



**UNIVERSITÀ DEGLI STUDI DI TRENTO**

**Facoltà di Scienze Matematiche,  
Fisiche e Naturali**

---

**UNIVERSITA' DEGLI STUDI DI TRENTO**

**Master in Technologies for System Integration & e-Government**

**An ontology-based comparative study of standards for  
business process engineering**

**Submitted by  
Abdul Gafar Manuel Meque**

**University Advisor: Roberta Ferrario  
Company Advisor: Claudio Masolo**

**Company: Laboratory for Applied Ontology**

Academic year 2011 / 2012



## Abstract

Since before the beginning of the millennium, many companies started to look for ways to optimize their business processes in order to improve productivity and/or improve efficiency and increase their results, towards that goal, along with the companies many scholars joined the in this new challenging field, proposing new ideas on how to plan the business processes, they came up with several process modelling languages, some of them proposing ideas based on well-known techniques and methodologies while the others where proposing new and invigorating ideas. In this way many Business Process Modelling Languages were created, ranging from general purpose Business Process Modelling to some more field specific Process Modelling Languages.

With the increase of computer-based agents and the need for automation of processes allied with the need for integration and interoperability among systems and processes as well, starting from internal interoperability to a more external interoperability which goes beyond simple exchange of information, calls for new ways of process modelling. In order to account for all the three aspects of interoperability, Technical Interoperability, Policy Interoperability and Semantic Interoperability, and not only the former which is what usually happens.

For the current work, I joined the Laboratory for Applied ontology as an intern, conducting research on the BPMLs and methodologies, reviewing the works done by Aagesen and Krogstie in (Aagesen, et al., 2010), and Recker and others (Recker, et al., 2008), (Recker, et al., 2005), Recker and Rosemann (Recker, et al., 2007) and Gruninger in (Gruninger, 2009), and also conducting my own studies, evaluating the languages premises with respect to DOLCE (Descriptive Ontology for Linguistics and Cognitive Engineering) (Masolo, et al. 2003), developed at the LOA (Laboratory for Applied Ontology), checking for the ontological choices ranging from the objectives behind the language to the intuitions behind the language constructs, good element categorization, and also the axiomatization of the relationships.

In an initial phase following languages were targeted BPMN (Business Process Modeling and Notation), BPEL (Business Process Execution Language), XPDL (XML Business Process Definition Language), PSL (Process Specification Language) and YAWL(Yet Another Workflow Language) but, during the course of work, three of them (XPDL, YAWL and BPEL) were dropped for various reasons.

As the result of the study, BPMN proved to have many ontological deficiencies and to be inappropriate for automated interoperable processes, while PSL proved to be very useful for process exchange and excellent for interoperable processes due to its very powerful ontology but at the same time it proven to be inappropriate for process modeling, for it is design to specify one time instances.

## **Dedication**

I dedicate this research work to my family and friends who supported during this tough journey.

## **Acknowledgements**

First of all, I would like to say thanks for your interest in my work. The present thesis represents the result of my graduation internship work, carried out from September 2011 to May 201 in Trento, Italy at the Laboratory for Applied Ontology, as a partial fulfilment of the requirements for a master degree in Technologies for System Integration and e-Government at University of Trento.

Secondly, I would like to thank all my supervisors, Roberta Ferrario and Claudio Masolo, who provided me with guidance throughout the research.

My sincere appreciation goes to the Laboratory for Applied Ontology, for providing me a very pleasant work environment and all the great colleagues I have had the opportunity to meet and who were more than willing to help me, in particular Nicolas Troquard, Emanuele Bottazzi and Alessander Benevides.

Kind regards.

# Table of Contents

<b>ABSTRACT.....</b>	<b>III</b>
<b>DEDICATION .....</b>	<b>IV</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>V</b>
<b>TABLE OF CONTENTS .....</b>	<b>VI</b>
<b>LIST OF FIGURES.....</b>	<b>VIII</b>
<b>LIST OF TABLE .....</b>	<b>IX</b>
<b>1. INTRODUCTION.....</b>	<b>1</b>
<b>2. ONTOLOGY-BASED ANALYSES .....</b>	<b>2</b>
<b>2.1. ONTOLOGICAL ANALYSIS BASED ON BWW (BUNGE-WAND-WEBER ONTOLOGY) .....</b>	<b>2</b>
<b>2.2. ONTOLOGICAL ANALYSIS BASED ON DOLCE.....</b>	<b>3</b>
2.2.1. ONTOLOGICAL CHOICES ANALYSIS.....	3
2.2.2. ELEMENTS CATEGORIZATION.....	4
2.2.3. BASIC RELATIONS AND FUNCTIONS .....	5
<b>3. BUSINESS PROCESS MODELING LANGUAGES .....</b>	<b>7</b>
<b>3.1. BUSINESS PROCESS MODELING AND NOTATION .....</b>	<b>7</b>
3.1.1. LANGUAGE CONSTRUCTS AND PROPERTIES.....	7
3.1.1.1. Flow Object.....	7
Event.....	8
Activities .....	12
Gateways.....	15
3.1.1.2. Connecting Objects .....	16
3.1.1.3. Data .....	18
Data Objects, Data Input and Data Output .....	18
Data Stores .....	19
3.1.1.4. Artifacts .....	19
Text Annotations .....	19
Groups .....	20
3.1.1.5. Swimlanes.....	20

3.1.2.	INFORMAL CONSIDERATIONS ON BPMN PRIMITIVES.....	21
3.1.2.1.	General Considerations .....	21
3.1.2.2.	A look into the primitives: Activity .....	22
3.1.2.3.	A look into the primitives: Events .....	22
3.1.2.4.	A look into the primitives: Connecting Objects.....	23
3.1.2.5.	A look into the primitives: Swimlanes.....	23
3.1.1.6.	A look into the primitives: Artifact .....	24
3.1.2.	ONTOLOGICAL ANALYSIS BASED ON BWV ONTOLOGY .....	25
3.1.3.	ONTOLOGICAL ANALYSIS BASED ON DOLCE.....	27
3.1.3.1.	Ontological Choices Analysis .....	27
	<i>Cognitive Bias</i> .....	27
	<i>Knowledge Type</i> .....	27
	<i>Expressiveness Analysis</i> .....	28
	<i>Results</i> .....	28
3.1.3.2.	Elements Categorization .....	28
3.1.3.3.	Primitive Relations and Functions Analysis .....	31
<b>3.2.</b>	<b>PROCESS SPECIFICATION LANGUAGE .....</b>	<b>33</b>
3.2.1.	THE LANGUAGE STRUCTURE (THE ONTOLOGY) .....	33
3.2.1.1.	Language .....	33
	Lexicon.....	33
	Grammar.....	34
3.2.1.2.	Model theory.....	34
3.2.1.3.	Proof Theory.....	34
	PSL Core.....	34
	Foundational Theories.....	35
	Extensions.....	35
3.2.1.4.	Semantics of PSL Core .....	35
	Primitive Lexicon .....	36
	Defined Lexicon .....	36
	Axioms .....	37
3.2.2.	INFORMAL ANALYSIS OF PSL PRIMITIVES.....	41
3.2.3.	ANALYSES BASED ON DOLCE.....	45
3.2.3.1.	Ontological Choices Analysis .....	45
	<i>Cognitive Bias</i> .....	45
	<i>Knowledge Type</i> .....	45
	<i>Expressiveness Analysis</i> .....	46
	<i>Results</i> .....	46
3.2.3.2.	Elements Categorization .....	46
3.2.3.3.	Primitive Relations and Functions Analysis .....	47
<b>4.</b>	<b>CONCLUSIONS.....</b>	<b>48</b>
4.1.	CONTRIBUTIONS .....	48
4.2.	SUGGESTED IMPROVEMENT TO THE PRESENT WORK.....	48

4.3. OVERALL CONCLUSIONS .....	48
4.4. FUTURE WORK.....	49
<b><u>BIBLIOGRAPHY.....</u></b>	<b>51</b>

## List of Figures

Figure 1: Taxonomy of DOLCE basic categories. taken from (Masolo, et al., 2003) .....	5
Figure 2: BPMN Event Types from Wikipedia.....	8
Figure 3: Interrupting (Left) non-Interrupting (right) event sub-process.....	9
Figure 4: BPMN Flow Object – Events .....	12
Figure 5: BPMN Flow Object - Activity: Tasks.....	13
Figure 6: BPMN Ad-Hoc Process.....	14
Figure 7: BPMN Flow Object – Gateways .....	16
Figure 8: Data Association: (OMG BPMN Team, 2010) pg 229 .....	17
Figure 9: BPMN Connecting Objects .....	18
Figure 10: BPMN Data .....	19
Figure 11: Data Store .....	19
Figure 12: BPMBN Artifacts - Text Annotation .....	20
Figure 13: BPMN Artifact - Group.....	20
Figure 14: BPMN Swimlanes - A pool with 2 lanes.....	21
Figure 15: A Pool with three Lanes.....	24
Figure 16: Order Processing form BPMN 2.0 by Examples v 1 .....	30
Figure 17: Activity and Activity-Occurrence Example by NIST .....	42



## List of Table

Table 1: Types of BPMN Tasks.....	14
Table 2: BPMN Sub-Process Types .....	15
Table 3: BPMN to BWW mapping by Recker et al. 2005.....	26
Table 4: DOLCE to BPMN occurrence Mappings .....	29
Table 5: DOLCE Endurants to BPMN Mapping .....	31

# 1. Introduction

Over the past years, the need for a Modeling language that supports Business Processes has increased, many researchers and companies have devoted time and resources in the research and development of a BP modeling language that would be more or less used as standard. This thesis will focus on some of the more successful languages developed and in use so far; to better understand them and provide a good ontological view on them, we need first to have a clear understand of are languages, business, process, modeling and business process as a starting point.

According to the Business Dictionary<sup>1</sup>, “business process is a series of logically related activities or tasks (such as planning, production, or sales) performed together to produce a defined set of results”. In order to improve them in the future, current processes need to be carefully analyzed, and to do that, it is necessary to somehow record them. They can’t just be recorded in an arbitrary manner, they should be recorded in a way that it is understandable by the stakeholder of that process, as well to guarantee that any business analyst or managers with a minimum effort can understand them. The activity of representing the business process in a certain pre-defined manner is called modeling; the result of that activity is a Business Process Modeling.

Since one of the goals of having a Business Process (from now on BP) stored is to improve in the future, and it needn’t be the same person doing that, it creates a need to provide some standardized way of representing them for storage and recently also for exchange and execution, then the concept of BP modeling. For the purpose of this study, which is to provide a more general ontological considerations on some of the BP modeling languages, I will use some previous ontology-based works and then I analyze it with ontology concepts of DOLCE (Descriptive Ontology for Linguistics and Cognitive Engineering) (Masolo, et al. 2003), developed at the LOA (Laboratory for Applied Ontology), where the internship in which this thesis was developed has been carried out, since DOLCE provides good taxonomy, primitives and axioms.

To begin this study, two languages have been selected, based on their popularity, namely: BPMN (Business Process Modeling and Notation) and PSL (Process Specification Language), and as starting point, an overview of the basic construct of the language will be presented, and I will focus on their primitives and notations with respect to ontological principles, for this reason, in addition to the ontological analysis based on DOLCE, I will make some comments on the languages from a general and commonsensical ontology point of view.

---

<sup>1</sup> url: <http://www.businessdictionary.com/>

## 2. Ontology-based Analyses

In this chapter, I describe how the ontological analysis will be done, before proposing my own idea I will introduce a methodology used in some related works, so in section 2.1 I will describe an ontological analysis based on BWW representational model (BWW ontology), and in (section 2.2) I describe how the ontological analysis based on DOLCE will be done.

### 2.1. Ontological analysis based on BWW (Bunge-Wand-Weber Ontology)

The BWW framework defines a set of models based on an ontology developed by Wand & Weber in the 90ties based on a more philosophical ontology developed by Bunge in (1977), (Wand & Weber 1993; Recker et al. 2006), for evaluating modeling techniques used to model real life systems. From the set of models in the BWW framework, the BWW *representation model* specifies a number of constructs for which conceptual modeling languages that purport to model information systems domains need to provide representations. It can be used to analyze a modeling technique and predict the modeling technique's strengths and weaknesses, more particularly its capability to provide *complete* and *clear* description of the domain being modeled (Aagesen, et al., 2010). For brevity reasons the BWW representation model is not broadly discussed here, but in few words, a *BWW Representation model* is a set of thirty five (35) constructs used to evaluate information system (IS) modeling techniques; it was developed by Wand & Weber, based on Bunge's ontology, with the purpose of evaluating how well those IS modeling tools and techniques enable the modeling of real world phenomena's which in the end is the target of ISs, because according to the authors, information systems are models of the real world. Of those 35 constructs, 5 are fundamental ones, for they come from their (Wand and Weber) definitions of information system. Among these fundamental constructs we have: *Things*, *property*, *state* and *transformation*, for further details on the BWW representational model and its constructs please refer to (Weber, et al., 1990), (Wand, 1996) and (Green, et al., 1997).

As a tool for modeling language analysis, The BWW representational model is used as a reference for representational analysis of conceptual modeling languages for determining the language representational capabilities and deficiencies, assuming the goodness of BWW. By comparing BWW representational model constructs with the constructs of the target modeling language, and according to this comparison a language can be: ontologically complete and clear; ontological completeness is measured by the degree of construct deficit, and the ontological clarity is measured by the degree of construct overload, redundancy and excess.

According to the authors, the main reasons for choosing BWW are: its being an upper ontology, its being derived with Information System discipline in mind and a demonstrated usefulness for ontological analyses.

## 2.2. Ontological Analysis Based on DOLCE

DOLCE is an upper ontology, descriptive, in the sense that “it aims at capturing the ontological stands that shape human language and cognition; DOLCE is also an ontology of particulars, universals are not absent, but they are used to categorize the particulars, it focuses both on 3 and 4-dimensional objects, in the sense that, in the DOLCE domain we can find 3-D object, which are called *endurants* and 4-D objects which are called *perdurants*, and they are connected to each other by the relation of *participation*; and these objects have *qualities* (which in DOLCE are also individual entities).

As specified in (Masolo, et al., 2003), DOLCE has 37 basic categories, 7 basic relations and 80 axioms, making it somehow very axiomatized, and one of the reasons for choosing DOLCE as analysis ontology is exactly the nature of its “categories”, as specified in (Masolo, et al., 2003) they depend on human perception, their cultural inprints as well as their social conventions, in the sense that the aim of the DOLCE ontology is that of making explicit the conceptualizations that are behind a certain representation of a domain, trying to be as close as possible to a commonsensical conceptualization.

For the DOLCE based analysis, I will check the target language’s assumptions and categorization of elements, to verify if they are categorized based on ontological similarities and finally I will verify the relations, in the sense that I will check if the target language supports the types of relationships specified in DOLCE.

### 2.2.1. Ontological Choices Analysis

The first part of the analysis based on DOLCE, is the analysis of the ontological choices of the target language, three criterias are taken into account here, the bias, type of knowledge and expressiveness. A modeling language has good ontological choices with respect to DOLCE if it has a cognitive bias; its universe of discourse includes particulars and it uses a multiplicative approach.

**Cognitive Bias:** the ontology is based on human perception of their surroundings, including their social rules and cultural inprints, ultimately the concepts presents in the ontology are based on how people see and interpret the world around them, in the sense that instead of considering the reality for what it is, in DOLCE the reality is considered according to what the agents (human) perceive it to be. In my opinion, such cognitive focus would make the language easier to work with and to be understood, so in a way would make it more suitable for automatic reasoning and intereporability.

**Ontology of Particulars :** the ontology is restricted to particulars, and the use of universals is strictly for categorization purposes.

Multiplicative approach: the ontology is very expressive and even complex, since it aims at giving a reliable account of reality by allowing different entities to be co-localized in the same spatio-temporal coordinate, through the assumption that those entities are different due to their incompatible essential properties (Oberle, et al., 2007).

### 2.2.2. Elements Categorization

In DOLCE's universe of discourse we have only *Particulars* (being the topmost category), no other kinds of entities are considered, and those *Particulars* are categorized in a tree-like way. In the top most of the tree (the root) we have the *Particulars (PT)*, and those *Particulars* are divided into 4 sub-categories, *Endurant(ED)*, *Perdurants(PD)*, *Qualities* and *Abstracts*, depicting the 4 different types of entities in its knowledge domain, based on their ontological distinction and ultimately their cognitive biases.

*Endurants* are entities that exist throughout time, there can be physical or non-physical endurants, while the *Perdurants* are entities that "happen in time" (*Endurants* participate in *Perdurant*), and other two categories (*Quality* and *Abstract*), *Quality* represents the entities that inhere in both ED and PD (and that in information systems and databases are usually represented as attributes) and those entities that do not have causal power (Masolo, et al., 2003), and the *Abstract*, they represent the elements that do not exist neither in time nor in space, and ultimately they also do not have causal power (they can neither be cause nor effect of anything), the Figure 1: Taxonomy of DOLCE basic categories. taken from (Masolo, et al., 2003) depicts DOLCE basic categories, for further information on them, please refer to (Masolo, et al., 2003). So I will check if the language's element categorization follows a similar line of reasoning as the one presented here, by making considerations on the language's categorization, as the distinction among objects and events, qualities and abstractions, distinction of physical, non physical objects, amount of matter and features, and so on.

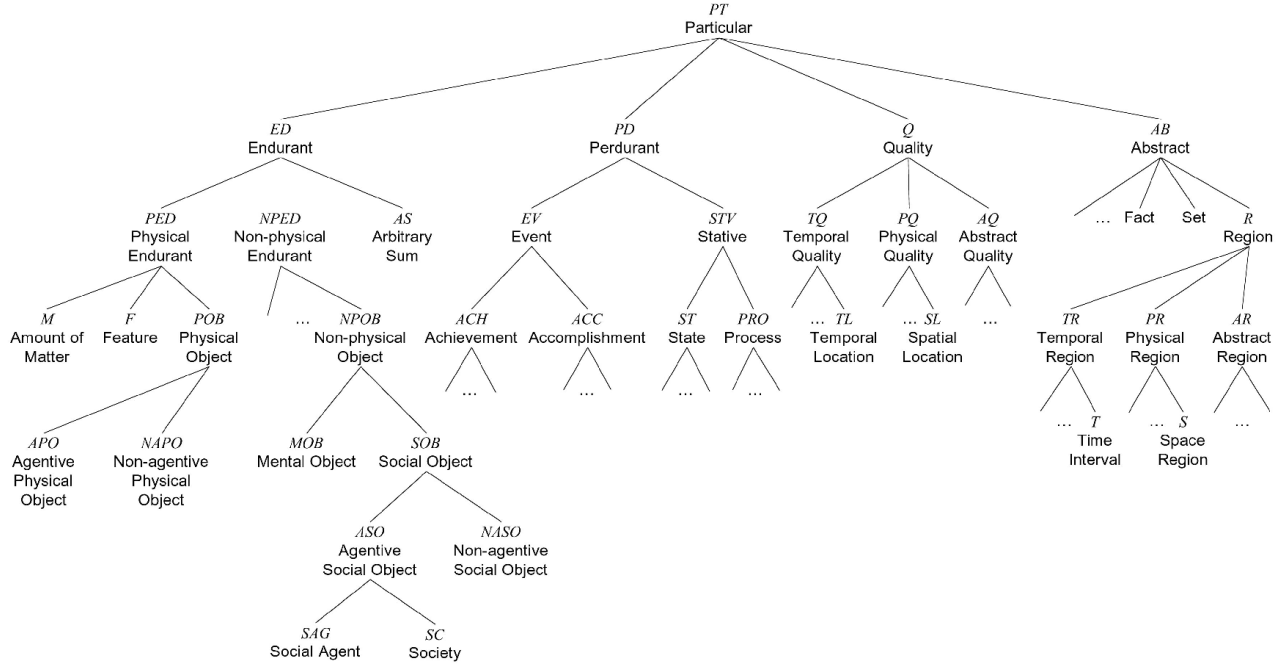


Figure 1: Taxonomy of DOLCE basic categories. taken from (Masolo, et al., 2003)

### 2.2.3. Basic Relations and Functions

In DOLCE, a set of basic primitive categories and relations is provided in order to characterize its ontology commitment, which includes: *Parthood*, *Temporary Parthood*, *Constitution*, *Participation*, *Quality* and *Quale*. The First Order Logic Formulas bellow, taken from (Masolo, et al., 2003) are the best way to describe each of the Relations and /or Functions.

**Parthood:** “*x is part of y*”  $P(x, y) \mapsto (AB(x) \vee PD(x)) \wedge (AB(y) \vee PD(y))$

**Temporary Parthood:** “*x is part of y during t*”  $P(x, y, t) \mapsto (ED(x) \wedge ED(y) \wedge T(t))$

**Constitution:** “*x constitutes y during t*”  $K(x, y, t) \mapsto ((ED(x) \vee PD(x)) \wedge (ED(y) \vee PD(y)) \wedge T(t))$

**Participation:** “*x participates in y during t*”  $PC(x, y, t) \mapsto (ED(x) \wedge PD(y) \wedge T(t))$

**Quality:** “*x is a quality of y*”  $qt(x, y) \mapsto (Q(x) \wedge (Q(y) \vee ED(y) \vee PD(y)))$

**Quale:** “*x is the quale of y (during t)*”  $ql(x, y) \mapsto (TR(x) \wedge TQ(y))$

$ql(x, y, t) \mapsto ((PR(x) \vee AR(x)) \wedge (PQ(y) \vee AQ(y)) \wedge T(t))$

Note: for the meaning of the elements in the formulas, such as (AB, PD, ED, etc) please refer to Figure 1: Taxonomy of DOLCE basic categories. taken from (Masolo, et al., 2003).

Ultimately in my DOLCE based analysis some aspects of the languages will be analyzed, based on the ontology, the analysis will comparative, but no one to one mapping or matching will done, only general consideration, regarding the *Basic Assumption* behind the language, the language's *Element Categorization*, the *Basic Relations and Functions* in the language, no measurement will be introduced, in the sense that I will not provide the percentuals to which a specific language is “good” or “acceptable” with respect to the ontology.

## **3. Business Process Modeling Languages**

### **3.1. Business Process Modeling and Notation**

BPMN is a graphical representation for specifying business processes in a business process model, it was previously known as *Business Process Modeling Notation*. According to its specification<sup>1</sup>, it was created with the primary goal of providing a notation that is readily understandable by business users, analysts who create the initial draft of the process and the technical developers who in the end are responsible for implementing the technology that would perform such processes, filling the gap between the process design and process implementations. It also adds the business-oriented notation visualization of XML based languages designed for execution, such as WS-BPEL (discussed in detail later), as one of the main goals as well.

One of the claims of BPMN developers is that it provides a human level of “inter-operability” which is not addressed by other web service-based XML execution languages, and also and before BPMN, business people used to design and visualize their Business Process in simple flow charts, creating a certain discrepancy between the initial model and the format of the execution languages, so it provides the solution for that, as well as the capability of understanding internal business procedures in a graphical way, and also the ability for communicating those business procedures in a standard manner.

#### **3.1.1. Language Constructs and Properties**

In the BPMN language the basic primitives are divided into five (5) categories, Flow Objects, Connecting Objects, Swimlanes, Data, Artifacts (OMG Group, 2010 pg 27) , this notation in turn, is further divided in core elements set (for simple notations, for simple processes) and the extended elements set (with additional graphical notations for modeling complex processes).

##### **3.1.1.1. Flow Object**

In a Business Process diagram in BPMN, there are three core elements (Events, Activities and Gateways), which are categorized as Flow Object, such name has been used to give the sense that they are part of the normal flow and they affect it the reduced number of core elements is to reduce the number of the shapes that a modeler need to know



## Event

An event represents something that happens in the course of the business process, usually having a cause and effect (trigger and result, respectively), in BPMN is a primitive type, graphically represented as circles, and are used to represent something that happens, the figure bellow depicts the 3 types of events, they differ from each other in the way they affect the flow (start, interrupt or end the flow).



Figure 2: BPMN Event Types from Wikipedia

According to the specific type of event, each of the super types of events can have a different notation, derived from the primitives above stated, those specific types are, Message, Timer, Multiple, Parallel Multiple, Rules (Conditional), Signal, Error, Cancel, Compensation, Escalation, Termination, and Link, for better understanding each of them, the table below shows the notation for each.

### Start Event

Start Event : as what the name says, it indicates the starting point of a process, it connects to sequence flow, but does not have an incoming Sequence Flow, meaning that a Sequence Flow cannot be connected to a Start Event, in BPMN 2.0 we have 10 types of Start Event, note that, the types of the events, refer to the trigger of the process, the mechanism that instantiates the process, which are: *None, Message, Timer, Escalation, Error, Compensation, Conditional, Signal, Multiple and Parallel Multiple*.

**None:** The trigger of the event is not specified. It is used for Sub-Processes that are triggered by their parent process (Virtual Architect Team, 2010).

**Message:** The trigger is a message that arrives from a specific participant. (Virtual Architect Team, 2010).

**Timer:** The trigger is a timer, which triggers the action in a time cycle fashion or at a specific time. (Virtual Architect Team, 2010).

**Escalation:** The trigger here is the need for escalation for satisfying a constraint, thus triggering an in-line Event Sub-Process (Virtual Architect Team, 2010).

**Error:** the trigger is an error that occurs, thus setting off the process; it is only allowed for triggering an in-line Event Sub-Process (Virtual Architect Team, 2010).

**Compensation:** the trigger here is the occurrence of a compensation which can only trigger an in-line Compensation Event Sub-Process. Since a process needs to be completed before this kind of event occurs, it cannot disrupt the normal flow. (Virtual Architect Team, 2010).

**Condition:** the satisfaction of a condition is the trigger of the process.

**Signal:** the trigger is a signal broadcasted from another process, (signal differ from message for the fact that the latter has a target and the former has not), since it has not specific target, it can trigger many processes (Virtual Architect Team, 2010).

**Multiple:** indicates many possible triggers, but only one of them can trigger the process, the attributes of the Event, will determine the other types of triggers that can apply (Virtual Architect Team, 2010).

**Parallel Multiple:** indicates many triggers, the process is triggered by all of them together, for Event Sub-Process (Virtual Architect Team, 2010).

**Interrupting or Non-interrupting event sub-process:** Start events can be attached to the border of a sub process, indicating the start of a sub process in line, those can be interrupting or non-interrupting, depending on the fact that the sub-process should be canceled or not: if not, multiple instances of the sub-process can run concurrently, it is important to note that, it is does not apply to Error Events, neither to compensation Events. (Virtual Architect Team, 2010) The figure bellow, from visual paradigm website, shows how the two are depicted in BPMN, one has a dashed edge and the other does not.



Figure 3: Interrupting (Left) non-Interrupting (right) event sub-process

## Intermediate Event

Intermediate Event as the name suggests, are events that occur in the course of the process, affecting it, but they cannot start a process, neither end one. Placed in a normal flow, they are used to respond to the event trigger or to set off the trigger, when responding to a trigger the marker will be unfilled (Catch), and when setting off a trigger, the marker will be filled (Throw). In BPMN 2.0

ten types of Intermediate events are defined: *None, Message, Timer, Escalation, Compensation, Conditional, Link, Signal, Multiple and Parallel Multiple*. See Figure 4: BPMN Flow Object – Events

**None:** The trigger is not specified, and they are valid only in the main flow of the process.

**Message:** The trigger here is an incoming message, which can make the flow continue (meaning that it was waiting for the message) and also send a message, but even in this case the message is sent in the normal flow.

**Timer:** the trigger is timer, which triggers the action in a cyclic time fashion or at a specific time-date and can only be used in a normal flow.

**Escalation:** the trigger here is the need for escalation, which triggers an Escalation. It's only a catch event.

**Compensation:** The trigger here is need for compensation, and it is used for compensation handling. And they can only be used in sub-processes.

**Conditional:** This type of event is triggered when a Condition becomes true.

**Link:** A Link is a mechanism for connecting two processes and they cannot link a process with a sub-process, can have more than one source but only one target.

**Signal:** the trigger here is a signal, can be used for sending or receiving signal, which should not be confused with a message, as outlined above.

**Multiple:** there are many Triggers assigned to the Event. It is used as a catching event (attached to an activity's boundary) or throwing event (in the normal flow); only one trigger is required for catching, in the other hand, when throwing all the triggers are required.

**Parallel:** the same as the Multiple event, differing only on the fact that all the assigned triggers are required to set off the event.

## **End Event**

As the name suggests, it shows where the process ends, and they do not have an outgoing sequence flow. BPMN 2.0 defines nine (9) different types of End Events according to the possible "results" of the process to differentiate from triggers, which are: *None, Message, Escalation, Error, Cancel, Compensation, Signal, Terminate, and Multiple*. See Figure 4: BPMN Flow Object – Events

**None:** In this type of event the result is not defined, usually marks the end of sub-process, returning the flow to its parent.

**Message:** This type of event indicates that the process ends with a message being sent to a participant.

**Error:** This type of event indicates that in the end of the process a named error will be generated.

**Escalation:** This result means that in the end of the process, an escalation will be triggered.

**Cancel:** This type of event is used in Transactions Sub-process to indicate that it should be canceled and also that a Transaction Protocol Cancel message should be sent to all entities involved in the transition.

**Compensation:** This result indicates that compensation is needed, thus, some steps should be undone.

**Signal:** This type of event means that in the end of the process a signal will be broadcasted.

**Terminal:** This type means that all the activities in the process should be terminated immediately.

**Multiple:** This type of event means that there are multiple possible results and all of them will occur.

**Link:** This type of event is a mean for connecting the end of one process to another process.

Type		Start	Intermediate	End
None	Unknown/not disclose trigger or related to a Sub-Process			
Message	A message is a trigger of the event or the ending point (for End events).			
Timer	A timer is the trigger of the event, in a cycle way or as a delay mechanism			
Paralel Multiple	Multiple triggers assigned to the event, start of a sub-process or catch of a process			
Multiple	Multiple triggers assigned to the event			
Rule	Event triggered when a condition is true			
Signal	A Signal is the trigger of the event, or a the end a signal should be sent.			
Error	An Error triggers an in-line Sub-Process. Or in the end a named error is generated			
Cancel	Indicates that a transation subprocess should be canceled			
Compensation	This type of End/Intermediate indicates that a Compensation is necessary			
Escalation	Triggers an in-line Event Sub-Process. An escalation is caused			
Terminate	Imediate Process termination			
Link	Mechanism for connection two envents			

Figure 4: BPMN Flow Object – Events

## Activities

An activity is a work that is performed in the course of a business process; it can be simple atomic activity or a non-atomic (compound) (White, 2006). The language allows for two types of activities Tasks and Sub-Process, both the Task and a Sub-Process share a similar shape, have a slight difference, on the plus sign added to Sub-Process (collapsed).

### Task

A task is perceived as the atomic activity to be executed within the process, to define the execution behavior of the task, some markers can be added, and each task is allowed to have up to

two different markers, which are: Loop, Multi-Instance (Parallel or Sequential) and Compensation markers; see Figure 5: BPMN Flow Object - Activity: Tasks.

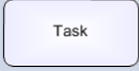
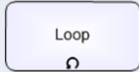


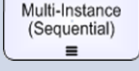
Marker	Meaning	Notation
Task	An atomic activity that is included within a Process	
Loop	Loop marker, process is performed more than one time	
Multi-Instance Parallel	A number Instances of the task executed at the same time	
Compensation	Compensation Marker	
Multi-Instance Sequential	A number of instances of the task executed sequentially	

Figure 5: BPMN Flow Object - Activity: Tasks

As of BPMN 2.0 some new notations were added to depict situations where an activity is performed by a human being, a system, situations where the activity is performed manually, with the aid of a software, and as well as if the task requires the use of external web-services and if it could require to receive or send a message for its completion, see

Table 1: Types of BPMN Tasks


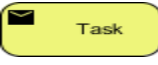

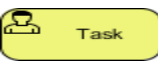
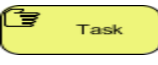
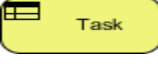
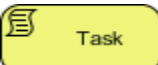
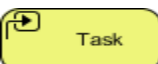
Name	Notation	Description
Service		The task uses some kind of service.
Send		The task sends message to a participant, and it is not complete until the message is sent.
Receive		The task waits for a message from a participant, and it is not complete until arrival of the message.
User		The task is performed by a human aided by some sort of application.
Manual		The task is performed by a human without any kind of engine
Business Rule		The task allows the process to provide input to a business rules engine, and to get the output from it.
Script		The task involves a script, which will be executed by a business process engine.
Reference		The task refers to another task for its content.

Table 1: Types of BPMN Tasks

## Sub-Process

Sub-Processes are compound activities, that contain another process flow within them, share the same basic graphical notation as a Task, in BPMN there is a mechanism for collapsing and expanding to better assist the modeler, and they can have the same markers as Tasks (see Figure 5: BPMN Flow Object - Activity: Tasks) for the same purpose, plus a different marker called ad-hoc which indicates that there is no sequence neither multiplicity constraints in their execution, which is determined by the performer.

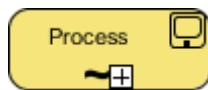


Figure 6: BPMN Ad-Hoc Process

Sub-Processes can be embedded, reusable or have a reference to another Sub-Process, in these cases, an incremented notation is used (the same as stated in section 0 and some additional marker).




Name	Representation	Description
Embedded		The process's internal details are modeled in another process.
Reusable		The process calls a pre-defined process.
Reference		The process refers to another process

Table 2: BPMN Sub-Process Types

## Gateways

A gateway is the language “mechanism” used to control the flow, determining the branching, forking or joining of paths within the process, and they can have different markers to clearly indicate the flow behavior, namely Decisions/branching (exclusive, inclusive, and complex), merging, forking, and joining. The picture below depicts all types of Gateways.










Type	Notation	Description
Exclusive Data-Based		Used to model alternative paths for the flow
Inclusive		Used to model alternative or parallel path, to those path that meet the conditions
Parallel		Used to join parallel flows
Complex		Used to model complex synchronization behavior
Exclusive Event-Based		Used to model alternative paths based on the events (Intermediate)
Exclusive Event-Based (Instantiating)		Used to model alternative paths based on the events Start
Parallel Event-Based (Instantiating)		Used to create parallel flows

Figure 7: BPMN Flow Object – Gateways

### 3.1.1.2. Connecting Objects

The BPMN language defines 3 basics “mechanisms” used to specify the directionality of the flow, one for the flow of the messages between processes (Message Flow), one for the flow of the activities themselves, in sense that they show the order in which the activities are performed (Sequence Flow), and the last one for associating flow objects with artifacts (Association). The Sequence Flow is depicted as single continuous arrowed line (filled head), the Message Flow is depicted as dashed arrowed line beginning with a small circle and the Association is depicted as dashed line.

**Sequence Flow:** is a connecting object used to connect flow objects in a process or Choreography<sup>2</sup>, thus, showing their order; it must have only one target and only one source. (OMG BPMN Group, 2010)

<sup>2</sup> “**Choreography** formalizes the way business Participants coordinate their interactions. The focus is not on orchestrations of the work performed within these Participants, but rather on the exchange of information (**Messages**) between these Participants.” (OMG BPMN Team, 2010)

- Sequence Flow with Gateways or Activities as sources, can have a condition expression, the path is taken only if the condition is evaluated to true; in case of Activity it (the activity) should have another S.F from it, and a mini-diamond is placed in the beginning of the arrow; in case of a gateway as a source, the mini-diamond is not used, and the gateway should not be of Parallel neither Event Type. See section (0).
- Sequence Flow can also be defined as default, for those flows taken when all the others S.F from the same source (Exclusive, Inclusive, Complex Gateways or Activity) evaluates to false, and to depict that, a small slash is drawn at the beginning of the arrow. See Figure 9: BPMN Connecting Objects.

**Message Flow:** It is a connecting object used to show the flow of message between two participants, meaning that the Message Flow crosses the boundaries of a Pool (pg. 20), connecting one flow object in one Pool to the boundary of another Pool or to another object in another Pool, and never in the same Pool. See Figure 9: BPMN Connecting Objects

**Association:** It is a connecting object used to “associate” Flow Objects (pg. 7) to Artifacts and/or information as well as for showing activities used in Compensation<sup>3</sup>; it is depicted as a single dotted line. . If there is a reason, an arrow head can be added to indicate directionality.

**Data Association:** It is a connecting object used to represent the movement of data between Data Objects and to “associate” inputs and outputs to Flow Objects.

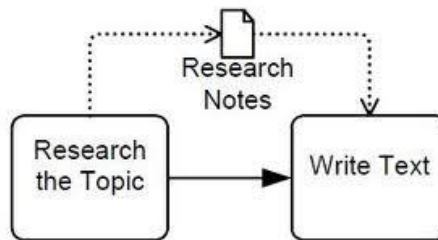


Figure 8: Data Association: (OMG BPMN Team, 2010) pg 229

<sup>3</sup> “**Compensation** is concerned with undoing steps that were already successfully completed, because their results and possibly side effects are no longer desired and need to be reversed”. (BPMN, 2010, pg. 311)








Description	Notation
Simple Sequence Flow	 Sequence Flow
An activity is the source	 Conditional Sequence Flow
Exclusive Data-Based Gateway or an activity as its source .	 Default Sequence Flow
Simple Message Flow	 Message Flow
Simple Association	 Association
Association with directionality	 Directional Association
Association with bi-directionality	 Bidirectional Association

Figure 9: BPMN Connecting Objects

### 3.1.1.3. Data

Data is an improvement in BPMN 2.0, to provide the ability to model the items that are created, manipulated, and used throughout the course of the process, and it can be represented in four different ways: Data Objects, Data Input, Data Outputs and Data Stores.

#### Data Objects, Data Input and Data Output

Data Objects are “artifacts” used to provide useful information about what a process does, they can tell how data, documents and other objects (electronic or physical) are used in the course of the process, and they are represented using a document-like shape, and special markers are used to specify if the data object is an input or an output data of a process, it is important to know that they are also associated to Flow Objects, but should not be confused with messages.

Note: for what concerns Data Input and Data Output, they are just Data Objects with a notation saying that they are used as input or as output.

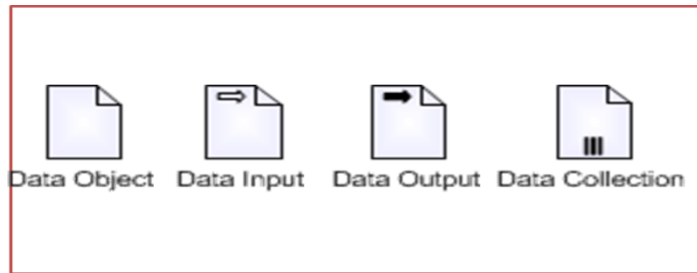


Figure 10: BPMN Data

## Data Stores

Data Stores are used to represent information systems that are out of the scope of the process, which are used by the activities to commit (make changes in information source) or query information.



Figure 11: Data Store

### 3.1.1.4. Artifacts

As Connecting objects, Artifacts are not part of the modeling domain; they are simple mechanism of the language for providing additional information of the model, information beyond the basic structure, such as information to add more descriptive information to the model, without any effect on the process. There are 2 basic types of Artifact, which are: Groups and Text Annotations.

## Text Annotations

Text Annotations are a mechanism for a modeler to provide additional information for the reader of a BPMN Diagram. The figure bellow shows how it is represented.

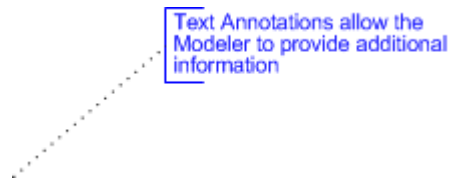


Figure 12: BPMBN Artifacts - Text Annotation

## Groups

Groups are simple visual mechanism for grouping elements in the diagram, mostly to aid the modeler and do not have effect in processes neither a meaning, and they cannot be connected by connectors of any kind, since they are not flow objects neither an activity.



Figure 13: BPMN Artifact - Group

### 3.1.1.5. *Swimlanes*

Swimlanes, as for the connecting objects and artifacts, are not part of the domain, they are mechanism to denote the participants in the process, a sort of graphical container, for the activities that each participant takes, and they can be arranged vertically or horizontally. It's important to know that the Participant can be a specific business entity (e.g., a company) or can be a more general business role (e.g., a retailer, seller, or manufacturer).

A swimlane is basically a Pool and a set of Lane, A Pool is a graphical container representing a Participant, and its responsible for separating a Process from other pools and they can be modeled without internal details (black-box), each pool can be partitioned into Lanes, for organizing and categorizing the activities inside the Pool, is important also to know that the usage of Lanes is not specified in BPMN, it is up to the modeler, generally they are used for internal roles, internal

systems, an internal departments, and many others. The figure below depicts the notation of a Pool with two Lanes.

The Sequence Flow can cross the boundaries between Lanes of a Pool, but cannot cross the boundaries of a Pool.

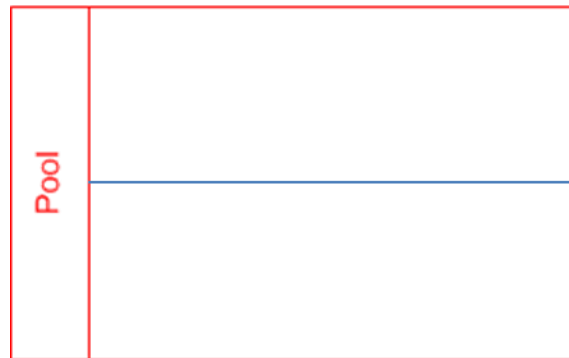


Figure 14: BPMN Swimlanes - A pool with 2 lanes

### 3.1.2. Informal Considerations on BPMN primitives

Here I will provide some considerations about some BPMN primitives, which will basically consist in stressing out the meaning of them, in the sense that I will analyze the notions introduced in BPMN from a commonsensical point of view. This will help in clarifying whether the BPMN primitives are understandable intuitively or if the users are required to see things from the point of view of the BPMN developers and so on.

#### 3.1.2.1. *General Considerations*

As stated in section (3.1.1) the BPMN constructs are grouped in five groups, according to their purpose in a Business Process Diagram, so I will start with the naming of those group of primitives ( Flow Object, Connecting Objects, Swimlanes, Data objects and Artifacts), here one can clearly see the intuition behind the choice of the terminology “Flow Object” for instance, since those 3 types of constructs (Activity, Gateway and Event), are actually “graphical objects” used to represent something regarding the flow of a process. The first problem that arises there is with the notion of Object, which is characterized by “anything that we can think, i.e. anything we can talk about” in Pierce’s words (Bergman, et al., 2003), and is valid for all five groups of constructs of BPMN, meaning that according to the notion of object all five, are also groups of some sort of object, this can lead to misconception, since the object is to group “things” according to their identifying characteristic(the role of *an ontology*), a good approach would be categorizing the BPMN in a tree-

like way as in DOLCE, in this way, it would be clear for instance that everything in the BPMN domain is an object, and from there one can start to categorize accordingly, and it would result in something like (Flow Objects, Connecting Object, Grouping Objects, Data Objects and Auxiliary Objects) or dropping the notion of object and calling them all Elements, something like (Flow Elements, Connecting Elements, Data Elements, Grouping Elements and Auxiliary Elements). The reason for dropping the “Artifact” terminology and adopt Auxiliary Elements is for its ambiguity, for instance in Signal Processing it means an error; in software development process (OMG UML Group, 2007) it means a by-product; or any “object” made by human being (Oxford English Dictionary), and the reason for replacing “object” with “element” is that the term “object” reminds of something that endures in time (so not very intuitive for events) and the term “element” instead reminds of something more abstract and not restricted to objects, it can include both objects and/or events, and this notion is shared among various fields, in a slightly different way, but we the same underlying idea, of something simple, basic that can be composed to form something more complex.

#### **3.1.2.2. *A look into the primitives: Activity***

After the more generalized point of view, let’s go down to the primitives, starting with the primitives in Flow Objects for instance, we have Activities, Events and Gateways; activities to represent “what is done” during the process flow, events to represent “what happens” during the process flow and the gateways “to route” the flow. Two sub-categories are found under Activity, namely Task and Sub-Process, the notion of “Task” introduced here is as an atomic element of the “modeling domain” and has to be intended as indecomposable activity and a sub-process is intended as a decomposable activity, in the sense that it is a process embedded within another process, and in BPMN there is no limit for the number of sub-processes a process can have, it allows for infinite nesting of processes, leaving the decision of controlling that to the modeler.

#### **3.1.2.3. *A look into the primitives: Events***

The notion of Events in BPMN is of “something that happens” in the course of the process, and there are more than 30 different visual notation for the types of Events, ranging from Start Events, Intermediate Events and End Events, which could be easily reduced if we add the other characteristics as simple attributes, for example a English word is easily understandable when reading, one does not need to memorize shapes to understand.

There is no apparent reason for having a distinct visual notation for Start, Intermediate and End events, since their position within the process clearly indicates whether it is a Start, Intermediate or an End Event, in the sense that it is something intuitive, for instance if there is an Event that has no

Incoming flows (no arrows connect to it), and from which the Sequence Flow is originated (from which the arrow is derived), it is clear that, that particular flow begins at that point, and if a Sequence Flow arrives to an Event and that Event has no outgoing flow, then it is clear that, that is an End Event and if an Event has both incoming and outgoing flows, then it is an Intermediate Event.

#### **3.1.2.4. *A look into the primitives: Connecting Objects***

Moving forward in my analysis and going to the Connecting Objects, for instance the Message Flow which connect two business participants (two different pools in a BPD), according to (OMG, 2010), the decoration of the Message Flow with an envelope is not mandatory, leaving the decision to the modeler; this might not be such a good idea, since it represents a message being sent, an envelope makes it clear, in the sense the reader will grasp the idea instantly (nothing says “message” more than an envelope icon), the presence of an envelope decoration in a Message Flow, has been used for that purpose everywhere.

#### **3.1.2.5. *A look into the primitives: Swimlanes***

The purpose of a Pool is well defined in (OMG, 2010); it is used to depict a Participant. It is an important thing to have the ability to depict participants, but problem comes when the notion of Lanes are introduced as a way of partitioning a Pool without specifying the usage of such partitioning, leaving the decision on how to use them up to the modeler, in the sense that different modelers might create the same BPD in different ways, increasing the ambiguity, which would lead to misunderstanding.

Let me assume a small business process based on the one found in (Geneva, 2009), a simple hiring process, which involves the H.R.D. (Human Resource Department), R.O. (Recruiting Office) and A.D. Accounting Department, this example does not make any assumptions on whether the R.O is within the H.R.D. nor if it is a department equivalent entity. The process goes as follows, the R.O. creates a job post and then the H.R.D. finds the candidates and then the R.O. conducts the interviews together with the H.R.D. After they have found an ideal candidate the, A.D. sets up the payroll for the newly hired employee and then R.O. gives the first day instruction to the employee and that’s all for the process, and the following Figure 15: A Pool with three Lanes shows how the process is captured in BPMN.



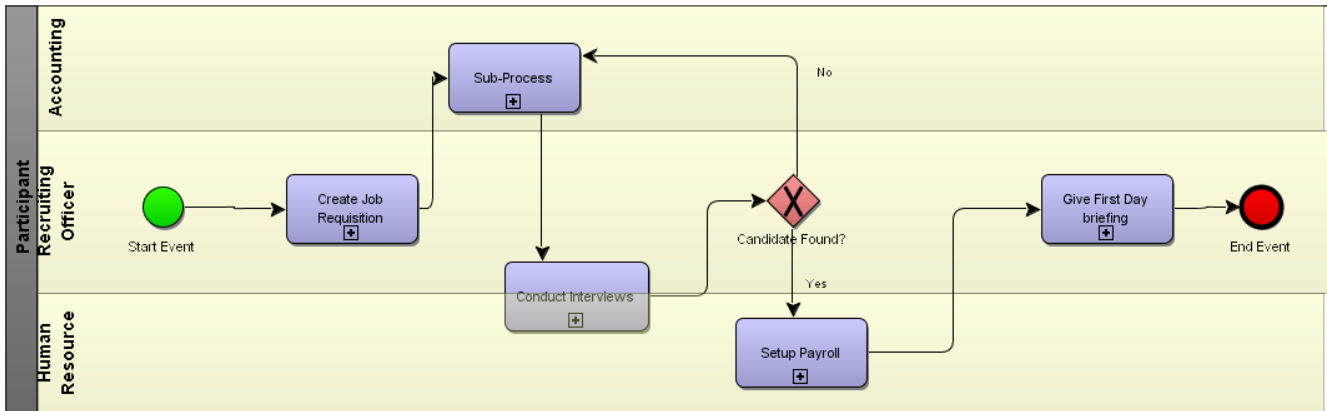


Figure 15: A Pool with three Lanes

Here we can see how the notion of an activity being done by two agents playing different Roles is being captured, since there is no clear definition on what Lanes are used for, except that they can be used for representing a specific Role in a Participant (assuming a participant to be a collective entity). So the modeler places the Activity (Sub-process) at the border of the two Lanes, since he/she is following his/her intuition and he/she is not violating the specification (OMG BPMN Group, 2010), which states “*The meaning of the Lanes is up to the modeler. BPMN does not specify the usage of Lanes.*” (OMG BPMN Group, 2010, pg. 314). Another modeler might, at his/her discretion, instead of placing the Activity between the two Lanes, use XOR gateway to split the flow into two paths to activities in the two lanes, and then merge the flows into one via another XOR gateway, so the same process can have different models without any semantic difference.

### 3.1.1.6. A look into the primitives: Artifact

The last of the BPMN element group is the artifact group, which includes Groups and Text Annotations, as already mentioned in section (3.1.1.4). They are just graphical notations used for syntactic purposes in the model, they do not have meaning in the model, they have an insignificant value, from the point of view of the model, so, it would be wiser to include them as tool mechanism, in the sense that instead of being part of the domain of the language and not having effect in what the language describes, they could be conceived of like the use of colors, lines and sizes, which is specified in (OMG BPMN Group, 2010) chapter 7.4, as a guidance to tool vendors and modelers as well, but not as a part of the language domain, as already extensively mentioned here, less symbols make it easy for users to understand, for they have less symbols to memorize.

### 3.1.2. Ontological Analysis based on BWW Ontology

The ontological analysis of BPMN based on BWW has been already made by several researchers, for instance in (Aagesen, et al., 2010), (Recker et. al, 2005), (Recker et. Al., 2007), to avoid doing a work already done, here I will just present a summary of their findings.

In their evaluation of BPMN (Aagesen, et al., 2010), they talk about the work done by Recker and others in (Recker et al., 2006) where they used the two criteria, *Ontological Completeness* (decided by the degree of *construct deficit*, which indicates to what level the modeling language maps to the constructs on the *representational model*), and the *Ontological Clarity* (decided by *construct overload*, where one of the modeling language's construct represents several BWW constructs, *construct redundancy*, where one BWW construct can be expressed by several language constructs and *construct excess*, where some constructs in the language are not represented in BWW model), based on the mapping done in (Recker, et al., 2005) .

In (Recker, et al., 2005), they did a two way mapping from BPMN to BWW constructs, the result of that mapping can be seen in Table 3: BPMN to BWW mapping by Recker et al. 2005.

From an informal analysis of the table one can clearly see the deficiency of mappings, in the sense that for the whole criteria described in section 2.1 of this thesis, BPMN was not successful, the detailed reasoning behind this mapping is out of the scope of this thesis and can be found in (Recker et al., 2008)

Ontological Construct	BPMN Construct	Ontological Construct	BPMN Construct
THING	Lane, Pool	Stability Condition	Rule, Conditional Flow
PROPERTY	N/A	Corrective Action	'Exception Task', Compensation Activity
In General	Attributes of Pools Attributes of Lanes	HISTORY	
In Particular		ACTS ON	Message Flow
Hereditary		COUPLING	Message Flow
Emergent		SYSTEM	Pool, Lane
Intrinsic		SYSTEM COMPOSITION	Pool, Lane
Mutual: Non-binding		SYSTEM ENVIRONMENT	Pool, Lane
Emergent		SYSTEM STRUCTURE	
Mutual: Binding		SUBSYSTEM	Pool, Lane
Attributes		SYSTEM DECOMPOSITION	Pool, Lane
CLASS	Lane, Data Object	LEVEL STRUCTURE	Pool, Lane
KIND	Lane	STABLE STATE	
STATE		UNSTABLE STATE	
CONCEIVABLE STATE SPACE		EXTERNAL EVENT	Start Event, Intermediate Event, End Event, Message, Timer, Error, Cancel, Compensation
STATE LAW		INTERNAL EVENT	Start Event, Intermediate Event, End Event, Message, Error, Cancel, Compensation, Terminate
LAWFUL STATE SPACE	Start Event, Intermediate Event, End Event, Message, Timer, Error, Cancel, Compensation, Terminate	WELL-DEFINED EVENT	Compensation, End Event
EVENT		POORLY-DEFINED EVENT	Message, Timer, Error, Cancel, Terminate, Start Event, Intermediate Event
CONCEIVABLE EVENT SPACE		Construct excess	Link, Off-Page Connector, Gateway Types, Association Flow, Text Annotation, Group, Activity, Looping, Multiple Instances, Normal Flow, Event (super type), Gateway (super type)
LAWFUL EVENT SPACE	Activity, Task, Collapsed Sub-Process, Expanded Sub-Process, Nested Sub-Process, Transaction		
TRANSFORMATION			
LAWFUL TRANSFORMATION	Default Flow, Uncontrolled Flow, Exception Flow		

Table 3: BPMN to BWB mapping by Recker et al. 2005

Using the two criteria described in (Aegesten & Krogstae, 2010), they put in evidence the following problems or shortcoming of BPMN:

- Not all BWB constructs can be represented in BPMN (*Construct deficit*), therefore, BPMN is not *ontologically complete*.
- Some BPMN constructs can be mapped to more than one BBW construct (*construct overload*), Lane for instance.
- We can find single BBW construct being mapped to more than one BPMN construct (*construct redundancy*).
- Finally we can find BPMN constructs not present in BBW ( *construct excess*), se table above

In (Recker, et al., 2005) the authors made a set of propositions, stating the problems that can arise from the lack of ontological completeness and clarity.

### **3.1.3. Ontological Analysis Based on DOLCE**

#### **3.1.3.1. *Ontological Choices Analysis***

This part of the analysis is divided into 3 parts, as stated in section (2.2.1), Cognitive Bias, type of knowledge and expressiveness. Those points are to measure how intuitive and expressive BPMN is taking DOLCE as baseline for expressiveness and intuitiveness, also the type of knowledge chosen for BPMN will be analyzed.

#### ***Cognitive Bias***

Taking a look into BPMN assumptions or underlying motivational ideas and comparing with DOLCE's, it is clear as shown in the following line from (White, 2006) "*The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users...*" that BPMN was created somehow with the purpose to be biased on the knowledge that business users have of their domain (the domain here refers to business process domain, being the universe of which BPMN talks about) in this way, BPMN describes conceptualizations already in the domain of business process, and it can be seen in practice, by the choices of the notations present in the language, which make use of the flowcharting technique just like UML (Unified Modeling Language), which is a standardized general purpose modeling language widely known and used in various areas (the similarities of BPMN and UML are also described in (White, 2004)). So I conclude that to some degree, BPMN has a cognitive bias, which is an important aspect since one of the objectives or design purposes of BPMN is the simplicity and easiness of understanding by the modelers and other business users (OMG, 2010).

#### ***Knowledge Type***

The second point of the analysis is to check whether the language deals with *Particulars or Universals* or with both of them, in other words, to check if the BPMN domain of discourse includes all kinds of entities (*Particulars or Universals*). This point is a bit clearer to understand; taking in consideration the whole idea behind BPMN and even the concept *model*, usually it refers to some sort of blueprint for something, for example, a model of car can be used to reproduce many cars, so the model does not represent any particular car, but all the cars that can be reproduced from it (a kind or type), so the domain of discourse of BPMN is all about *Universals*, so the three kinds of Universals (kinds or types, properties and relations (Loux, (2001))) are in some sense mentioned in it, which is suitable for Business process.

## ***Expressiveness Analysis***

The third point of the analysis is to check whether the language uses a multiplicative approach or not, for instance in DOLCE the authors take into account the possibility of given entities to have incompatible essential proprieties but, allowing them nonetheless to be co-localized in space-time, in the other hand, in BPMN there is no such notion as time, to be clearer, BPMN does not provide time constructs, so the temporal perspective is not fully captured in BPMN. This shortcoming has already been mentioned in (Gange, et al., 2008), there are now some independent efforts on adding some temporal constraints and dependencies to BPMN, and one of the examples is the work done by (Gange, et al., 2009) but that assumption is not made clear in BPMN.

## ***Results***

To summarize, with respect to DOLCE's basic assumptions, BPMN has proven not to have a strict domain of particulars, but it clearly has a cognitive bias, and finally to be not so clear about the possibility of using multiplicative approach, so it is one out of three (1/3) match between the assumptions of DOLCE and BPMN.

### ***3.1.3.2. Elements Categorization***

Going forward, to the elements categorization present in BPMN, it is not quit similar to the one in DOLCE, here I provide a table with some of the "categorization criteria" based on the distinction of the kinds to help on reasoning about them and using the Table 4: DOLCE to BPMN occurrence Mappings below we can see from the top categories that DOLCE is richly categorized in comparison with BPMN. As a starting point I will take in consideration the categorization of event-like entities in DOLCE, as a mean to evaluate how well BPMN occurrence elements are categorized; by BPMN occurrence elements here a mean *BPMN Events and Activities*.

In DOLCE we have two major types of occurrences, distinguished based on the notion of *Cumulativity*, which is the property of the mereological sum of two instances of type is also equal to an instance of the same type, in this way DOLCE identifies *stative* (cumulative) and *eventive* (non-cumulative) occurrences, furthermore DOLCE identifies two different types of *stative occurrences*, based on *homeomericity*, which is the property of all temporal parts of an instance of a certain type being described by the same expression used to describe the whole instance, so in this way we can have *states*(homeomeric) and *processes* (non homeomeric), DOLCE also presents a distinction for

*eventive occurrence*, based on *atomicity* which is the property of an occurrence to be atomic (so not further decomposable) or not.

Comparing the notations in BPMN regarding *occurrences* with DOLCE, I ended up with the following Table 4: DOLCE to BPMN occurrence Mappings , which shows the elements of DOLCE on the left and the elements of BPMN on the right, the elements in the table are organized according to the notions used in DOLCE to distinguish the different types of occurrence, the subcategories in the both ends (left and right) of the table are according to the naming given in DOLCE and BPMN respectively.

ONTOLOGY			Distinction Criteria And Examples				LANGUAGE		
DOLCE			Cumul	Homoe	Atom	Examples	BPMN		
Perdurante	Stative	State	Yes	Yes	No	floating		Activity	Process
		Process	Yes	No	No	reading	Task		
Eventive		Achievement	No	No	Yes	Ignition or termination	Start Event/ End Event	Events	
		Accomplishment	No	No	No	Triggering upon a stimulus	Inter. Event		

**Table 4: DOLCE to BPMN occurrence Mappings**

According to the mapping depicted in Table 4: DOLCE to BPMN occurrence Mappings, BPMN does not provide a clear distinction for State, but according to definition of Task in (pg. 12), both DOLCE constructs *State and Process* in BPMN can be represented as *Task* in BPMN; the notion of Process in BPMN is equivalent to the notion of Perdurants in DOLCE, so BPMN Process is not equivalent to DOLCE Process, what the authors consider process or business in BPMN (what is depicted using a BPD) is what is called *endurants* in DOLCE.

For the *Eventive occurrences*, BPMN has *Start Events* and *End Events* which are similar to *DOLCE Achievement*, and *Intermediate Events* which are similar to *DOLCE Accomplishment*, according to the definition of Events in (pg. 8) and also based on the nature of those events, for instance Intermediate Events cannot be considered as atomic because they involve the arrival of a stimulus and the execution of a trigger and it is bidirectional, in the sense that it has an incoming and outgoing flow.

Taking into consideration the notion *atomicity* present in DOLCE and in BPMN, one can see that according to BPMN a Task is atomic but, with respect to DOLCE, a BPMN Task might not be atomic. A simple Task is atomic in BPMN because it represents a unit of work done towards the completion of the process, and that unit of work cannot be broken down into a more fine grained

details, and it is represented by an atomic symbol with no details in it (rounded rectangle with a label), but if one considers the *atomicity* in DOLCE which is related to an occurrence not having parts, it opens room for different interpretations, and in the end it rises questions whether BPMN Events should be considered atomics or not.

Given a simple Business Process Diagram (BPD), for example the Order Processing BPD from (OMG BPMN Team, 2010), where a “Quotation Handling” activity is depicted as an atomic activity (Task) and there is an “Approve Order” activity depicted as “composite” activity (Sub-process), we can see that atomicity here is based on the modeler choice, he decides whether the activity is atomic or not, by modeling the activity as a Task he/she is saying that it is atomic, and by modeling it as a sub-process he/she is saying that it is non-atomic.

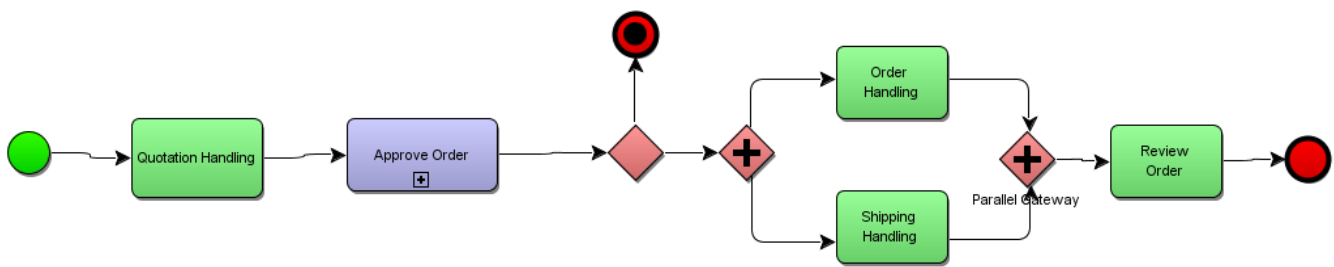


Figure 16: Order Processing form BPMN 2.0 by Examples v 1

In DOLCE they provide well axiomatized time dimension, providing the modeler with means to specify the temporal granularity of an occurrence, thus it allows for defining the atomicity more or less by the nature of the occurrence, and with a specific granularity an instant of a given occurrence can be atomic, if that instance has no parts, for example, if we think about the “Quotation handling” task, by nature, it is not really something atomic, because it might involve many different steps, depending on some other company specific factors, in this way it is not atomic anymore. Although it is represented atomically (with no details), it actually represents something not atomic, but BPMN does not provide means to express that.

For what I would call *Objects (DOLCE Endurant)*, BPMN has some elements that are in some sense equivalent, the table below illustrates an attempt of mapping the element of DOLCE against BPMN. The first thing to understand here is that I did not take into consideration the elements used as artifacts to help in the model, I took into consideration only the elements used to talk about the business process domain, that also have semantic value in a BPD therefore elements such as *BPMN Connecting Objects* and *the BPMN Artifacts* are not included in the mapping. Taking the remaining elements, the ones used in the mapping, and matching them to DOLCE, it is clear that there are some DOLCE elements that are not present in BPMN, but this is easily justified by the natures of both the DOLCE and BPMN, since the scope of one is broader and the scope of the other is a more domain specific in comparison, Linguistics and Cognitive Engineering and Business process respectively.

ONTOLOGY		Distinction Criteria And Examples				LANGUAGE			
DOLCE		Agentive	Social	Mental	Examples	BPMN			
Endurants	Physical	Agentive Physical Object	Yes	-	-	Human person	Pool, Lane		Swimlanes and Data Objects
		Non Agentive Physical Object	No	-	-	Book	Data Object		
		Feature	-	-	-	A boundary	*		
		Amount of Matter	-	-	-	Some silver	*		
	Non physical	Mental Object	-	No	Yes	A percept	*		
		Society	Yes	Yes	No	A company	Pool, Lane		
		Social Agent	Yes	yes	No	A legal person	Pool, Lane		
		Non-agentive Social Object	No	yes	no	A rule	*		

**Table 5: DOLCE Endurants to BPMN Mapping**

From the table above is also clear that in BPMN there is no distinction (at least not clear) for the different kinds of objects, the only visible distinction is with the *non-agentive physical object* which is modeled as *BPMN Data*, for the other types of endurants, they are all modeled as *BPMN Pools or Lanes*, or they are not taken into account, one possible and plausible explanation for the absence of such elements is their relevance in the field of business process, but it doesn't excuse the absence of a clear and detailed specification on Pools and Lane, a specification that would reduce the ambiguity and increase the comprehensibility of the BPDs by the model readers without having to know the roles within the company for which the model is made also for better serialization to an executable language, which is achieved through well-defined syntax and semantics, and the more descriptive it is the better.

### **3.1.3.3. Primitive Relations and Functions Analysis**

The last point of this analysis is to check the relations that are present in BPMN against the ones in DOLCE. In BPMN the relations are different from those in DOLCE (*Parthood, Temporary Parthood, Constitution, Participation, Quality, Quale*), while in DOLCE we have relations to explain the most basic things, such as how an *Object* is constituted or if it takes part in an *Occurrence*, in BPMN on the other hand, there are no explicit or formal definitions or any formal considerations about relations, but only some constraints on how to build the models; even though the relations are not explicitly defined, they can be seen from those modeling constraints and rules, those relations are the *Participation*, and *Parthood*, those are in the case where we can clearly see that a *BPMN Task* is a part of a process, taking the process depicted in Figure 16: Order Processing form BPMN 2.0 by Examples v 1 the activities "Order Handling" and "Shipping Handling" they are



parts of the “Order Processing” process and we can assume that there is an entity that executes the tasks, that entity is participating in that business process. These relations are implicitly assumed in BPMN, while they are explicitly formalized in DOLCE.

## 3.2. Process Specification Language

A process specification language is a language composed by an ontology and one or more representations, used to specify a process or a flow of processes, as well as their settings and supporting parameters, for descriptive or prescriptive purposes. (Schlenoff, et al., 2000)

### 3.2.1. The Language Structure (The Ontology)

The Process Specification Language (PSL) is an ontology designed to represent primitive concepts for describing processes, using first-order logic. For providing such ontology, PSL specifies the notions of *Language*, *Model theory* and *proof theory* (Schlenoff, et al., 2000); PSL ontology has some considerable degree of maturity for it has been developed over many years by various authors from academia and industry contributors as well. PSL consists of thousand axioms.

#### 3.2.1.1. Language

As stated in the previous paragraph, PSL specifies the notion of language, which is a **lexicon** (a set of logical (Boolean connectors and quantifiers) and non-logical symbols (constants, function symbols, and predicates) chosen to represent the basic concepts in the PSL ontology)) and a **grammar** (rules on how to combine the lexicons to form a *wff*).

#### Lexicon

For the non-logical part of the lexicon, PSL in its basic ontology, defines for primary types of **entities** ( ‘*activity*’, ‘*activity-occurrence*’, ‘*object*’ and ‘*timepoint*’ ), **functions** (*beginof* and *endof*) that return the start and end time of an ‘*activity*’ respectively, and **predicates** (*is-occurring-at*, *occurrence-of*, *exists-at*, *before*, and *participates-in*) used to express the various relations among the elements of the ontology.

**OBJECT:** is a concrete or abstract thing that can participate in an **ACTIVITY** (people, chairs, car parts, gun, numbers, etc.), they be created and used as a resource at a certain point in time, so they can have a *begin* and *end* point.

An **Activity-Occurrence:** intuitively is the occurrence of a given **Activity**, and it is characterized by its **TimePoints** and the set of **Objects** that participate in that **Activity**. (E.g. The eruption of Nablo volcano in Eritrea in 2011).

**TimePoints:** are simple points in time, they are ordered by the BEFORE relation. This relation is transitive, non-reflexive and total ordered. In PSL Core there time intervals are not considered as primitives because they can be defined with respect to *Timepoints* and *Activities*.

The notions of PSL Core are axiomatized formally as a first-order theory, to capture the basic properties of the PSL ontology in a precise way. The basic axioms for ACTIVITIES, OBJECTS, and TIMEPOINTS are listed in the (Primitives Lexicon) subsection below.

## **Grammar**

For the grammar, PSL has its grammatical roots in the KIF (Knowledge Interchange Framework) for it is based on FOL and also because KIF provides a necessary rigorousness for defining concepts in a unambiguous way and the PSL grammar is specified using BNF (Backus–Naur Form) which allows for a precise definition of the language, making it possible for translations from it to others “well defined” representational languages.

### **3.2.1.2. Model theory**

The model theory is what provides a rigorous mathematical characterization of the terminologies in PSL, identifying each term in the language with an element a mathematical structure (lattice, partial ordering or vector space), which will be used as the base for reasoning on the terms of the language and their relationships.

### **3.2.1.3. Proof Theory**

The PSL proof theory is made of three components: *PSL Core*, *one or more foundational theories*, and *PSL extensions*, which provides interpretations for the terms in the ontology.

## **PSL Core**

The PSL Core is a set of axioms that are written in the basic language of PSL; they are sound and complete regarding the model theory, meaning that they provide a syntactic representation of the model, in the sense that all the axioms are true for every model of the language of the theory, and if a sentence of the language of PSL is true in every model of PSL then it can be derived from the

axioms. (Schlenoff, et al., 2000), for this, the PSL Core is said to provide semantics for the terms in the PSL language and is it a relatively simple theory that is adequate for expressing a wide range of basic processes.

## **Foundational Theories**

In its definition PSL includes one or more foundational theories (non-definitional extensions) to compensate the weakness of PSL axioms in terms of pure logical strength<sup>4</sup>, because in the PSL Core there are not many assumptions regarding what is beyond the required for describing business process, (Schlenoff, et al., 2000), and also the theory (see 3.2.1.3) is not sufficiently strong to provide definitions for many auxiliary notions that are needed to describe a broader range of processes in a detailed manner. There are currently six (6) foundational theories: *Outer core*, *Duration and Ordering Theory*, *Resource theories*, *Actors and agent theories* (Cutting-Decelle, et al., 2002).

## **Extensions**

The Extensions is the 3<sup>rd</sup> (third) component of PSL *Proof Theory* (3.2.1.3), it provides a mechanism for expressing information that involves concepts not contained in *PSL Core*; as stated in section (3.2.1.3) in *PSL Core* subsection, the core theories are not adequate for expressing complex process, which require more expressive resources which might not be in *PSL Core*, to avoid flooding *PSL Core* with all possible concepts that could be useful for describing processes, the better solution was to provide separate extensions, those can be added to PSL Core when it is need, to satisfy a specific kind of complexity that a particular process might present, thus increasing the expressive power without. In order to add a new extension, one needs to add new constants or predicates to the basic PSL language, and for every single one of them, provide one or more axioms constrained to its interpretation, thus defining their “semantics”

### **3.2.1.4. Semantics of PSL Core**

---

<sup>4</sup> “Logical strength is the degree of support that the premises confer on a conclusion--the degree to which the premises, if true, make it likely that the conclusion is true as well. The stronger an argument is, the tighter the relationship between its premises and conclusion; the weaker it is, the looser the relationship.” (Co, 1998)

PSL Core semantics is comprised by a set of primitive and defined lexicons, supported by 17 axioms and 5 definitions, which are written in PSL language, and some KIF notation.

## **Primitive Lexicon**

In the primitive lexicon we can find relations, functions and constants, it is important to note that timepoints for instance can be specified as `inf-`, meaning that it is before all the timepoints or `inf+`, meaning that it is after all the timepoints; this is only for convention, because in the essence of PSL time is in fact infinite.

### Relations

- `(object ?x)` reads: x is an object
- `(activity ?a)` reads: a is an activity
- `(activity_occurrence ?occ)` reads: occ is an activity occurrence
- `(timepoint ?t)` reads: t is a timepoint
- `(before ?t1 ?t2)` reads: t1 is before t2
- `(occurrence_of ?occ ?a)` reads: occ is a particular occurrence of activity a
- `(participates_in ?x ?occ ?t)` reads: x participates in occ at t.

### Functions

- `(beginof ?occ)` returns the begin Timepoint of occ
- `(endof ?occ)` returns the end Timepoint of occ

### Constants *or* individuals

- `inf+` : infinite +
- `inf-` : infinite -

## **Defined Lexicon**

### Relations:

- `(between ?t1 ?t2 ?t3)` reads: t2 is between t1 and t3
- `(beforeEq ?t1 ?t2)` reads: t1 is before t2
- `(betweenEq ?t1 ?t2 ?t3)` reads: t2 is between t1 and t3
- `(exists_at ?x ?t)` reads: object x exists at time t
- `(is_occurring_at ?occ ?t)` reads: occ is occurring at t3

Note: is important to remind that all *formulaes* above are sequential, in the sense that the term  $X$  specified in the (Primitive Lexicon section) above, as an object is the same through all the *formulaes* in the current subsection.

## Axioms

The set of textual axioms bellow are exactly as they are in (Schlenoff, et al., 2000), but some their KIF representation might be a different.

**Axiom 1** *The before relation only holds between timepoints.*

```
(forall (?t1 ?t2)
  (if (before ?t1 ?t2)
      (and (timepoint ?t1)
            (timepoint ?t2))))
```

**Axiom 2** *The before relation is a total ordering.*

```
(forall (?t1 ?t2)
  (if (and (timepoint ?t1)
            (timepoint ?t2))
      (or (= ?t1 ?t2)
           (before ?t1 ?t2)
           (before ?t2 ?t1))))
```

**Axiom 3** *The before relation is irreflexive.*

```
(forall (?t1)
  (not (before ?t1 ?t1)))
```

**Axiom 4** *The before relation is transitive.*

```
(forall (?t1 ?t2 ?t3)
  (if (and (before ?t1 ?t2)
            (before ?t2 ?t3))
      (before ?t1 ?t3)))
```

**Axiom 5** *The timepoint inf- is before all other timepoints.*

```
(forall (?t)
  (if (and (timepoint ?t)
            (not (= ?t inf-)))
      (before inf- ?t)))
```

**Axiom 6** *Every other timepoint is before inf+.*

```
(forall (?t)
  (if (and (timepoint ?t)
            (not (= ?t inf+)))
      (before ?t inf+)))
```

```
(before ?t inf+)))
```

**Axiom 7** *Given any timepoint  $t$  other than  $inf^-$ , there is a timepoint between  $inf^-$  and  $t$ .*

```
(forall (?t)
  (if (and (timepoint ?t)
           (not (= ?t inf-)))
      (exists (?u)
        (between inf- ?u ?t))))
```

**Axiom 8** *Given any timepoint  $t$  other than  $inf^+$ , there is a timepoint between  $t$  and  $inf^+$ .*

```
(forall (?t)
  (if (and (timepoint ?t)
           (not (= ?t inf+)))
      (exists (?u)
        (between ?t ?u inf+))))
```

**Axiom 9** *Everything is either an activity, activity occurrence, timepoint, or object.*

```
(forall (?x)
  (or (activity ?x)
      (activity_occurrence ?x)
      (timepoint ?x)
      (object ?x)))
```

**Axiom 10** *Objects, activities, activity occurrences, and timepoints are all distinct kinds of things.*

```
(forall (?x)
  (and (if (activity ?x)
          (not (or (activity_occurrence ?x) (object ?x) (timepoint ?x))))
      (if (activity_occurrence ?x)
          (not (or (object ?x) (timepoint ?x))))
      (if (object ?x)
          (not (timepoint ?x)))))
```

**Axiom 11** *The occurrence relation only holds between activities and activity occurrences.*

```
(forall (?a ?occ)
  (if (occurrence_of ?occ ?a)
      (and (activity ?a)
           (activity_occurrence ?occ))))
```

**Axiom 12** *Every activity occurrence is the occurrence of some activity.*

```
(forall (?occ)
  (if (activity_occurrence ?occ)
      (exists (?a)
        (and (activity ?a)
              (occurrence_of ?occ ?a)))))
```

**Axiom 13** *An activity occurrence is associated with a unique activity.*

```
(forall (?occ ?a1 ?a2)
  (if (and (occurrence_of ?occ ?a1)
```

```

                (occurrence_of ?occ ?a2))
(= ?a1 ?a2)))

```

**Axiom 14** *The begin and end of an activity occurrence or object are timepoints.*

```

(forall (?a ?x)
  (if (or (occurrence_of ?x ?a)
          (object ?x))
      (and (timepoint (beginof ?x))
           (timepoint (endof ?x))))))

```

**Axiom 15** *The begin point of every activity occurrence or object is before or equal to its end point.*

```

(forall (?x)
  (if (or (activity_occurrence ?x)
          (object ?x))
      (beforeEq (beginof ?x) (endof ?x))))

```

**Axiom 16** *The participates\_in relation only holds between objects, activity occurrences, and timepoints, respectively.*

```

(forall (?x ?occ ?t)
  (if (participates_in ?x ?occ ?t)
      (and (object ?x)
           (activity_occurrence ?occ)
           (timepoint ?t))))

```

**Axiom 17** *An object can participate in an activity occurrence only at those timepoints at which both the object exists and the activity is occurring.*

```

(forall (?x ?occ ?t)
  (if (participates_in ?x ?occ ?t)
      (and (exists_at ?x ?t)
           (is_occurring_at ?occ ?t))))

```

### 3.2.1.5. Definitions

As well as for the axioms the textual definitions here presented are exactly as they were extracted from (Schlenoff, et al., 2000), with minor changes in the KIF representation, but reflecting the same thing.

➤ *Definition 1. Timepoint **q** is **between** timepoints **p** and **r** iff **p** is before **q** and **q** is before **r**.*

```

(defrelation Between (?p ?q ?r) :=
  (and (Before ?p ?q) (Before ?q ?r)))

```



➤ *Definition 2. Timepoint  $p$  is **BeforeEq** timepoint  $q$  iff  $p$  is before or equal to  $q$ .*

```
(defrelation BeforeEq (?p ?q) :=  
  (and (Point ?p) (Point ?q)  
    (or (Before ?p ?q) (= ?p ?q))))
```

➤ *Definition 3. Timepoint  $q$  is **BetweenEq** timepoints  $p$  and  $r$  iff  $p$  is before or equal to  $q$ , and  $q$  is before or equal to  $r$ .*

```
(defrelation BetweenEq (?p ?q ?r) :=  
  (and (BeforeEq ?p ?q)  
    (BeforeEq ?q ?r)))
```

➤ *Definition 4. An object exists-at a timepoint  $p$  iff  $p$  is **BetweenEq** its begin and end points*

```
(defrelation Exists-at (?x ?p) :=  
  (and (Object ?x)  
    (BetweenEq (Beginof ?x) ?p (Endof ?x))))
```

➤ *Definition 5. An activity **is-occurring-at** a timepoint  $p$  iff  $p$  is **BetweenEq** the activity's begin and end points.*

```
(defrelation Is-occurring-at (?a ?p) :=  
  (and (Activity ?a)  
    (BetweenEq (Beginof ?a) ?p (Endof ?a))))
```

➤ *Definition 6.  $a$  is a subactivity of  $b$  iff every object in  $a$  at a timepoint  $t$  is also in  $b$  at  $t$ , and  $b$  is occurring at every timepoint that  $a$  is.*

```
(defrelation Subactivity (?a ?b) :=  
  (and (forall (?x ?t)  
    (=> (In ?x ?a ?t)  
      (In ?x ?b ?t))  
    (forall ?p  
      (=> (Is-occurring-at ?a ?p)  
        (Is-occurring-at ?b ?p))))))
```

➤ *Definition 7. Activity  $a$  immediately precedes activity  $b$  iff  $a$ 's endpoint is  $b$ 's beginpoint.*

```
(defrelation Imm-precedes (?a ?b) :=  
  (and (Activity ?a)  
    (Activity ?b)  
    (= (Endof ?a) (Beginof ?b))))
```

### 3.2.2. Informal Considerations on PSL primitives

PSL's main part is the PSL Core, which is made of four (4) primitive classes, two (2) primitive functions and three (3) primitive relations, those primitive elements of PSL Core are derived from set of intuitions which are expressed by seventeen (17) axioms. Here I will analyze the elements based on their supporting axioms, in addition five (5) more relations are defined in PSL Core, the grounds for those relations are provided by a set of definitions.

Starting with the primitive classes we have in PSL, the first intuition says that there are four kinds of entities required to reason about processes, and axioms 9 and 10 already mentioned in section (3.2.1.4) regarding PSL axioms, which says that (everything is either an activity, an activity-occurrence, an object, or a timepoint and there are all different kinds of things); basically those two axioms are saying that in the domain of BP, those are the four kinds of primitives we can find. To clearly understand whether those elements are enough or not enough or even excessive, I take into consideration the intended meaning of each one of them, but intuitively.

When talking about processes in general, even when talking about business process in particular, what everyone has in mind is that those processes are carried out by some sort of entities. Those entities might have similarities but they differ from each other in various ways as well, ranging from computer programs, machinery to a human being. All these kinds of entity might take part in a process, and the only reasonable PSL class that can be used to represent those is the *Object* class, this is something that requires a more formal analysis and will be addressed in the next chapter.

From the four primitive classes, let me take for instance the Activity, the informal definition of activity is a class or type of action, and the Activity-Occurrence is a specific instance or occurrence of an activity in space and time, axioms 12 and 14 pertain to them, and those axioms state that an *activity-occurrence* is the occurrence of a single activity and it has a begin timepoint and an end timepoint, and that the end timepoint is always greater than the begin timepoint. *Objects* are mentioned in the axiom 14 pertaining to its existence. The need for this entity is questionable when we have the *occurrence-of* relation.

The first relation presented in PSL is the *occurrence-of*, the axioms 11, 12 and 13 pertain to it, and they clearly mention that it's a binary relation between an *activity* and an *activity-occurrence*. A question that one might ask here is "why do we need both *occurrence-of* and *activity-occurrence* predicates?". Well, let's take the example provided in the PSL website at NIST<sup>5</sup>,

---

<sup>5</sup> National Institute of Standards and Technology, where PSL was created

For example, the activity denoted by the term (paint House1 Paintcan1) is an instance of the class of Painting activities:

**(Painting (paint House1 Paintcan1))**

There may be multiple distinct occurrences of this instance:

**(occurrence\_of Occ1 (paint House1 Paintcan1))**  
**(occurrence\_of Occ2 (paint House1 Paintcan1))**

**(= (beginof Occ1) 1100)**  
**(= (endof Occ1) 1200)**  
**(= (beginof Occ2) 1500)**  
**(= (endof Occ2) 1800)**

There may be another instance of the class of Painting activities

**(Painting (paint House1 Paintcan2))**

that has no occurrences.

Figure 17: Activity and Activity-Occurrence Example by NIST

In the Figure 17: Activity and Activity-Occurrence Example by NIST the symbols Occ1 and Occ1 refer to *activity-occurrence*, therefore let's assume some more detailed formulas:

```
(activity_occurrence occ1)
(activity_occurrence occ2)
(occurrence_of occ1 (paint House1 Paintcan1))
(occurrence_of occ1 (paint House1 Paintcan1))
(= (beginof Occ1) 1100)
(= (endof Occ1) 1200)
(= (beginof Occ2) 1500)
(= (endof Occ2) 1800)
```

If the *occurrence\_of* predicate was *occurrence\_of (?a ?d)* where *d* is the time duration for the activity and *a* is the activity that occurs, the following equation suffice to express that an instance of the activity *a* occurs and it takes *d* amount of minutes or seconds or days. It is not ideal for specific occurrence but, very useful for modeling. And also have the same predicate as *occurrence\_of (?a, t<sub>i</sub>, t<sub>j</sub>)*, where *a* is the activity that occurs, and t<sub>i</sub> and t<sub>j</sub> are two dynamics calculated timepoints where t<sub>i</sub> is before t<sub>j</sub>, to allow the begin and end time of the process to be calculated using the functions *beginof* and *endof*.

```
(occurrence_of ?(paint House1 Paintcan1) 2hours )
(occurrence_of ?(paint House1 Paintcan1) ?beginof ?endof)
```

The next relation is the *before* relation, which is mentioned in axioms 1, 2, 3 and 4, this a very important relation, because when modeling business process, we need to specify the ordering of the events or actions, but instead of using only *timepoint* which indicates a specific period in time which is principle does not repeat, a better choice would be use the *beginof* and *endof* to specify the sequencing of events and activities, in this way, a single model would fill its purpose of represent something abstract that can be executed many times, at different locations in space and time, which is exactly what cannot happen using fixed *timepoints*, in this way it would allow for both Abstract and Executable business process.

Moving forward to the next predicate, we have the *participates-in* relation, the observations I made about *before* are also important to take into consideration for the *participates-in*, because they both have *timepoint* as one of the arguments, the axioms pertaining to it are 16 and 17, and since the *activity-occurrence* is one of the arguments as well, the observations about it should be taken into consideration as well, furthermore the axioms 16 and 17 states that the *participates-in* relation holds for *activity*, *object* and *timepoint*, and that an object can only participate in activities at timepoints where it exists and the activities is occurring this is perfect, and even useful when specifying the concurrent execution of activities of the same type, by using specific objects, it is useful here to remember what characterizes an occurrence of activity is not only the *activity-occurrence*, but also the relation pertaining the entities involved in that activity instance when it is occurring.

As stated in the first paragraph, in addition to the primitive relations, PSL Core also provides five (5) defined relations, *Between*, *BeforeEq*, *BetweenEq*, *exists-at*, *is-occurring-at*, those relations are supported by the definitions 1 to 5 respectively, those definitions are all about the time, *timepoints* and activity ordering and object existence, although not explicitly defined that way, because they provide the means for handling time related aspects of the process, for instance the *between* relation helps to specify that a particular *timepoint* is between two others, and by that helps when defining an activity that should occur for instance any time after the begin or end of another particular activity, and the *beforeEq* and *betweenEq* are very useful when there is a need to specify *timepoint* with a great degree of details, by allowing to specify a *timepoint*  $t_3$  anywhere exactly from *timepoint*  $t_1$  to *timepoint*  $t_2$  (*betweenEq*), somewhere between  $t_1$  and  $t_2$  (*between*), anywhere before a  $t_x$  until  $t_x$ , in supporting of the *before* primate relation, and also supporting the *participates-in* relation by clarifying the existence of an object and occurrence taking in consideration time, which is fundamental in business process modeling.

The PSL Core indeed is highly axiomatized and also provides good definitions for business process, but it shows clearly that its strengths are in specifying processes to be done or implemented as one-time process) as an implementation specification (Executable Business Process Model), at least that is what the axioms and definitions transpire, rather than Abstract Business Process Model.

Despite the shortcomings pointed out throughout this chapter, which is only taking into account the PSL core, is important to mention that PSL also provides what they call non-definitional extensions of PSL Core, which are also called PSL core theories, that provides one or more primitives non contained in PSL Core (Gruninger, 2009), those theories are: *Subactivity Theory*,

*Theory of Occurrence Trees*), *Theory of Discrete States*, *Theory of Atomic Activities*, *Theory of Complex Activities* and *Activity Occurrence*. In addition to those core theories PSL also presents some other extensions (theories), to better describe things such as duration and ordering, resources and actors, time and state. PSL also provides the ability for self-created extensions to better suits one's needs (NIST-MSID, 2004).

### 3.2.3. Analyses Based on DOLCE

#### 3.2.3.1. *Ontological Choices Analysis*

This part of the analysis is divided into 3 parts, as stated in section (2.2.1), Cognitive Bias, type of knowledge and expressiveness.

##### *Cognitive Bias*

According to (NIST-MSID, 2004) and (Gruninger, 2009), and which also mentioned in one of the working group reports where it says: “The goal of the NIST Process Specification Language (PSL) project is to investigate and arrive at a neutral, unifying representation of process information to enable sharing of process data among manufacturing engineering and business applications.” (Steven, et al., 1998)” and also from the statement “The purpose of PSL-Core is to axiomatize a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes” from (NIST-MSID, 2004) it is clear that the authors of PSL intended to create something that would be intuitive and neutral, at the same time unifying, to enable process exchange, integration and interoperability at the semantical level, but the choice of the Knowledge Interchange Format (KIF) which has a rigorous Backus-Naur Form (BNF) makes harder for regular business users to understand, even though the authors make use of some non KIF symbols, such as English-based terms related to the process engineering field, so, PSL is both descriptive and revisionary.

##### *Knowledge Type*

The second point of the analysis is to check whether the language deals with *Particulars or Universals* or with both of them, by checking if PSL’s the FAQ page on (NIST-MSID, 2004) which says “It refers directly to runtime occurrences of process in first order constraints, rather providing a notation or syntax “programming”. For example, a process for assembling a product may occur many times during the life of a factory. PSL refers to each occurrence of assembly separately to enable the user to write temporal constraints on what is the allowable progression of the process.” From this statement and by the axioms provided in PSL and already mentioned in section (3.2.1.4), the domain of discourse of PSL is not restrict to *Particulars*.

## ***Expressiveness Analysis***

The final point of this first part of the analysis is to check whether the language uses a multiplicative approach or not, as mentioned in section (2.2.1). In DOLCE for instance the authors take into account the possibility of given entities to have incompatible essential proprieties which allows them to describe entities that are co-localized in space-time, in the other hand, in PSL the authors take in consideration the temporal behavior of the processes but, they do not provide a clear and axiomatized support for modeling the spatial behaviors of processes but, regarding to objects and activities, PSL allows an object to exist where an activity is occurring, if and only if that object is participating in that occurrence according to axiom 17, which is about the *participates-in* relation. The idea of having different activity-occurrences at the same time leaves open the possibility to have different activity-occurrences at the same space-time.

## ***Results***

To summarize, with respect to DOLCE's basic assumptions, PSL ontology has proven to have a domain of both particulars and universals; some sort of mixed approach, being both descriptive and revisionary; and finally no axiom prohibits using multiplicative approach.

### ***3.2.3.2. Elements Categorization***

The element categorization in PSL is very different from DOLCE, while in DOLCE we the elements categorized in tree-like way, in PSL we have only for categories of elements specified in PSL core theory, and some other elements added by some extensions. The PSL ontology is organized differently if we consider the organization present in DOLCE, which categorizes the elements according to their role in the process and also according to their role as language artifacts.

Based on axiom 1, which states that everything in the domain of PSL is either an *Activity*, *Activity-occurrence*, *Timepoint* or *Object*, one can see that in PSL the authors did not take into account the possibility for modeling *Endurants* such as *Features* and *Amount of Matter*, neither for *Physical Quality* such as *Spatial Location*, and they also did not take into account some *Abstract* entities such as *Time Interval* and *Space Region*. The absence of an explicit distinction of features, amount of matter from objects or the absence of the distinction between the two different types of objects can be justified by the nature of PSL but, this could lead to ambiguous process models or specifications.

Other shortcoming in PSL, is the absence of *space regions* and *time intervals* in the Core (PSL Core), time and space variations are very important when dealing with process specification, since they both answer the important questions of from where to where and for how long a given activity should be carried out by an agent, also because most of event-like entities they perdure in time, meaning that usually they are not atomic. The only event-like entities that are atomic with respect to time are the achievement and they are result of accomplishments and/or processes, so there is a need for reasoning about time with more than just time points even for atomic entities.

### **3.2.3.3. Primitive Relations and Functions Analysis**

The final point of this analysis based on DOLCE is to verify the relations present in PSL against the ones in DOLCE. PSL presents a set of eight (8) relations, three (3) of them primitive and five (5) defined (*p\_before*, *p\_occurrence-of*, *p\_participates-in*, *p\_between*, *p\_beforeEq*, *p\_beforeEq*, *p\_betweenEq*, *p\_exists-at*, *p\_is-occurring-at*), while DOLCE five (5) basic relations (*d\_Parthood*, *d\_TemporaryParthood*, *d\_Constitution*, *d\_Participation*, *d\_Quality*, *d\_Quale*), here am using prefixes p and d to denote PSL and DOLCE, respectively. Due to its nature, only one of the PSL relations share similarities with one of the DOLCE's relations, the *p\_participates-in* relation matches the *d\_participation* relation, they match both in nature and in the number of arguments they take, the other relations such as *d\_Quality* and *d\_Quale* and the two kinds of parthood are not explicitly mentioned in PSL.



## **4. Conclusions**

This chapter presents the main contributions of this thesis, suggests improvements to my evaluation methodology, and also presents some conclusions and points out possible future paths where further research work is needed.

This chapter is structured as following: Section 4.1 summarizes the main contributions of this thesis. Section 4.2 suggests what can be improved in the present work. Section 4.3 present the overall conclusions and section points out a possible future work.

### **4.1. Contributions**

In the present work, I have proposed a business process modeling language ontological evaluation methodology based on DOLCE ontology, a variation of the methodology that uses BWW ontology, which has already been mentioned in (Aagesen, et al., 2010) and others works such as (Recker, et al., 2007) and also pointed out some ontological shortcomings of PSL and BPMN languages, most of them relevant to the field of semantic interoperability and process exchange. The evaluation method I present here points out the weaknesses of a language using DOLCE as a reference, ranging from the ontological choices to the element categorization and relation specification.

### **4.2. Suggested Improvement to the present work**

The evaluation methodology I proposed and used in this work lacks on formalism, in the sense that it still need a set of formalizations and some sort of measurements and metrics mechanisms, providing quantifiable results, either in percentage or in a custom evaluation scale. The formalism should also include how the analysis is done, specifying the steps to be taken.

### **4.3. Overall Conclusions**

There are some critical differences between PSL and BPMN in the aspects of objective, vocabulary, domain and realization method, adding to this fact, the idea of DOLCE as a reference for evaluating the ontological goodness of a modeling language is at the best, something new, there isn't a framework for Information System or formal languages evaluations based on DOLCE, but throughout this work I have used it as a reference for checking some specific aspects two modeling languages, BPMN and PSL.

The objectives behind the creation of PSL are mainly oriented to manufacturing processes while BPMN is oriented to business process in general, so BPMN has a wider scope than PSL, and PSL is more oriented to specification of process instances while BPMN is more oriented to modeling.

BPMN uses a simple and intuitive notation for represent complex business process while PSL uses a complex FOL based statement for specifying processes. In PSL the authors have chosen Knowledge Interchange Format (KIF) as the description as the process description to allow the extraction of the process's ontology, this is a good approach from the point of view automation because it makes the process machine readable and comprehensible but in the other hand it is a bit difficult for an average business user without specific training in KIF or FOL to able to understand all aspects of the process, even experienced users might take time to understand a complex process specified in PSL, there is where BPMN comes in with an opposite approach, by making the models easy for human but hard for machines. We need a language that can in some sense use these two approaches, providing a language that is easy to use by both humans and machines, at the same time maintaining the semantics component of business process.

In this work, I pointed out some issues in both languages, nonetheless they are not useless they both have strengths and weaknesses from the ontological point of view, for instance the axiomatization of PSL and its formal definitions and BPMN's categorization of elements and cognitive bias are some of the goodness that one can find, in the other hand the incompleteness of the elements necessary to model business process in PSL, elements such as the notion of space and the lack of time in BPMN are some of the shortcomings. Another major shortcoming is the poor axiomatization for relationships, for business process in general, it is important to formally define and axiomatize the relations; this is one of the many ways of improving semantical interoperability. PSL provides some relations, sufficient enough to reason about manufacturing processes, and it accomplishes its purpose to some acceptable extent.

#### **4.4. Future work**

For the future projects, I will propose a language that merges what is good in both PSL and BPMN, adding to some new methodologies and also specify a generic ontology for business process based on an upper ontology as DOLCE which has both linguistical and cognitive engineering perspective to support the language, the idea is to have both, graphical notation and mathematical notation which can be serialized to xml representation for integration and exchanges. The graphical notation for the end user to be able to understand the model or specification with minimal effort as possible, and the underlying mathematical specification to facilitate automated reasoning.



## Bibliography

**Aagesen, Gustav and Krogstie, John. 2010.** Analysis and Design of Business Processes Using BPMN. [book auth.] J. vom Brocke and M. Rosemann. *Handbook on Business Process Management 1, International Handbooks on Information Systems*. Berlin : Springer, 2010, pp. 212-235.

—. 2010. Analysis and Design of Business Processes Using BPMN. [book auth.] J. vom Brocke and M. Rosemann. *Handbook on Business Process Management 1, International Handbooks on Information Systems*. Berlin : Springer, 2010, pp. 212-235.

**Bergman, Mats and Paavola, Sami. 2003.** Commens Pierce Dictionary: Object. *The Commens Dictionary of Pierce's Terms*. [Online] 2003. [Cited: 07 07, 2012.] <http://www.helsinki.fi/science/commens/terms/object.html>.

**Co, WW Norton &. 1998.** wwnorton. [Online] 1998. [Cited: 07 09, 2012.] <http://www.wwnorton.com/college/phil/logic3/ch5/logical.htm>.

**Cutting-Decelle, A.F, et al. 2002.** A new Challenge for CE: PSL a stadardized language for an integrated supply chain management. Cranfield : A.A. Balkema Publishers, 2002, pp. 1051-1055.

**Gange, D. and Trudel, A. 2009.** Time-BPMN. [ed.] Birgit Hofreiter. *IEEE Conference on Commerce and Enterprise Computing*. Vienna : IEEE Computer Society, 2009, pp. 361-367.

**Gange, Denis, Trudel, André and University, Acadia. 2008.** The Temporal Perspective: Expressing Temporal Constraints and Dependencies in Process Models. [book auth.] Layna Fischer. *2008 BPM & Workflow Handbook - SPOTLIGHT ON HUMAN-CENTRIC BPM*. s.l. : Workflow Management Coaliton, 2008.

**Geneva, Rick. 2009.** Pools and Lane, Learn how to Swim first. *Rick Geneva – ProcessModeling.info*. [Online] April 8, 2009. [Cited: July 16, 2012.] <http://www.rickgeneva.com/wp/posts/swimlane-lane-or-pool-learn-to-swim-first/>.

**Green, Peter, Wand, Yair and Weber, Ron. 1997.** Panel: Ontological Analysis of Information Systems Analysis and Design (ISAD) Grammars - Experiences and Results to Date. *ECIS*. 1997.

**Gruninger, Michael. 2009.** Using the PSL Ontology. [book auth.] S. Staab. *Handbook of Ontologies*. Berlin : Springer-Verlag, 2009.

**Loux, Michael J. (2001).** PART I - Universals. *Metaphysics: Contemporary Readings*. London : Routledge, (2001).

**Masolo, Claudio, et al. 2003.** *WonderWeb Deliverable D18, Ontology Library (Final)*. Trento : ISTC-CNR, 2003.

**NIST-MSID. 2004.** PSL Core - Process Specification Language. *Process Specification Language (PSL)*. [Online] National Institute of Standards and Technology, 2 17, 2004. [Cited: 07 07, 2012.] [http://www.mel.nist.gov/psl/psl-ontology/pslcore\\_page.html](http://www.mel.nist.gov/psl/psl-ontology/pslcore_page.html).

**Oberle, Daniel, et al. 2007.** DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *Web Semantics: Science, Services and Agents on the World Wide Web*. 2007, Vol. Volume 5, Issue 3.

**OMG BPMN Group. 2010.** *BPMN Elements*. s.l. : OMG, 2010. pp. 29-30.

**OMG BPMN Team. 2010.** *BPMN 2.0 by Example*. [Document] s.l. : Object Management Group, Inc., 2010. dtc/2010-06-02.

—. **2010.** Choreography. [book auth.] OMG-BPMN Team. *Business Process Model and Notation v 2.0*. s.l. : OMG, 2010, p. 325.

**OMG. 2010.** *Business Process Modeling Language v 2.0*. s.l. : OMG Group, 2010.

**OMG UML Group. 2007.** OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2. [book auth.] OMG. *OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2*. s.l. : OMG, 2007, p. 197.

**Recker, J., et al. 2008.** *An Exploratory Study of Process Modeling Practice with BPMN*. s.l. : BPMCenter Report, 2008.

**Recker, Jan, et al. 2005.** Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. Sydney : s.n., 2005.

—. **2005.** Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. Sydney : s.n., 2005.

**Recker, Jan, Rosemann, Michael and Krogstie, John. 2007.** Ontology-Versus Pattern-based evaluation of Process Modeling Languages: A Comparison. *Communications of the Association for Information Systems*, . 2007. 20(48), pp. 774-799.

—. **2007.** Ontology-Versus Pattern-based evaluation of Process Modeling Languages: A Comparison. 2007, pp. 774-799.

**Schlenoff, Craig, et al. 2000.** *The Process Specification Language (PSL) Overview and Version 1.0 Specification*. Gaithersburg : National Institute of Standards and Technology, 2000.

**Steven, R. Ray, et al. 1998.** *Process Specification Language: An Analysis of Existing Representations*. s.l. : NIST, 1998. NIST Interagency Report 6160 .

**2010.** Tasks & Sub-Process in BP-VA. *Visual Paradigm*. [Online] Deceber 05, 2010. [Cited: July 03, 2012.] [http://www.visual-paradigm.com/support/documents/bpvauserguide/601/616/32729\\_taskandsub-p.html](http://www.visual-paradigm.com/support/documents/bpvauserguide/601/616/32729_taskandsub-p.html).

**Tenney, Richard L and Simovici, A Dan. 1989.** *Theory of Formal Languages With Applications*. Singapore : World Scientific Publishing Co. Pte. Ltd., 1989. ISBN 981-02-3729-4.

**Virtual Architect Team. 2010.** Business Process Modelling. [book auth.] Virtual Architect Team. *Business Process Virtual Architect User's Guide*. s.l. : Visual-Paradigm, 2010.

**Wand, Yair. 1996.** Ontology as a foundation for meta-modelling and method engineering. *Information & Software Technology*. 1996, Vol. 38.

**Weber, Ron and Wand, Yair. 1990.** An Ontological Model of an Information System. 1990, Vol. 16.

**White, Stephen A. 2006.** *Introduction to BPMN*. s.l. : IBM Corporation, 2006.

—. **2004.** Process Modeling Notations and Workflow Patterns. 2004.