu à la création d'une classe OWL
2. Chaque relation entre classes UML est représentée par une propriété OWL de type ObjectProperty dont le domaine et le co-domaine sont les classes OWL correspondant à ces classes UML.
3. Pour chaque attribut de classe UML :
   — Si le type de cet attribut est une autre classe UML, une propriété OWL de type ObjectProperty est créée, avec pour domaine la classe à laquelle appartient l'attribut et pour co-domaine la classe associée au type de l'attribut.
   — Sinon, une propriété OWL de type DatatypeProperty est créée, liant la classe OWL créée à partir de la classe UML à laquelle est associé l'attribut au type de donnée XML spécifié dans le diagramme UML.
4. Les cardinalités sont traduites en restrictions de cardinalité sur les propriétés. Soient UClass1 et UClass2 deux classes UML, OClass1 et OClass2 les deux classes OWL correspondantes, UProperty une relation UML ou un attribut, et OProperty la propriété OWL correspondante. Soit n un entier et x un entier supérieur à 0.
   — UClass1 UProperty [x..x] UClass2 => OClass1 (subClassOf OProperty **exactly** x) OClass2
   — UClass1 UProperty [x..n] UClass2 => OClass1 (subClassOf OProperty **min** x) OClass2
   — UClass1 UProperty [0..x] UClass2 => OClass1 (subClassOf OProperty **max** x) OClass2

5.2 Traduction en OWL de spécifications exprimées en LN

L'équipe OMG, qui a développé le La définition du modèle BPMN par l'équipe OMG comporte, en plus du diagramme UML, un ensemble de spécifications rédigées en langage naturel dont une partie n'est pas prise en charge par le diagramme UML. Or, pour des questions de cohérence et de validation de la représentation des processus, il est important de formaliser le sous-ensemble de ces spécifications qui concernent les concepts de notre ontologie. Nous donnons quelques exemples dans la Table 4 (les formalisations issues des spécifications en langage naturel sont notées en caractères gras).

La formalisation de ces spécifications consiste à exprimer des restrictions sur des propriétés existantes (voir les 2 premiers exemples de la Table 4), ou encore à spécialiser des concepts existants (voir les 2 derniers exemples de la Table 4 où les concepts *ConvergingGateway* et *ConditionalSequenceFlow* sont donc de nouveaux sous-concepts de *Gateway* et

Spécification	**Formalisation du texte**
A Start Event **MUST** be a source for a Sequence Flow (p.245)	StartEvent *subClassOf* (CatchEvent *and* ... ***and*** **has_outgoing *some* SequenceFlow**)
The list of BPMN elements that **MUST NOT** be used in an Ad-Hoc Sub-Process : Start Event, End Event (p.182)	**AdHocSubProcess *SubClassOf not* (has_flowElements *some* (StartEvent or EndEvent))**
A Gateway with a gatewayDirection of converging MUST have multiple incoming Sequence Flows, but **MUST NOT** have multiple outgoing Sequence Flows. (p. 290)	**ConvergingGateway *equivalentTo*** **(Gateway** **and (has_incoming *min* 2 SequenceFlow)** ***and*** **(has_outgoing *exactly* 1 SequenceFlow))**
A Timer Event is an Event that has exactly one TimerEventDefinition. (p. 274)	**TimerEvent *equivalentTo*** **(Event** ***and*** **(has_eventDefinition *exactly* 1 TimerEventDefinition))**
An Intermediate Event **MUST** be a source for a Sequence Flow. (p. 259)	IntermediateEvent *equivalentTo* (IntermediateCatchEvent *or* IntermediateThrowEvent) **IntermediateEvent *subClassOf* (has_outgoing *some* SequenceFlow)**

TABLE *4 – Formalisation de spécifications de BPMN*

de *SequenceFlow* respectivement). Cette formalisation permet une classification automatique des instances et une vérification du respect des spécifications. BBO a été implémentée en OWL à l'aide de l'outil d'édition d'ontologie Protégé.

5.3 Documentation

Enfin, nous avons documenté BBO en ajoutant la description de chaque concept et de chaque relation à l'aide de la propriété *rdfs :comment*. Pour les concepts et relations issus des modèles réutilisés, les descriptions originales ont été respectées.

6 Évaluation et Discussion

Nous avons vu en Section 4 que le modèle sans doute le plus abouti pour représenter un processus métier était le modèle BPMN, mais nous avons aussi souligné ses limites. Ne disposant donc pas d'ontologie de référence pour évaluer BBO, nous proposons d'estimer sa qualité selon des critères quantitatifs et qualitatifs.

6.1 Évaluation quantitative

Afin d'évaluer le modèle conceptuel de BBO, nous avons calculé deux métriques proposées par (Tartir & Arpinar, 2007). Soient NH le nombre de relations d'hyperonymie (is-a), NR le nombre de relations autres que l'hyperonymie, et NC le nombre de concepts. Les indicateurs sont ainsi calculés :

— RD = NR/(NR + NH) RD est compris entre 0 et 1, et il indique le taux de diversité des relations. Plus RD est proche de 1, plus les relations sont diverses au sein du modèle.
— SD = NH / NC indique la profondeur de l'ontologie et correspond au nombre moyen de relations d'hyperonymie entre chaque concept et la racine. Un SD faible traduit le fait que l'ontologie est "profonde" et détaille les connaissances d'un domaine spécifique.

La Table 5 donne le nombre actuel de concepts et de relations dans le modèle conceptuel de BBO, ainsi que les valeurs des deux métriques RD et SD.

#Concepts	#Relations autres que is-a	#Relations is-a	Métriques
158	145	135	RD = 145/(145+135)= 0.52 SD = 135/158= 0.85

TABLE *5 – Métriques du schéma BBO*

La valeur de RD est 0.52, ce qui montre que BBO n'est pas qu'une simple taxonomie, mais une ontologie riche en relations. De plus, BBO a un SD faible inférieur à 1, ce qui témoigne de sa profondeur et d'une bonne couverture du domaine.

6.2 Évaluation qualitative

L'évaluation qualitative a été menée de façon empirique pour vérifier que BBO est bien (i) consistante ; (ii) expressive, i.e. capable de représenter différents modèles de processus métiers ; et (iii) capable de répondre aux questions portant sur les compétences techniques.

6.2.1 Consistance

Nous avons vérifié la consistance de BBO en utilisant trois raisonneurs HermiT, Fact et Pellet au sein de Protégé, et cela avant et après son peuplement.

6.2.2 Expressivité

Nous avons peuplé BBO avec deux processus métier, choisis aléatoirement, et provenant des deux partenaires industriels, TAS et Continental. Pour des raisons de confidentialité, nous n'allons présenter que des extraits anonymes des processus originaux.

Exemple 1. : Ce premier exemple concerne l'extrait du processus TAS présenté en Table 1. Pour plus de lisibilité, nous représentons l'exemple avec des éléments graphiques du langage graphique de BPMN, et nous affectons un identifiant (en rouge) à chaque élément (Figure 8).

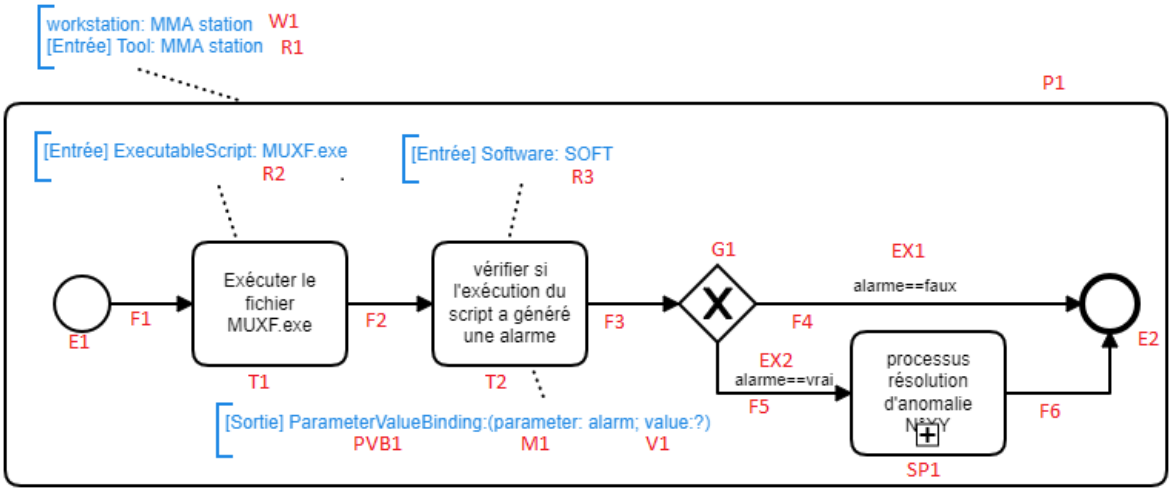

FIGURE *8 – Représentation graphique de l'extrait présenté dans la Table 1*

Le premier cercle est un *StartEvent*, point de démarrage du processus. Les deux premiers rectangles représentent les deux premières tâches. Au terme de la deuxième tâche, deux cas

sont possibles selon la valeur de la variable *alarme*, d'où l'utilisation de la passerelle (Gateway) représentée par un losange. Si la condition (alarme==vrai) est vraie, le sous processus (rectangle avec le symbole "+") doit être exécuté avant de terminer le processus. Le dernier cercle avec une bordure épaisse est un *EndEvent*, qui marque de la fin du processus.

Toutes les instances d'une classe OWL sont de type `owl:NamedIndividual`. La Table 6 donne les triplets qui lient les instances aux classes de BBO, en utilisant les identifiants de la Figure 8. Les assertions sont écrites en Turtle avec un espace de nommage de base correspondant à l'URI de BBO (pour simplifier, nous avons omis le préfixe BBO e.g., `:Process` au lieu de `BBO:Process`). La tâche T2 a une spécification de sortie qui est la valeur V1 du paramètre M1 (paramètre alarme), représenté en créant une instance PVB1 de *ParameterValueBinding* qui relie le paramètre à sa valeur qui va être renseignée durant l'exécution.

La table 7 montre les assertions correspondant aux instances des relations de type ObjectProperty. D'autres assertions correspondant aux relations de DataProperty seraient nécessaires pour une représentation complète, mais nous les omettons pour des raisons de lisibilité.

:P1 a :Process.	:R3 a :Software .
:E1 a :StartEvent .	:M1 a :Parameter .
:Fi a :SequenceFlow . #(i=1,..,6)	:V1 a :ParameterValue .
:Ti a :Task #(i=1,2)	:PVB1 a :ParameterValueBinding .
:G1 a :Gateway .	:W1 a :Workstation .
:IOi a :InputOutputSpecification #(i=1,...,3)	:EXi a :ConditionalExpression #(i=1,2)
:R1 a :Tool .	:SP1 a :SubProcess .
:R2 a :ExecutableScript .	:E2 a :EndEvent .

TABLE *6 – Triplets définissant les instances des classes OWL*

:P1 takesPlaceAt :W1 .	:EX1 :has_valueToBeEvaluated :V1.
:P1 :has_flowElements :E1.	:EX2 :has_valueToBeEvaluated :V1.
:P1 :has_flowElements :F1, :F2, :F3, :F4, :F5, :F6.	:F1 :has_sourceRef :E1.
:P1 :has_flowElements :Ti i=1,2 .	:F1 :has_targetRef :T1.
:P1 :has_flowElements :G1 .	:F2 :has_sourceRef :T1.
:P1 :has_ioSpecification :IO3 .	:F2 :has_targetRef :T2.
:IO3 :has_resourceInput :R1 .	:F3 :has_sourceRef :T2.
:P1 :has_flowElements :SP1 .	:F3 :has_targetRef :G1.
:P1 :has_flowElements :E2 .	:F4 :has_sourceRef :G1.
:IO1 :has_resourceInput :R2 .	:F4 :has_targetRef :E2.
:T1 :has_ioSpecification :IO1 .	:F4 :has_conditionExpression :EX1.
:IO2 :has_resourceInput :R3 .	:F5 :has_sourceRef :G1.
:IO2 :has_parameterValueBindingOutput :PVB1 .	:F5 :has_targetRef :SP1.
:PVB1 :has_parameter :M1 .	:F5 :has_conditionExpression :EX2.
:PVB1 :has_parameterValue :V1 .	:F6 :has_sourceRef :SP1.
:T2 :has_ioSpecification :IO2 .	:F6 :has_targetRef :E2.

TABLE *7 – Axiomes correspondant aux propriétés OWL*

Exemple 2. : La Table 8 présente un autre exemple fourni par Continental. L'instruction 45 précise que les tâches 41 à 44 doivent être répétées 9 fois. BBO permet la représentation de cette spécification. En effet, il suffit de créer une instance SLC de *StandardLoopCharacteristics* qui est une sous-classe de *LoopCharacteristics* (Figure 2). *StandardLoopCharacteristics* a un attribut, appelé *loopMaximum*, qui permet de spécifier le nombre max d'itérations. Pour cet exemple, on affecte la valeur 9 à l'attribut *loopMaximum* de SLC. Ensuite, il faut créer une instance de *SubProcess* (SP) qui inclut les tâches 41 à 44, et la lier à SLC par la propriété *has_loopCharacteristics* (Table 9).

```
43. ...................
44. ...................
45. Répéter les opérations 41 à 44, 9 fois en suivant
les instructions sur l'écran du poste d'emballage
```

TABLE *8 – Extrait anonymisé d'un processus métier Continental*

:SP a :SubProcess.	:SLC :loopMaximum 9.
:SP :has_flowElements :T41, :T42, :T43, :T44.	:SP :has_loopCharacteristics :SLC.
:SLC a :StandardLoopCharacteristics.	

TABLE *9 – Les assertions de l'exemple 2*

Exemple 3. Dans cet exemple, nous mettons l'accent sur la quatrième instruction de l'exemple présenté Table 10. L'opérateur qui l'exécute est censé saisir la valeur de la télémesure TM_MM et vérifier si sa valeur est comprise entre -0.8V et +0.8V. La représentation de cette instruction fait appel aux concepts et relations que nous avons rajoutés pour représenter les paramètres d'entrée/sortie d'une activité (Figure 4). En effet, la télémesure qu'envoie le satellite est un paramètre qui prend différentes valeurs selon la télécommande envoyée.

Comme indiqué dans la Table 11, la télémesure TM_MM est représentée comme une instance de la classe *QuantitativeParameter*. Pour spécifier l'unité de mesure, nous rajoutons une instance de la classe *UnitOfMeasure* qui référence via la proprité *has_unit* une instance de la classe *UO_0000218* (la classe des Volts de l'ontologie des unités de mesure UO, importée dans BBO). Puis, nous créons EV, une instance de *ParameterExpectedValue* dont les valeurs min et max sont spécifiées par les attributs *minValue* et *maxValue*, à savoir -0.8 et +0.8. Nous créons aussi une instance de *ParameterValue* (PV) pour spécifier que T4 va produire une valeur durant l'exécution du processus. PVB est une instance de *ParameterValueBinding* qui lie le paramètre à ses valeurs, à l'aide des propriété *has_parameter*, *has_parameterValue* et *has_parameterExpectedValue*. Enfin, nous rajoutons une instance de *InputOutputSpecification* qui fera la liaison entre T4 et PVB grâce aux priopriétés *has_ioSpecification* et *has_parameterVaueBindingOutput*.

```
1. ...................
2. Envoyez la télécommande CGHFR.
3. Attendre 45 secondes.
4. Vérifier et noter la valeur de la télémesure
suivante TM_MM (-0.8V,+0.8V)
```

TABLE *10 – Extrait anonymisé d'un processus métier TAS*

Ainsi, nous avons pu vérifier que BBO permet de représenter les connaissances contenues dans les deux types de processus métier traités.

6.2.3 Questions basées sur les compétences techniques

Nous avons pu exprimer toutes les "competency questions" ayant servi à spécifier le modèle (Section 3.2) en SPARQL, et avons exécuté ces requêtes sur BBO peuplée, dans l'environnement Protégé (Table 12).

Pour certaines questions, il n'était pas possible de vérifier les résultats des requêtes : (i) soit parce qu'il n'y a pas d'instance dans la base de connaissances qui réponde à la requête ; c'est le cas de la question 7 car les fiches décrivant les processus métiers ne mentionnent pas les agents ou les rôles responsables de l'exécution des activités ; (ii) soit parce que la

:T4 a :Task.	:EV :maxValue +0.8.
:TM_MM a :QuantitativeParameter.	:PV a :ParameterValue.
:UM a :UnitOfMeasure.	:PVB a :ParameterValueBinding.
:Volt a UO :UO_0000218.	:PVB :has_parameter :TM_MM.
:UM has_unit :Volt.	:PVB :has_parameterValue :PV.
:TM_MM has_unitOfMeasure :UM.	:PVB :has_parameterExpectedValue :EV.
:IO a :InputOutputSpecification.	:IO :has_parameterValueBindingOutput :PVB.
:EV a :ParameterExpectedValue.	:T4 :has_ioSpecification :IO.
:EV :minValue -0.8.	

TABLE *11 – Les assertions de l'exemple 3*

réponse relève de l'aspect dynamique de ces processus métier ; la question 5 par exemple cherche l'activité qui suit une activité donnée. Cela dépend parfois de l'évaluation de certains paramètres durant l'exécution, tel que le paramètre "alarme" dans la Figure 8.

6.2.4 Synthèse

Cette évaluation a montré que BBO est riche en relations autres que "is-a", et que son niveau de profondeur offre une bonne couverture du domaine. BBO permet ainsi de représenter les processus métiers avec une granularité fine, comme nous l'avons observé à travers le peuplement avec les deux processus. Enfin, on était capable de formaliser toutes les competency questions à l'aide de BBO. Sur la base de cette évaluation empirique, nous estimons que BBO présente les qualités requises pour répondre aux besoins de l'agent virtuel en termes de description des processus métier. BBO est accessible à partir de l'URL `https://www.irit.fr/recherches/MELODI/ontologies/BBO`.

7 Conclusion et Perspectives

La modélisation des processus métier est un domaine actif de recherche qui attire plus d'attention avec l'émergence de l'industrie 4.0. Dans cet article, nous avons proposé une nouvelle ontologie, appelée BBO, pour représenter les processus métier. Pour construire cette ontologie, nous avons réutilisé des fragments d'ontologies et de modèles existants, notamment BPMN 2.0. L'évaluation que nous avons menée a montré que BBO (i) n'est pas qu'une simple taxonomie, (ii) n'a pas une structure plate, et peut donc décrire avec finesse les connaissances du domaine, (iii) est une ontologie consistante, (iv) est capable de représenter formellement les "competency questions" issues des besoins exprimés par les experts et de la littérature. BBO a été implémentée en OWL et mise à disposition de la communauté scientifique.

Sur la base de cette évaluation, BBO permet de représenter les informations nécessaires à l'exécution des processus métiers. Par contre, BBO n'assure pas encore la représentation de l'historique des exécutions. Ce point est la prochaine étape de notre travail. Par ailleurs, peupler l'ontologie manuellement est une besogne longue et fastidieuse, et sujette aux erreurs. Nous projetons de la réaliser automatiquement ou semi-automatiquement, par un traitement automatique du texte présent dans les documents techniques décrivant les processus métiers.

8 Remerciements

Le projet AVI-REX est financé par la Région Occitanie, le FEDER-FSE Midi-Pyrénées et le programme Garonne 2014-2020 sous le label 2017-AVIREX-IRIT-READYNOV INDUSTRIE DU FUTUR. Les auteurs remercient leurs partenaires industriels, SIMSOFT INDUSTRY, Thales Alenia Space et Continental.

N°	Question en langage naturel	Requête en SPARQL
1	`Quelles ressources sont requises pour une activité ?` -Plusieurs versions de cette question peuvent être posées selon la nature de la ressource (ex : les outils nécessaires, les composants nécessaires, etc.)	Select ?A ?R where { ?A a BBO :Activity. ?R a BBO :Resource. ?A BBO :has_ioSpecification ?io. ?io BBO :has_resourceInputs ?R}
2	`Quelles sont les activités qui composent un processus ?` - Cette question peut aussi être posée pour un sous-processus.	Select ?P ?A where { ?P a BBO :Process. ?A a BBO :Activity. ?P BBO :has_flowElements ?A}
3	`Quelles activités doivent être achevées avant de commencer une activité donnée ?` - Cette question traite la dépendance entre les activités. Une activité A dépend d'une activité B, si B produit une sortie (ressource ou valeur de paramètre) qui est l'entrée de A. La requête suivante renvoie tous les couples d'activités (A,B) tels que A dépend de B.	Select Distinct ?A ?B where {{ ?B BBO :has_ioSpecification ?IOb. ?IOb BBO :has_resourceOutputs ?R. ?A BBO :has_ioSpecification ?IOa. ?IOa BBO :has_resourceInputs ?R.} UNION { ?B BBO :has_ioSpecification ?IOb. ?IOb BBO :has_parameterValueBindingOutputs ?PVB. ?A BBO :has_ioSpecification ?IOa. ?IOa BBO :has_parameterValueBindingInputs ?PVB.}}
4	`Quel est le site d'exécution d'une activité ?`	Select ?A ?S where { ?A a BBO :Activity. ?S a BBO :ManufacturingFacility. ?A BBO :takesPlaceAt ?S.}
5	`Quelle est l'activité qui suit une activité ?` -La réponse à cette requête n'est pas toujours simple. Par exemple, dans la Figure 8, l'activité qui suit T2 ne peut être connue qu'au moment de l'exécution car elle dépend de la valeur de « alarme » que l'opérateur doit saisir. Dans le cas de T2, la requête va donner un ensemble vide, mais il faut chercher la nature de l'élément suivant Event, Gateway ou Activity pour continuer l'exécution du processus.	Select ?A ?nextA where { ?A a BBO :Activity. ?nextA a BBO :Activity. ?A BBO :has_outgoing ?SF. ?SF BBO :has_targetRef ?nextA.}
6	`Quel est le type de chaque ressource ?` - La taxonomie des ressources de BBO permet de donner un type précis à chaque ressource. Cette taxonomie peut être spécialisée pour chaque entreprise afin de prendre en compte ses ressources. Ainsi, on a ajouté `Switch` comme sous-classe de `Component` pour TAS et `LecteurCodeBar` comme sous-classe de `Device` pour Continental.	Select Distinct ?R ?typeR where { ?R a BBO :Resource. ?R a ?typeR.}
7	`Qui peut exécuter chacune des activités ?` -l'affectation des agents aux différentes activités peut se faire directement ou par rôle, d'où l'union des deux requêtes.	Select ?A ?agent where { { ?agent a BBO :Agent. ?A a BBO :Activity. ?agent BBO :isResponsibleFor ?A.} UNION { ?agent a BBO :Agent. ?A a BBO :Activity. ?role a BBO :Role. ?agent BBO :has_role ?role. ?role BBO :isResponsibleFor ?A.}}
8	`Quels sont les composants de chaque produit ?` -il est possible de répondre à cette question grace à la relation isComposedof entre WorkProduct et Resource	Select ?w ?r where { ?w a BBO :WorkProduct. ?r a BBO :Resource. ?w BBO :isComposedOf ?r}
9	`Y-a-t-il un manuel d'utilisateur pour un outil donné ?` -la requête renvoie la liste de tous les outils qui ont un manuel d'utilisateur, il suffit de filtrer par le code ou le nom de l'outil spécifié pour avoir la réponse.	Select ?T, ?UM where { ?T a BBO :Tool. ?UM a BBO :UserManual. ?T BBO :has_technicalDocument ?UM}
10	`What is the unit measure of a given parameter?` - l'unité de mesure d'un paramètre se compose d'une partie principale (unité) et eventuellement un prefix (mili, centi, etc.), d'où le mot clé optionnal pour le prefix.	SELECT ?P ?unit ?prefix WHERE { ?P a BBO :QuantitativeParameter. ?P BBO :has_unitOfMeasure ?unit. optional { ?P BBO :has_prefix ?prefix} }

TABLE *12 – Exemples de questions basées sur les compétences techniques en SPARQL.*

Références

ABDALLA A., HU Y., CARRAL D., LI N. & JANOWICZ K. (2014). An Ontology Design Pattern for Activity Reasoning. In *5th Workshop on Ontology and Semantic Web Patterns (WOP2014)*, p. 78–81, Riva del Garda, Italy.

AWAD A., GROSSKOPF A., MEYER A. & WESKE M. (2009). *Enabling resource assignment constraints in BPMN*. Rapport interne, Hasso Plattner Institute.

BERNERS-LEE T., HENDLER J. & LASSILA O. (2001). The Semantic Web. *Scientific American*, **284**(5), 28–37.

BOCCIARELLI P., D'AMBROGIO A., GIGLIO A. & PAGLIA E. (2016). A BPMN extension to enable the explicit modeling of task resources. In *CEUR Workshop Proceedings*, volume 1728, p. 40–47.

CABRAL L., NORTON B. & DOMINGUE J. (2009). The business process modelling ontology. In *4th international workshop on semantic business process management*, p. 9–16.

CHUNGOORA N., YOUNG R. I. M., GUNENDRAN G., PALMER C., USMAN Z., ANJUM N. A., CUTTING-DECELLE A.-F., HARDING J. A. & CASE K. (2013). A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry*, **64**(4), 392–401.

FALBO R. D. A. & BERTOLLO G. (2009). A software process ontology as a common vocabulary about software processes. *International Journal of Business Process Integration and Management*, **4**(4), 239–250.

FERNÁNDEZ M., GÓMEZ-PÉREZ A. & JURISTO N. (1997). METHONTOLOGY : From Ontological Art Towards Ontological Engineering. In *Symposium on Ontological Engineering of American Asociation for Artificial Intelligence (AAAI)*, p. 33–40.

FRAGA A., VEGETTI M. & LEONE H. (2018). Semantic interoperability among industrial product data standards using an ontology network. In *20th International Conference on Enterprise Information Systems (ICEIS)*, volume 2, p. 328–335, Madeira, Portugal.

GRÜNINGER M. & FOX M. S. (1995). The Role of Competency Questions in Enterprise Engineering. In *Benchmarking—Theory and practice*, p. 22–31. Springer, Boston, MA.

KARRAY M. H., CHEBEL-MORELLO B. & ZERHOUNI N. (2012). A formal ontology for industrial maintenance. *Applied ontology*, **7**(3), 269–310.

MARCINKOWSKI B. & KUCIAPSKI M. (2012). A business process modeling notation extension for risk handling. In *11th International Conference on Computer Information Systems and Industrial Management (CISIM)*, volume 7564 LNCS, p. 374–381, Venice, Italy.

NATSCHLÄGER C. (2011). Towards a BPMN 2.0 ontology. In *3rd International Workshop on Business Process Modeling Notation*, p. 1–15, Lucerne, Switzerland.

OMG (2011). *Business Process Modeling Notation, v2.0 - Specification*. Rapport interne, Object Management Group.

PEDRINACI C., DOMINGUE J. & DE MEDEIROS A. K. A. (2008). A core ontology for business process analysis. In *5th European Semantic Web Conference, ESWC, Tenerife, Canary Islands, Spain*, p. 49–64.

ROSPOCHER M., GHIDINI C. & SERAFINI L. (2014). An ontology for the Business Process Modelling Notation. In *8th International Conference on Formal Ontology in Information Systems (FOIS)*, p. 133–146, Rio de Janeiro, Brazil.

ROY S., DAYAN G. S. & DEVARAJA HOLLA V. (2018). Modeling industrial business processes for querying and retrieving using OWL+SWRL. In *On the Move to Meaningful Internet Systems (OTM)*, p. 516–536, Valletta, Malta.

RUÍZ F., VIZCAINO A., PIATTINI M. & GARCÍA F. (2004). An Ontology For The Management Of Software Maintenance Projects. *International Journal of Software Engineering and Knowledge Engineering*, **14**(3), 1–27.

STROPPI L. J. R., CHIOTTI O. & VILLARREAL P. D. (2011). A BPMN 2 . 0 Extension to Define the Resource Perspective of Business Process Models. In *14th Iberoamerican Conference on Software Engineering (CIbSE)*, p. 25–38, Rio de Janeiro, Brasil.

TARTIR S. & ARPINAR I. B. (2007). Ontology evaluation and ranking using OntoQA. In *1st International Conference on Semantic Computing (ICSC)*, p. 185–192, California, USA.

USCHOLD M., KING M., MORALEE S. & ZORGIOS Y. (1998). The enterprise ontology. *The knowledge engineering review*, **13**(1), 31–89.

VOGEL-HEUSER B. & HESS D. (2016). Guest Editorial Industry 4.0–Prerequisites and Visions. *IEEE Transactions on Automation Science and Engineering*, **13**(2), 411–413.