Business environment-aware management of service-based business processes

Olfa Bouchaala

NNT : 2016SACLL004

THESE DE DOCTORAT
DE
L'UNIVERSITE DE SFAX
ET DE
L'UNIVERSITE PARIS-SACLAY
PREPAREE A TELECOM SUDPARIS

ÉCOLE DOCTORALE N°573
Interfaces : approches interdisciplinaires / fondements, applications et innovation

Spécialité de doctorat : Informatique

Par

Mme Olfa BOUCHAALA CHARFEDDINE

Gestion sensible au métier des processus à base de services

Thèse présentée et soutenue à Evry, le 30 septembre 2016 :

Composition du Jury :

Mme D. Grigori, Professeur, Université Paris-Dauphine, France, Président
Mme. C. Souveyet, Professeur, Université Paris 1, France, Rapporteur
Mme. N. Bellamine Ben Saoud, Maitre de Conférence (HDR), Université de la Manouba, Tunisie, Rapporteur
M. S. Bhiri, Maitre de Conférence (HDR), Université de Monastir, Tunisie, Examinateur
M. S. Tata, Professeur, Télécom SudParis, France, Directeur de thèse
M. M. Jmaiel, Professeur, Université de Sfax, Tunisie

Résumé

Les entreprises adoptent de plus en plus les processus métiers pour automatiser leurs activités. Cependant, ces entreprises opèrent dans un environnement métier très dynamique. Un environnement métier représente tous les facteurs externes à l'entreprise qui influencent son fonctionnement tels que les facteurs sociaux, économiques, etc. En opérant dans un environnement métier dynamique, les politiques des entreprises changent. Par conséquent leurs processus doivent être gérés et adaptés de point de vue métier. Ainsi, nous référons dans cette thèse par gestion sensible aux changements de l'environnement métier pour les processus métiers qui consiste à configurer ces processus afin de changer leurs comportements en réponse aux évènements de l'environnement métier.

Il existe trois types d'approche de gestion sensible aux changements de l'environnement métier : à savoir les approches impératives, déclaratives et hybrides. Les approches déclaratives sont basées sur les règles. Ainsi, elles sont très flexibles puisqu'on peut ajouter, supprimer ou modifier une règle lors de l'exécution. Cependant, elles sont très couteuses en termes de temps à cause de l'inférence de l'environnement métier. De plus, le processus doit être re-modéliser en règles. En contre partie, les approches impératives sont très efficaces mais trop rigide puisqu'il faut spécifier les actions de gestion lors de la modélisation des processus. Les approches hybrides, à leur tour, essaient de concilier entre les approches impératives et déclaratives afin d'augmenter le niveau concurrentiel des entreprises. Néanmoins, elles nécessitent un effort d'alignement entre la logique métier et la logique du processus. En outre, nous constatons que certaines approches ne sont pas faisables en pratique puisqu'ils n'utilisent pas les standards des processus. De plus, l'efficacité et la flexibilité sont antagonistes.

Pour résoudre ces problèmes, nous fixons les objectifs suivants : (1) concilier les techniques déclaratives et les techniques impératives en une approche hybride pour tirer profit de leurs avantages, (2) préserver les standards des processus, et (3) minimiser l'effort des concepteurs. Nous avons ainsi proposé une nouvelle approche hybride pour la gestion des processus métiers. Nous avons modélisé la gestion d'un processus métier par un processus de gestion connecté au premier qui permet de le superviser et le configurer. Ce processus de gestion est généré grâce à une modélisation sémantique des relations entre les processus métiers, les services et l'environnement métier. Nous avons également implémenter cette approche et l'évaluer en comparaison avec les approches existantes de gestion sensible aux changements de l'environnement métier.

Mots clés: processus métiers, environnement métier, efficacité, flexibilité

Abstract

Over the last decade, companies have increasingly search for managing their processes from technical and business perspectives. Management, in this case, can be of three types: technical management, business goals-based management, and business environment-aware management. While a lot of work has been done on technical and business goals-based management, only few work dealt with business environment-aware management (BEAM).

However, there is also a great need to manage business processes from a business environment point of view due to continuous business environment changes. Indeed, companies struggle to find a balance between adapting their processes and keeping competitiveness. While the imperative nature of business processes is too rigid to adapt them at run-time, the declarative one of the purely rule based business processes is, however, very time consuming. Hybrid approaches in turn try to reconcile between these approaches aiming to reach the market requirements. Nevertheless, they also need an effort for aligning business logic and process logic.

Therefore, in this thesis, we focus on business environment-aware management of service-based business processes (SBPs) aiming at conciliating imperative and declarative approaches. Our challenge is to develop a hybrid management approach that preserves industry standards to describe and to manage SBPs as well as minimizes designers' efforts. Based on a semantic modeling of business environment, business processes as well as their relationships, and a control dependency analysis of business processes, we are able to synthesize a controller, itself modeled as a process, connected to the business process to be monitored and configured at run-time. We also validated the feasibility of our management approach by implementing the framework Business Environment-Aware Management for Service-based Business processes (BEAM4SBP). Experimentations show the efficiency of our approach with respect to other BEAM approaches.

Keywords: Service-based business processes - business environment-aware management - efficiency - flexibility

In loving memory of my father
My role model, my inspiration and my motivation
My mother
No words are sufficient to describe her sacrifices.
Her unconditional support and encouragement impelled me
to set high goals and the confidence to achieve them.
My sister and my brother
Thanks for supporting me all along
and always being by my side.
My husband and my daughter
I give my deepest expression of love and appreciation
for the encouragement that you gave
and the sacrifices you made during this graduate program.

Olfa Bouchaala

Acknowledgement

The work presented in this dissertation was carried out in Telecom SudParis, France and National engineering school of Sfax, Tunisia.

First and foremost, I would like to express my special appreciation and thanks to my supervisor, Professor Samir TATA at Telecom SudParis. I am really appreciating all his contributions of time, support, encouragement and invaluable guidance to make my Ph.D.

I would like to express my deep gratitude to my supervisor, Professor Mohamed JMAIEL at National Engineering School of Sfax, Tunisia for his generous support and consistent kindness throughout the course of the work.

I would like to thank all my colleagues in the ReDCAD laboratory at National school of engineering of Sfax and my colleagues in Simbad team at Telecom SudParis.

Olfa Bouchaala

Table of contents

List of Figures

List of Tables

List of abreviations

AOP	Aspect Oriented Programming
BEAM	Business Environment Aware Management
BEAM4SBP	Business Environment-Aware Management for Service-based Business Processes
BGM	Business Goals-based Management
BPEL	Business Process Execution Language
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
SBP	Service-based Business Processes
BPSS	Business Process Specification Schema
BWW	Bunge-Wand-Weber
CDG	Control Dependency Graph
CEP	Complex Event Processing
CFG	Control Flow Graph
DAT	Dependency Analysis Tool
DDG	Data Dependency Graph.
DRL	Drools Rule Language
ECA	Event-Condition-Action
EPC	Event-driven Process Chain
IS	Information System
IT	Information Technology
KPI	Key Performance Indicator
LTL	Linear Temporal Logic
MOC	Maximum Ontological Completeness
MOO	Minimum Ontological Overlap
OMG	Object Management Group
PDT	Post-Dominators Tree
PPM	Process Performance Metric
PRR	Production Rule Representation
R2ML	REWERSE I1 Rule Markup Language
ReteOO	Rete algorithm for Object-Oriented systems
SBVR	Semantics of Business Vocabulary and Business Rules
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SRML	Simple Rule Markup Language
SWRL	Semantic Web Rule Language
TM	Technical Management

UML	Unified Modeling Language
UMO	Upper Management Ontology
WfMC	Workflow Management Coalition
WSCI	Web Service Choreography Interface
WSDL	Web Service Definition Language
XPDL	XML Process Definition Language

Introduction

Research context and problem statement

More and more companies are using business processes in order to automate their activities [8].

Over the last decade, companies were interested in managing their business processes from technical and business perspectives. Indeed, business processes encounter highly dynamic business and execution environments. Therefore, in order to keep high level of competitiveness within markets, companies have ever been interested in managing performance and quality of their processes to face business and technical changes at runtime [9, 10, 11, 12, 13].

Management of business processes can be performed according to three different sides: business goals, business environment and execution environment. The business goals of a process define the targets of a business enterprise such as increasing revenues or reducing outgoings. The execution environment represents the Information Technology (IT) where the service operates. It includes process engines, applications, servers, machines, network and their properties, etc. Whereas, the business environment connotes all factors external to the business enterprise that greatly influence its functioning. It covers economic, political, natural, social factors (*e.g.* during a sales promotion, there is a decrease of clothes prices), etc.

Accordingly, we identify three types of business process management at run-time: technical management, business goals-based management, and business environment-aware management (BEAM). The technical management [14] focuses on monitoring and configuring execution environments of business processes (*e.g.* process containers, memory, CPU). Business goals-based management [10, 15, 16] aims at aligning business goals with IT services. It is often specified using Key Performance Indicator (*e.g.* order fulfillment lead time < 3 days). However, BEAM [17, 18, 11, 4] connotes monitoring and configuring business processes in order to change their behavior in reaction to business environment events.

We notice that there were great efforts in dealing with technical management of business processes (*e.g.* [9], [14] and [19]). There were also great efforts in managing business processes from a business goal point of view that aims to align business goals and IT aspects of business processes (*e.g.*[15], [16], [10], [20] and [21]).

Beside technical management as well as alignment of IT and business goals, we argue, in this thesis, that there is also a great need to manage business processes from a business environment point of view (i.e. BEAM). Indeed, enterprises are more and more within highly competitive and constantly changing business environments.

Thus, they need to focus on adapting their business processes to address competitions and changes within such environments.

We distinguish two main types of approaches of BEAM: imperative and declarative. On one hand, declarative approaches describe both the business process and its management based on rules (e.g. Event-Condition-Action rules) (see for example [17, 22]). They are flexible, since their managing rules are well adapted to be added, removed or changed at runtime. Nevertheless, they are time consuming because of inference over rules that should be made in reaction to each change in a business environment. In addition, declarative approaches may not adopt standard notations for business process modeling such as Business Process Model and Notation (BPMN) [23] or Business Process Execution Language (BPEL) [24]. As a result, the process designer ought to re-model the process based on rules. On the other hand, imperative approaches describe both the business process and its management as an imperative process. It consist in hard coding management actions into the managed business processes. Consequently, they may preserve standard notation for business process modeling and may be very efficient, in terms of execution time. Nevertheless, they are too rigid due to over-specifying processes at design time. It is then difficult if not impossible to change the management rules at runtime.

Therefore, different approaches (e.g. [18, 5, 11, 4]) put forward hybrid techniques by separating business logic which defines the company's policies (described using Business Rule) and the process logic (described using imperative business process). This separation needs an effort of business process and business rule integration by using for example aspect-oriented approaches which later on raises issues on maintainability and transformation [25]. Furthermore, some approaches may need designers efforts (e.g. [3, 11]) or did not preserve industry process standard language (e.g. [2]).

Hence, declarative approaches may be flexible in general and may not utilize process standards which make them difficult to use in practice. Furthermore, they are very time-consuming. Imperative approaches are efficient but they are not flexible enough to easily address changes in business environments. Hybrid approaches that separate business process and business rule need efforts of alignment at run-time.

Thesis overview

Objectives

In order to deal with the limitations and issues stated in Section , our objective is to propose a novel hybrid approach for BEAM that allows: (1) conciliating imperative and declarative approaches, (2) preserving business process industry standards, and (3) minimizing process designers efforts.

Conciliating imperative and declarative approaches in an integrated hybrid

approach allows getting benefit from both of them. This could become necessary in order to keep a high level of competitiveness by reconciling between the efficiency and the flexibility of business processes.

Preserving business process industry standards when describing and managing business processes makes the BEAM approach feasible in practical use. Indeed, companies are more and more using business process standards. Hence, instead of proposing new languages for managing business processes, there is a great need to keep using industry standards.

Minimizing process designers efforts makes the BEAM approach easier in practice for companies. The main goal is to discharge the process designer from additional tasks such as re-modeling business processes using new language or adding some configuration information such as variability points.

Approach

To achieve our targeted goals, we propose a novel hybrid approach for business environment-aware management of service-based business processes. Our proposed approach consists in modeling, monitoring and configuring actions as a management process connected to the business process to be managed (that we call managed process). Monitoring reads properties of services that compose the business process while configuration alters values of these properties. Our main challenge is to generate the management process based essentially on a semantic model describing business environment events, business processes, services and their relationships.

As a result, contrary to the imperative approaches, in our approach, the management process defines several management paths. Therefore, it can encapsulate different management behaviors. The choice of a management path is based on events of the business environment which are semantically described. Consequently, our approach presents a degree of flexibility inherited from declarative approaches. In addition, our approach is proved efficient, compared to declarative approaches, since it models the management of a business process as a process connected to it. Moreover, our approach minimizes process designers efforts since it doesn't define new management rules but generates automatically the management process based only on semantic annotations. Third, as opposite to hybrid approaches separating business processes and business rules, our approach is based on a synergic use of semantic services and an upper management ontology which facilitates the alignment between the business environment and business processes.

Contributions

The major contribution of this thesis is an **algorithm for generating a management process** for monitoring and configuring a given managed process. Therefore,

we investigated, throughout this thesis, how to generate the management process automatically. The issue is how to compose management operations in order to alter service properties.

In order to generate this management process, we propose a semantic model. We also adopted and adapted a dependency analysis strategy in order to explicitly describe the structure of the managed process.

Hence, the second contribution of this thesis is a **semantic model** describing business processes, services, business environment and relationships between them. It describes essentially an upper management ontology correlated with a domain ontology. It is used for annotating semantic services and facilitating the management process generation.

The third contribution consists in adapting a **dependency analysis strategy** in order to identify dependency analysis relationships between services or activities of a managed process written in BPMN. In fact, generally, data and control dependencies are dimensions solely for programming (or programming-like) langages. However, we argue that the unstructured and highly parallel real world processes written in BPMN render them inadequate which requires some adaptations. Thereby, in this thesis, we adapted and tailored their dependency analysis strategy according to BPMN 2 requirements.

To test the feasibility of the proposed approach, we implemented a business environment-aware management framework getting as input the managed process and the corresponding domain ontology, while resulting the adequate management process.

To evaluate the expressiveness of our approach, its efficiency and flexibility, we established qualitative and quantitative evaluations.

Outlines

This thesis includes 8 chapters organized in three parts.

The first part includes chapters 1 and 2 which present the state of the art and the background of our research work. In chapter 1, we discuss briefly the service and SBPs facets and their different management types. We also present a typical scenario to motivate and illustrate our approach. In chapter 2, we review the work related to our thesis. We study different approaches on business environment-aware management. We introduce their models and analyze their solutions. This analysis allows us to identify the objectives of our research work.

Chapters 3, 4 and 5 represent the second part which details the contributions of our thesis. Chapter 3 overviews our approach to manage service-based business processes against business environment events and represents the proposed semantic model used for facilitating the management process generation. In chapter 4, we

present our adopted and adapted dependency analysis strategy for facilitating the management process generation. It explicits both control and data dependencies for BPMN business processes. Chapter 5 shows how to generate a management process based on both the proposed semantic model and the dependency analysis of a managed process.

In the third part of this thesis (chapters 6, 7 and 8), we present applications and experiments to validate and evaluate our approach. Indeed, in chapter 6, we present our *BEAM4SBP* framework and its integration into the Activiti business process engine. Both chapters 7 and 8 represent qualitative and quantitative experiments of our approach compared to the existing business environment-aware management approaches.

Finally, we summary our work and give an outlook to the future work.

Part I

State of the art

Chapter 1

Facets and management types of service-based business processes

Introduction

Business processes represent a key concept for automating companies' activities [8]. The *Workflow Management Coalition (WfMC)* defines a business process by: *"A set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships"* [26]. Service-based business processes (SBPs) are business processes where some activities are realized by services.

Business processes and their services have two types of facets: business and technical facets. The business facet represents business goals and business environments of processes and services, while the technical facet represents the implementations and the execution environment of services and processes.

SBPs are more and more encountering highly dynamic business and execution environments. Therefore, companies have ever been interested in managing performance and quality of their processes to face business and technical changes at run-time [9, 10, 11]. In fact, managing processes and services in this case consists in two main activities: monitoring and configuration.

According to facets of SBPs, we distinguish three types of management: technical management, business goals-based management and business environment-aware management. The technical management monitors service/business process implementation (e.g. QoS) and configures accordingly the execution environment [27]. The business goals-based management monitors business goals and configures technical facets [10]. However, the business environment-aware management monitors changes in the business environment and configures technical facets.

In the following sections, we briefly discuss issues on service and business process facets. Besides, we introduce an overview of their different management sides based on a running example in order to give general bases for the upcoming chapters.

1.1 Facets of service-based business processes

SBPs, we are dealing with, are used to meet a specific business goal by executing a set of coordinated activities where some of them are realized through services. Thereby, before introducing business process facets, we start by depicting service facets.

1.1.1 Service facets

As shown in Figure. 1.1, a service has business facets and technical facets. Business facets include business goals and a business environment. In addition, technical facets comprise both an implementation and an execution environment. In the following, we detail each facet:

- **Business goals** represent what the service achieves from a business perspective. They are clear targets that need to be reached to satisfy a business solution.
- **Business environment** connotes all factors external to the business company that greatly influence its functioning. It covers economic, social, political, natural, etc. factors.
- **Execution environment** represents the Information Technology (IT) where the service operates. It includes process engines, applications, servers, machines, network and their properties, etc.
- **Service implementation** includes technical and operational aspects. The technical aspects tell "what the service does" (e.g. interface, operation, input, output, pre-condition...) as well as "how to access to it" (e.g. binding, protocols...). Whereas, the operational aspects describe how the service works internally. They comprise functional and behavioral characteristics. They can be described imperatively (e.g. C, JAVA) or delaratively (e.g. rule-based: ECA-rule, ECAPE-L [17]). The service behavior can change by modifying the behavior of its operations or by changing their coordination.

1.1.2 Business process facets

Similarly to a service, a business process has business goals and an implementation including its operational aspects and technical aspects (Figure. 1.1). It operates within an execution environment and interacts with a business environment. The business goals of a process define the targets of a business enterprise such as increasing revenues or reducing losses. The operational aspects which represents the functional and behavioral aspects can also be written in imperative description by service compositions or processes such that some activities are implemented by services (e.g. BPEL [24], XPDL [28]) or declarative description based on rules.

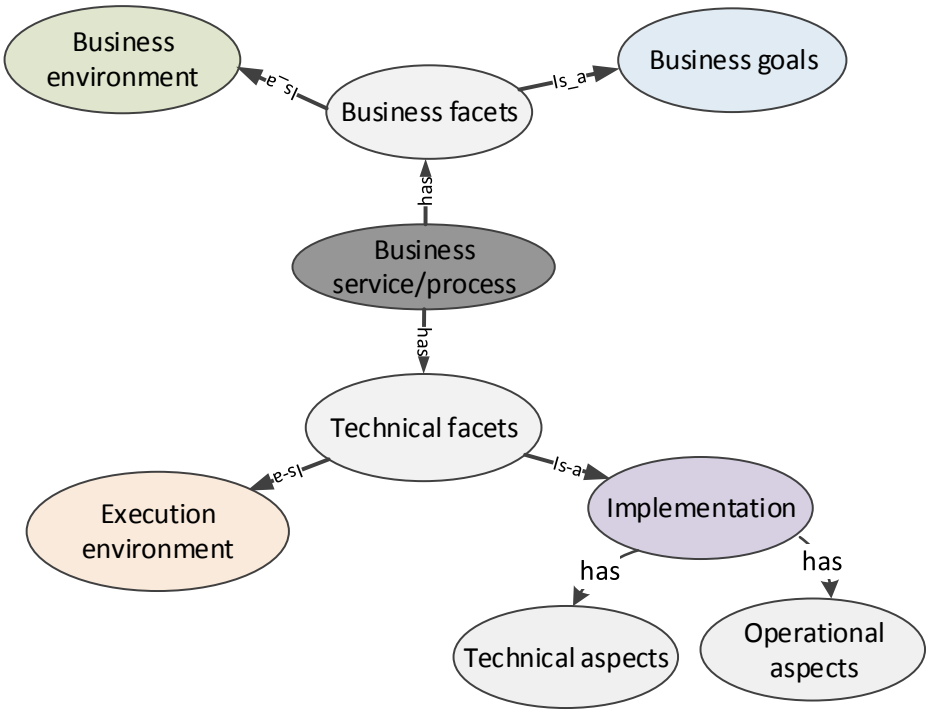


Figure 1.1: Business service/process facets.

1.1.2.1 Imperative description of business process operational aspects

The operation aspects of business processes are described using modeling languages. A first family of modeling languages for business processes consists in imperative languages (or procedural languages) which focus on how different process activities are performed (i.e. the execution scenario which represents an ordered sequence of activities is described in an explicit way). There are many imperative business process modeling languages. We distinguish high level process languages (e.g. BPMN, UML) and execution process languages (e.g. XPDL, BPEL).

High level business process languages

- ***Business Process Model and Notation (BPMN) standard*** Even though there is a large variety of modeling techniques, the most well-known is Business Process Model and Notation (BPMN). It is a standard with a graphical notation which provides companies with the capability of understanding their internal business procedures.

  The first version of BPMN was developed by BPMI (Business Process Man-

agement Initiation) in 2004 and then becomes maintained by OMG (Object Management Group) in 2006. The current version BPMN 2.0 was published in 2011. Its full specification is provided at [29].

- ***UML Activity diagram*** UML (Unified Modeling Language) is a standard proposed by OMG for graphical modeling used in object-oriented development. It has become a key language in software engineering since it expresses the needs and requirements using several diagrams (class diagram, activity diagram, etc.). UML is also used to model business processes using the activity diagram [30]. This latter has been revised to be more suitable for process modeling in UML 2.0.

Business process execution languages

- ***XML Process Definition Language (XPDL)*** is proposed by the *Workflow Management Coalition* which consists of many workflow system providers. XPDL specify business processes by defining activities, transitions, partners and interactions between them [31]. XPDL is adopted by many workflow engines.

- ***Business Process Execution Language (BPEL)*** is an OASIS standard executable language for specifying actions within business processes with web services [24]. BPEL messages are typically used to invoke remote services, orchestrate process execution and manage events and exceptions. Figure 1.2 illustrates a portion of BPEL purchase order process.

1.1.2.2 Declarative description of business process operational aspects

The operational aspects of business processes can be described declaratively such that a process is considered as a set of states and a set of constraints controlling transitions between states. In order to present these states and constraints, declarative languages use different paradigms.

Case-handling paradigm The case-handling is a new paradigm proposed by Van der Aalst et al. in [32] in order to model business processes flexibly. Contrary to the imperative modeling using predefined process control structures to identify what should be done, case handling focuses on what can be done to achieve a business goal. It is strongly based on data as the typical product of these processes (called cases) allowing designers to Execute, Skip or Redo process activities according to cases availability. The case- handling paradigm was adopted in the workflow management system called *FLOWer* [33].

```
<process name="PurchaseOrder">
...
 <onMessage partnerLink="ComercialService"

  <if>
   <Condition>
     Bpel: IsClientIsRegistered("ClientName") =  false
   <Condition>
   <Reply "Purchase order annulled"/>
  </if>

  <else>

   <invoke operation="requestBillPayment"/>

   <if>
    <Condition>
     Bpel: PaymentInCash()= true
    <Condition>
    <invoke operation="Save Payment Bill in casch"/>
   </if>

   <else>
   <invoke operation="Save Payment Bill in credit card"/>
   </else>

  </else>

 </onMessage>

</process>
```

Figure 1.2: A BPEL subprocess for purchase Order process [1].

Linear Temproral Logic (LTL) paradigm The Linear Tempral Logic can be used to model business processes declaratively. The LTL is a modal temporal logic with modalities referring to time. i.e. A proposition could be false at one point of time, then becomes true later on. This paradigm was used in the *ConDec* language propsed by Pesic et al. [34].

Deontic Logic paradigm The deontic logic can also be used to model business process in declarative manner. This logic formalizes the possible variants of business activity execution: the obligation, prohibition and permission. This will take into account obligations and permissions in business interactions. The deontic logic has been used in the *PENELOPE* language proposed by Goedertier et al. [35].

Event-Driven process paradigm The Event-Driven process paradigm is proposed to model and manage processes based on events by using the complex Event processing (CEP) formalism. This latter identifies the situations of a business process by selecting or aggregating events to launch the execution of a specific activity. This paradigm is used in the EPC (Event-driven Process Chain) method to specify business processes [36].

Rule-based modeling paradigm The rule-based modeling paradigm proposes to model the operational aspects by a set of rules using declarative languages. The Object Management Group (OMG) defines a business rule as *"a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. The business rules which concern the project are atomic that is, they cannot be broken down further."*[37].

According to [38], we distinguish four categories of business rules: *integrity*, *derivation*, *reaction* and *production*.

- ***Integrity rules*** describe constraints. They consist of a constraint modality and a constraint assertion, which is a sentence in some logical language [38].
- ***Derivation rules*** are also known as deduction rules. They express conditions that result in conclusions. These rules define the validity of facts and can be used to infer new facts based on known facts [39].
- ***Reaction rules*** are also known as Event-Condition-Action (ECA) rules, alternative-action rules, or post-conditions. They specify the event which occurs and express a specific situation. Then, the condition is evaluated and a subsequent of actions will react.
- ***Production rules*** consist of a condition and a produced action. They do not contain an event as the reaction rules.

There exist various rule modeling languages. Each one has its own expressive power. In the following, we shortly present these rule languages:

- ***REWERSE I1 Rule Markup Language (R2ML)*** The REWERSE I1 Rule Markup Language is developed by REWERSE Working Group I1 in 2007. It is a comprehensive and user-friendly XML rule format that allows: (i) interchanging rules between different systems and tools, (ii) enriching ontologies by rules, (iii) connecting rule system with R2ML-based tools for visualization, verbalization, verification and validation [40]. R2ML can support the four categories of rules which are integrity, derivation, reaction and production.

- ***Simple Rule Markup Language (SRML)*** The Simple Rule Markup Language was announced by ILOG in 2001. It describes a generic rule language consisting of a subset of language constructs common to the popular forward-chaining rule engines [41]. It does not use vendor-specific proprietary languages; therefore, rules can easily be translated to any other rule engine language. This makes it useful as an Interlingua for rule exchange between Java rule engines [41].

- ***Production Rule Representation (PRR)*** Production Rule Representation is an OMG standard published in 2002. As stated in [42], the goals of PRR are : (i) accelerating adoption of production rule components in everyday software systems, (ii) improving the modeling of production rules, especially with respect to UML, (iii) allowing interoperability across different vendors providing production rule implementations. The PRR supports production rules in a forward chaining inference engine.

- ***Semantics of Business Vocabulary and Business Rules (SBVR)*** Semantics of Business Vocabulary and Business Rules [43] is an adopted standard of the OMG. It provides a formal representation to business rules written in plain English or any other natural language. The SBVR specification is applicable to the domain of business vocabularies and business rules of all kinds of business activities in all kinds of organizations.

- ***Semantic Web Rule Language (SWRL)*** The Semantic Web Rule Language is a product of the World Wide Web Consortium (W3C). The SWRL supports the integrity rule. However, it is not yet standardized [44].

1.2 Running example

Service and Business process facets are illustrated here by considering a representative example of ClothStore (fictitious name), a clothing store holding an e-commerce site. ClothStore offers products to its customers, interacts with two suppliers and a shipper for processing orders. It holds certain products in stock, and orders others from suppliers in case of product lack.

The business process of the online shopping purchase order processing related to ClothStore is illustrated in the BPMN diagram shown in Fig. 1.3. This process behaves as follows: The customer sends a purchase order request with details about the required products and the needed quantity. Upon receipt of customer order, the seller checks product availability. If some of the products are not in stock, the alternative branch "ordering from suppliers" is executed. When all products are available, the choice of a shipper and the calculation of the initial price of the order

are launched. Afterwards, the shipping price and the retouch price are computed simultaneously. The total price is then computed in order to send invoice and deliver the products to the customer. Finally, a notification is received by the shipper assuring that the order is already delivered.

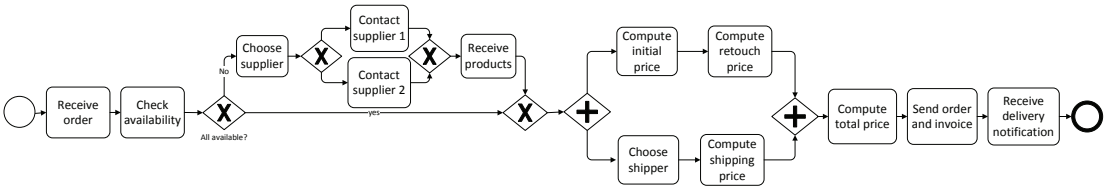


Figure 1.3: A BPMN Purchase order process.

The BPMN process shown in Fig. 1.3 represents an imperative **business process implementation** of the business process described above. It can be described declaratively based for instance on rules. For example, the activity "Check availability" can be described using an ECA-rule:
ON order Receipt
IF the order is valid
Do check availability

This latter business process of the ClothStore e-commerce site is running on top of three data center to store products and customers information. These data center represent the business process **execution environmen**t.

ClothStore fixes its revenues to be above 100.000 € per three months.In order to reach this **busines goal**, the ClothStore business process should be adapted in response to **business environment** changes. For example, during festive seasons, there is an increase in the demand for new clothes. Consequently, there is a need for much more servers and machines to execute the huge number of customers' requests.

Based on the service and business process facets illustrated above, we present in Section 1.3, the different types of management of SBPs.

1.3 Management of service-based business processes

Today, the maturation of Business Process Management (BPM) has enabled the linkage of business processes to IT services. Thereby, the management of a business process is highly relying on the management of its services.

As stated earlier, the management step consists of two main activities: monitoring and configuration. The monitoring aims at capturing and measuring required information. Subsequently, the configuration acts by setting or modifying some characteristics

based on the monitored information. There are four source facets to retrieve information and two target facets to act on (Figure 1.4). In fact, information can be retrieved from the execution environment, the business environment, business goals and operational aspects. In case of deficiency or inadequacy, the execution environment or the operational aspects should be reconfigured. Indeed, the configuration of a business process includes: substituting a service by another one, configuring a service behavior or configuring the process control flow (e.g. an operator of the service composition). According to the business service/process facets that are involved in the management, we distinguish three types of service-based business process management at run-time (Figure 1.4): technical management, business goals-based management and business environement-aware management.

In the following subsections, we detail these management types.

1.3.1 Technical Management

The Technical Management (TM) of SBPs aims at monitoring technical facets and configuring accordingly the execution environment in case of quality degradation (Figure 1.4). As stated in Section 1.1, the operational aspects of SBPs are based on Service Oriented Architecture (SOA) [45]. Since services constitute among others the building blocks of a SBP, the global QoS of a business process depends on the QoS of its services as well as their coordination. Indeed, TM is usually engaged in measuring non-functional (e.g. QoS related) properties [19] (Figure 1.4). For each service in the example introduced in Section 1.2, we can compute its response time, availability rate, etc. These characteristics are considered in the Service Level Agreement (SLA).

In case of SLA violation, the execution environment should be reconfigured. For instance, the operational aspects of a process can be configured by substituting the low quality service by a new one having the same functionality [9]. The goal is then to select the best service available at run-time, taking into consideration process constraints and the execution context [46]. Thereby, the adaptation in this case relies on service selection and binding.

When monitoring the process load, in case of degradation, IT managers can adopt an elasticity solution by reconfiguring virtual machines. In [47], Duong et al. propose to add or remove virtual machines on demand. In [48, 49], the authors propose to compute the optimal number of virtual machines to be deployed according to variations of demands. Authors in [50, 51] use duplication/consolidation mechanisms to provide elasticity of dynamic service deployment. Indeed, if the availability of a service is of low quality, the execution environment can deploy a new service replicate. While, the consolidation decision aims at removing an unnecessary copy of a service in order to meet its workload decrease.

Besides, in order to provide better quality, process instances can be migrated

from one site to another. Indeed, process instance migration consists in transferring a running process instance to another engine in order to continue process execution at another site if any issue affects the initial hosting site. This may help the resolution of execution problems when mobile devices change local contexts or when process resource requirements increase dynamically [52].

1.3.2 Business Goals-based Management

Business Goals-based Management (BGM) of SBPs is the configuration of their technical facets based on the monitoring of business goal metrics (Figure 1.4). Over the last decade, companies have increasingly search for managing their processes from a business perspective. In fact, a major concern for companies is to ensure the efficiency and performance of their business processes relatively to business goals. More precisely, it consists in aligning business goals with IT infrastructure and service/ business process quality [10, 20, 21].

For example, the business goal of a shipping service can be: "the number of undelivered orders may not exceed 2 per month". To meet this goal based on the business goal metric "undelivered orders ≤ 2", the availability of the shipping service should be 99%. Since 2003, the term "business oriented management" has been used to name aligning business goals with IT services [20]. Alternative used terms include: "Business-Driven IT management [10], "Business centric monitoring" [53], business/IT alignment, etc. In [54], authors present the challenge of business/IT alignment by presenting weaknesses of Service Level Agreement (SLA) in capturing the different service business needs. Thereby, Bratanis et al. [21] introduce the Business Level Agreement (BLA) which is a contract combining non-functional and functional characteristics of a service.

At business process level, process quality goals are specified in terms of Key Performance Indicator (KPI) [15, 16]. The used KPIs are key metrics (with technical or business meaning) with target values which are to be achieved in certain analysis period (e.g., order fulfillment lead time < 3 days). They are monitored using Business Activity Monitoring (BAM). In [55, 56, 57], authors propose a method to adapt processes by identifying influential factors of business process performance. Since KPIs potentially depend on numerous lower level PPMs (Process performnace metrics) [16] and QoS metrics, the adaptation action "substitute a service with another service" may be realized.

1.3.3 Business Environment-Aware Management

Business Environment-Aware Management (BEAM) of SBPs is the configuration of their technical facets based on the monitoring of business environment metrics (Figure 1.4). In fact, the competitiveness of business enterprises is deeply related to

adapting their processes against business environment changes [17, 18, 11, 4]. The business environment is dynamic in nature. It keeps on changing and differs from place to place according to social, political, technological... factors.

These changes have a direct impact on the service and process behaviors. Subsequently, it affects the operational aspects (Figure 1.4). For example, during sales promotions, there is a contestable and non-constant decrease of prices. In that case, the behavior of the service "calculate initial price" will change frequently since the discount rate can be modified constantly. Then, the purchase order process changes its behavior accordingly. Besides, during a festive season the execution path can change when choosing suppliers in response to the higher demand of clothes. Configurable services or business rules may be used to take into account these changes.

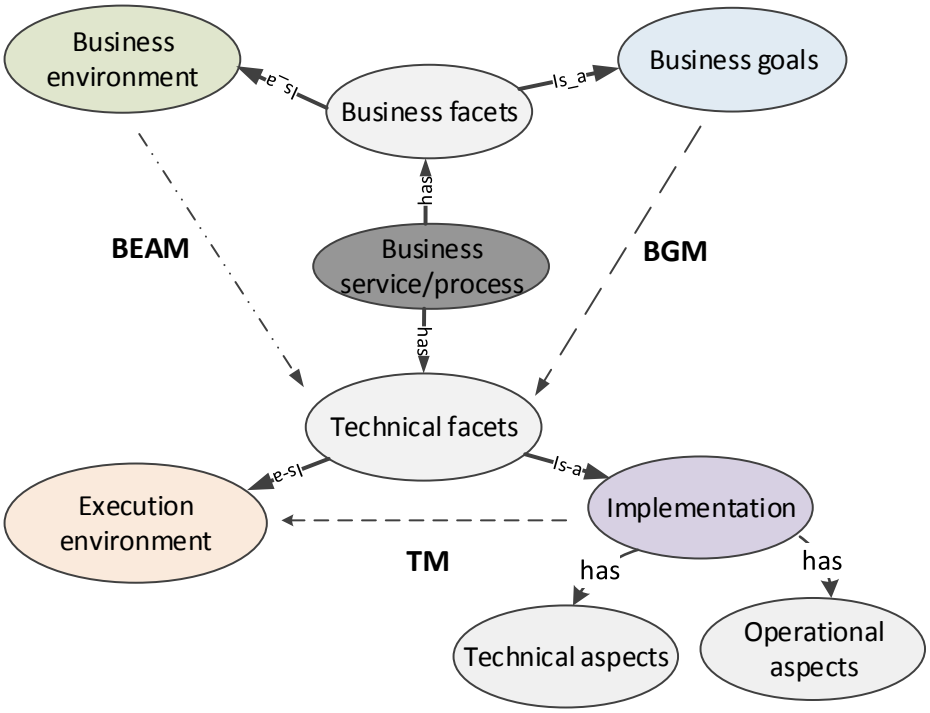


Figure 1.4: Business service/process management types.

Conclusion

In this chapter, we identified three types of business process management namely, technical management, business goals-based management and business environment-aware management. However, we notice that there was a great effort in adopting

technical management. There were also efforts in managing SBPs from a business point of view. Nevertheless, this type of management is effectively a business goals-based management, often called business/IT alignment.

Beside technical management and alignment of IT and business goals, we argue that there is also a great need to manage business processes from a business environment point of view (i.e. BEAM). Indeed, enterprises are more and more within highly competitive and constantly changing business environments. Thus, they need to manage their BPs to address competitions and changes within business environments.

In this thesis, we focus on BEAM of SBPs. More details about existing approaches of BEAM of SBPs are given in the following chapter.

Chapter 2

Business Environment-Aware Management approaches

Introduction

In this thesis, we are interested in BEAM of SBPs. As we stated in Section 1.3, the management of SBPs consists in two main activities: monitoring and configuration. The monitoring captures business events sent by the business environment, while the configuration alters operational aspects accordingly.

The operational aspects of business processes can be described by imperative processes (e.g. BPMN, BPEL) (see Section 1.1.2.1). They can also be described declaratively based on case-handling paradigm, Linear Temporal Logic paradigm, deontic logic paradigm, event-driven process paradigm or rule-based modeling paradigm (see Section 1.1.2.2). In this thesis, we focus on declarative descriptions based on rules.

The configuration of a business process can also be described using rules, imperative processes or both of them. The management of a business process (called later-on managed process) can be accordingly defined by three different approaches: declarative, imperative or hybrid.

The imperative approach describes both the managed process and its configuration based on an imperative process. Regarding the declarative approach, the managed process and its configuration are described based on rules. As for hybrid approaches, they integrate imperative and declarative approaches. Most hybrid approaches integrate imperative business processes and business rules [11]. The managed process and its configuration are respectively described based on an imperative process and rules.

In the following sections, we present these management approaches illustrated by the running example. Besides, we review existing businesss environment-aware management approaches.

2.1 Types of business environment-aware management approaches

2.1.1 The imperative approach

In the imperative approach, the operational aspects (Figure 1.1) of the managed process are described based on an imperative process (Section 1.1.2). The configuration is described as an imperative process, too. Indeed, it consists in adding a set of process fragments to the managed process. Hence, all the configurations are modeled within the managed process as process fragments described imperatively.

In imperative approaches, managed processes and their configuration are described using process languages such as BPEL, XPDL and BPMN.

Coming back to the running example, the configurations of the purchase order process are shown in bold in Figure 2.1. The operational aspects of the managed process represent the rest of activities. Indeed, in case of sales promotions, the configuration represents the fragment consisting of five activities computing the discount rate delimited with two gateways: Event based gateway and exclusive gateway.

2.1.2 The declarative approach

In declarative approaches, the operational aspects of the managed process are described declaratively (rule-based). Alike, its configuration is described declaratively based on rules. In fact, the configuration consists in adding, modifying or deleting a rule in case of business environment change. The declarative approaches are then purely based on rules.

In declarative approaches configuration may be described using many rule modeling languages such as Semantics of Business Vocabulary and Business Rules [43], Simple Rule Markup Language [41], Rewerse l1 Rule Markup Language [40], ECAPE-L [58], etc.

Resuming with the running example reported in Section 1.2, Figure 2.2 shows the activities of the purchase order process modeled based on rules according to the ECAPE model of the ECAPE-L rule language [58]. For example, Rule 1 expresses the policy of receiving an order. This rule is activated by "begin process" event that represents customer order. The execution of "receive order" activity triggers "order received" event. The latter will activate Rule 2 that expresses the policy of checking availability (Figure 2.2). Hence, the activities "Receive order" and "check availability" of the purchase order process are implemented by Rule 1 and Rule 2. During a sales promotion, a discount rule (Rule 13) is added and the relationships with the existing rules are established (Figure 2.2).

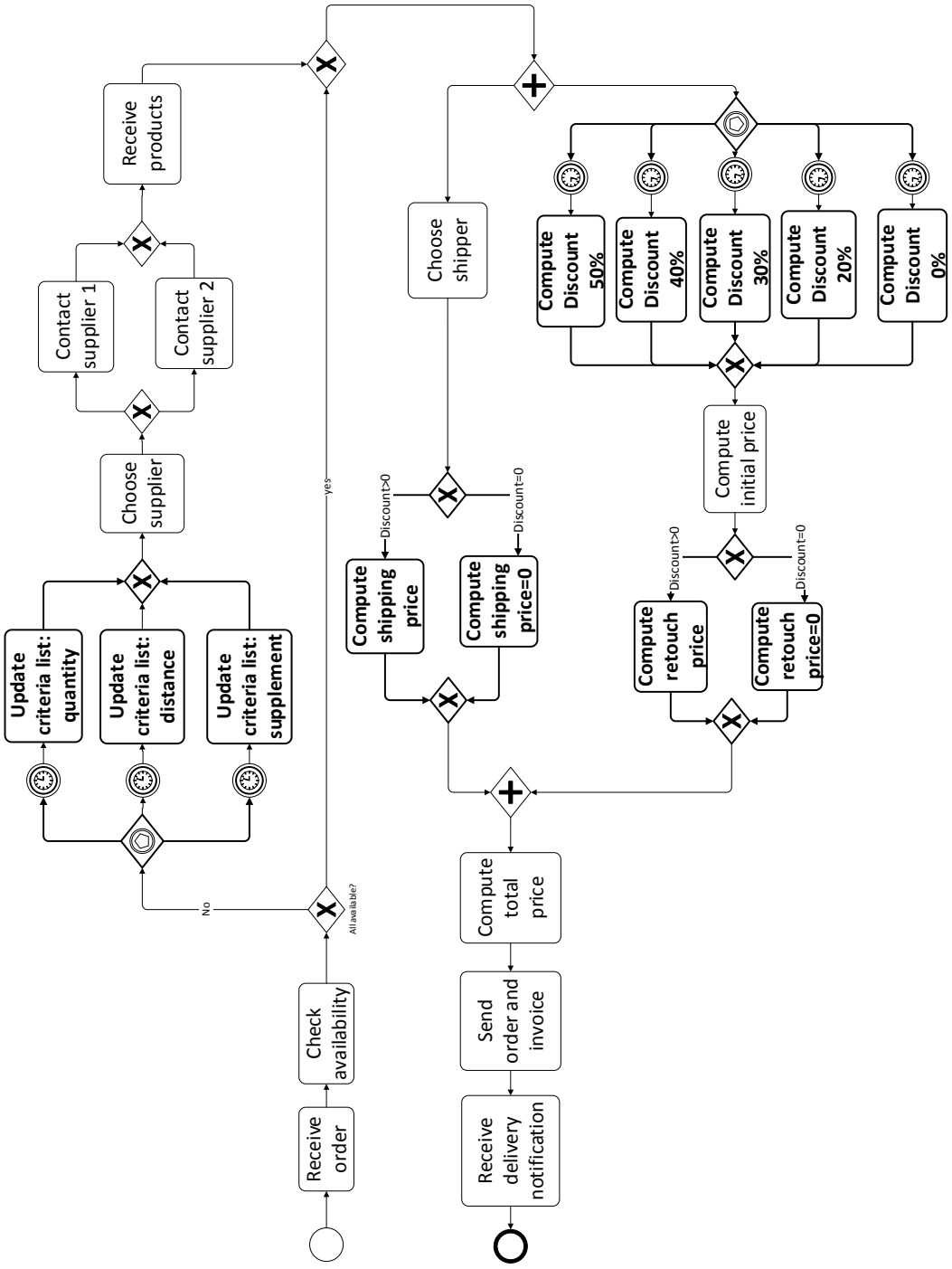


Figure 2.1: Imperative approach for managing purchase order process.

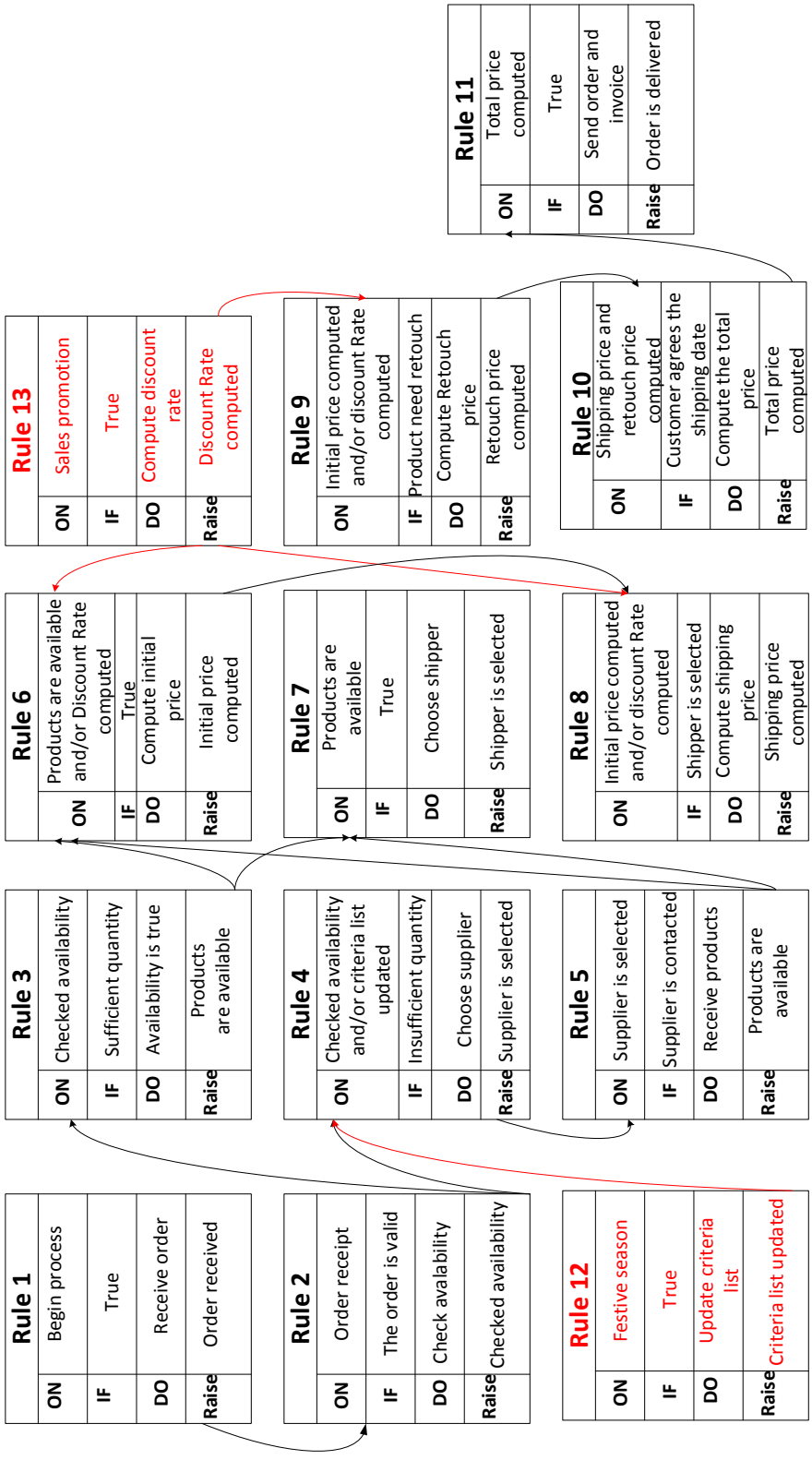


Figure 2.2: Declarative approach for managing purchase order process.

2.1.3 The hybrid approach

Generally speaking, hybrid approaches combine declarative and imperative approaches. Indeed, the operational aspects are described as an imperative process where some activities are described declaratively based on rules.

The configuration of the managed process is then based on rules that are integrated in some activities (called business rule tasks). Hence, the configuration consists in adding, modifying or deleting business rules in business rule tasks.

The managed process is then described using for example BPMN 2.0, which include business rule task.

Figure 2.3 shows the purchase order process written in BPMN 2.0 such that certain activities are described declaratively based on business rule tasks. Indeed, instead of implementing five activities for computing the discount rate in the imperative approach or adding a discount rule in the declarative approach, the activity "compute initial price" is performed by business rule task invoking a discount rule (Rule 2).

2.2 Review of existing business environment-aware management approaches

As today's business environments keeps changing, there is a need for business processes to be adaptive yet competitive. Unfortunately, the static nature of business process definitions (imperative description) makes it impossible to configures them at run-time and the redeployment of a modified process is required. On the other hand, all the rule based approaches apply pure rule based methodology (declarative approches) which can be time-consuming.

Therefore, different approaches try to add a dimension of flexibility while keeping efficiency. This is done by using rules, variability, etc.

2.2.1 Rule-based approaches

Different approaches [18, 5, 11, 4] try to integrate these two techniques in a joint approach by separating business logic (described by Business rules) and process logic (described by imperative BP). This separation needs in turn an effort of BP and BR integration.

Charfi et al. [25] focus on Aspect Oriented Programming(AOP) in order to integrate Business rules and the process logic at run-time. Indeed, the business rules are proposed to be implemented in an aspect-oriented extension of BPEL called AO4BPEL. AO4BPEL is used to weave the adaptation aspects into the process at run-time. Although they preserve BP standards, the weaving phase can strongly limit the process efficiency at run-time since it can raise issues on maintainability

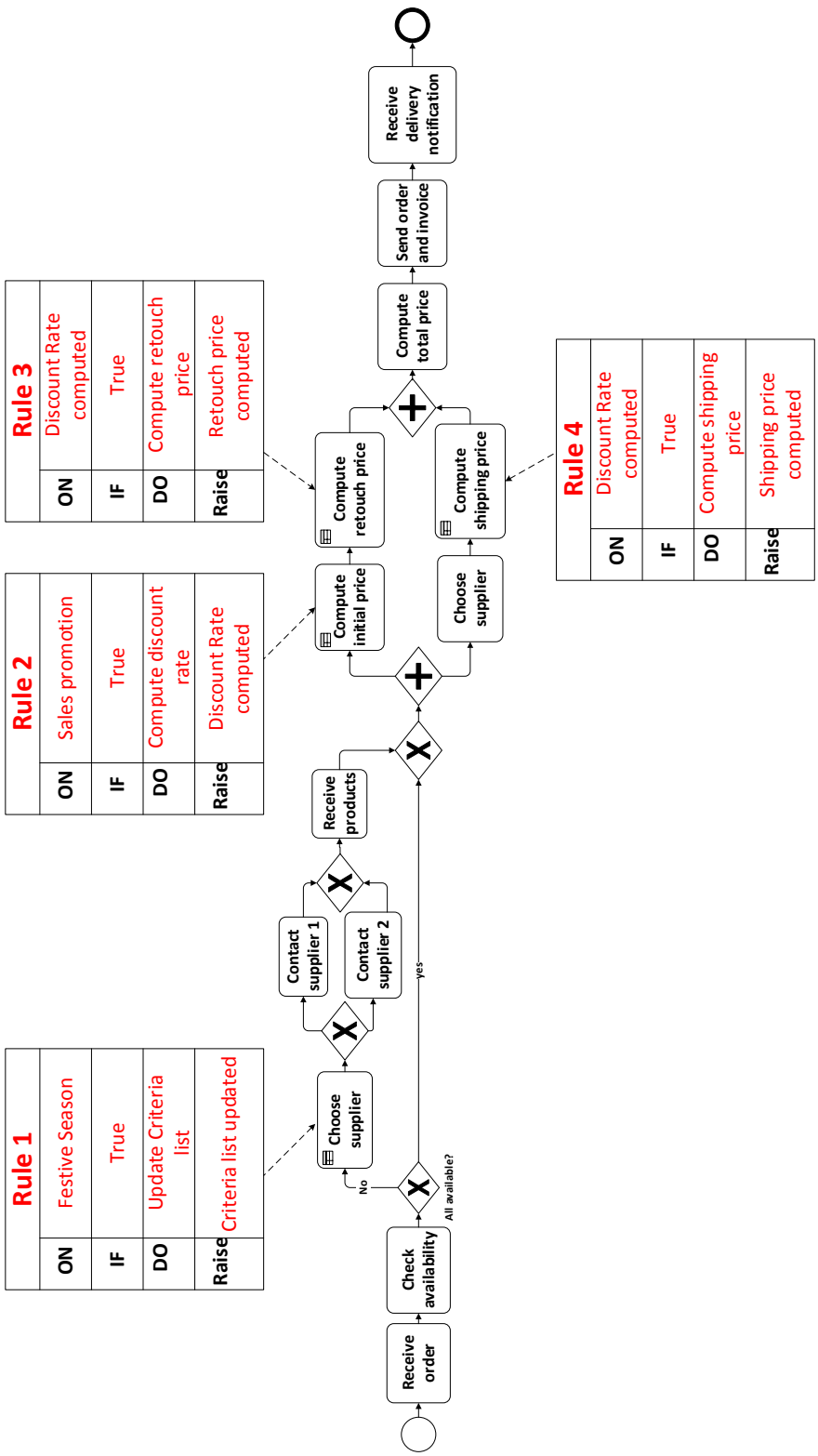


Figure 2.3: Hybrid approach for managing purchase order process based on business rules.

and transformation. The business rules actions and results are translated to business process constructs and to so-called "point-cuts" (statements to relate the aspect to specific points in the code such as every assign activity). This requires a modified BPEL engine to be able to cope with the additional aspects.

Ouyang et al. [11] introduce an ECA-based control-rule formalism to modularize the monolithic BPEL process structure. Only one classification of rules is defined that handle the control flow part of the composition linking activities together. In this work, the designer should also take into account the defined ECA-control rule and specifies his process accordingly.

Authors in [2], propose a hybrid solution presenting a modeling language that integrates both rule- and process-oriented modeling perspectives. As shown in Figure 2.4, the language (Rule-based BPMN –rBPMN) is based on the integration of the Business Process Modeling Notation with the REWERSE Rule Markup Language (R2ML). The integration is based on the RuleGateway. This element is an extension of the BPMN Gateway class and refers to one or more R2ML Rules. Thus, an R2ML Rule can be placed into a process as a Gateway. Each rBPMN rule gateway might be associated with more than one rule. In [59], authors use the rule patterns of this new language to create flexible processes: control flow decision pattern, data constraints and dynamic business process composition. Although this approach provides a dimension of flexibility and efficiency, proposing a new language remains a handicap for companies that use existing business process standards.

In [3], Cheng et al. provided a bottom-up approach to integrate process models and business rules models in both design and analysis stages (Figure. 2.5). This approach is specific to integrate BPMN process models with SBVR rules models. It used XPDL to translate the BPMN diagrams to text representation and then used these tags to map the business rules models, and finally producing a new model that includes both the process and the rules. The approach did not invent a new language. It made use of XPDL and its ability to translate BPMN diagrams into XML tags. The main contribution is providing a list of the main components of the process language (BPMN) and the rules language (SBVR), and using XPDL to map these components to each other. Nevertheless, the mapping phase might be time-consuming at run-time.

In [60], Zoet et al. aligns business process management with business rules. Indeed, they propose business rule categorization that is aligned to the business process management lifecycle. BRM formulates constraints based on descriptions and facts while BPM addresses business operations from an activity approach. In this paper, authors propose to synchronize them based on rule categorization. They distinguish between: structural sequencing rule, Actor Inclusion rule, Outcome control rule.

Gong and Janssen introduce in [4] a combination between semantic services and business rules in order to manage business processes at run-time. They create business processes dynamically (Figure 2.7). Rules are mapped into decision services which are

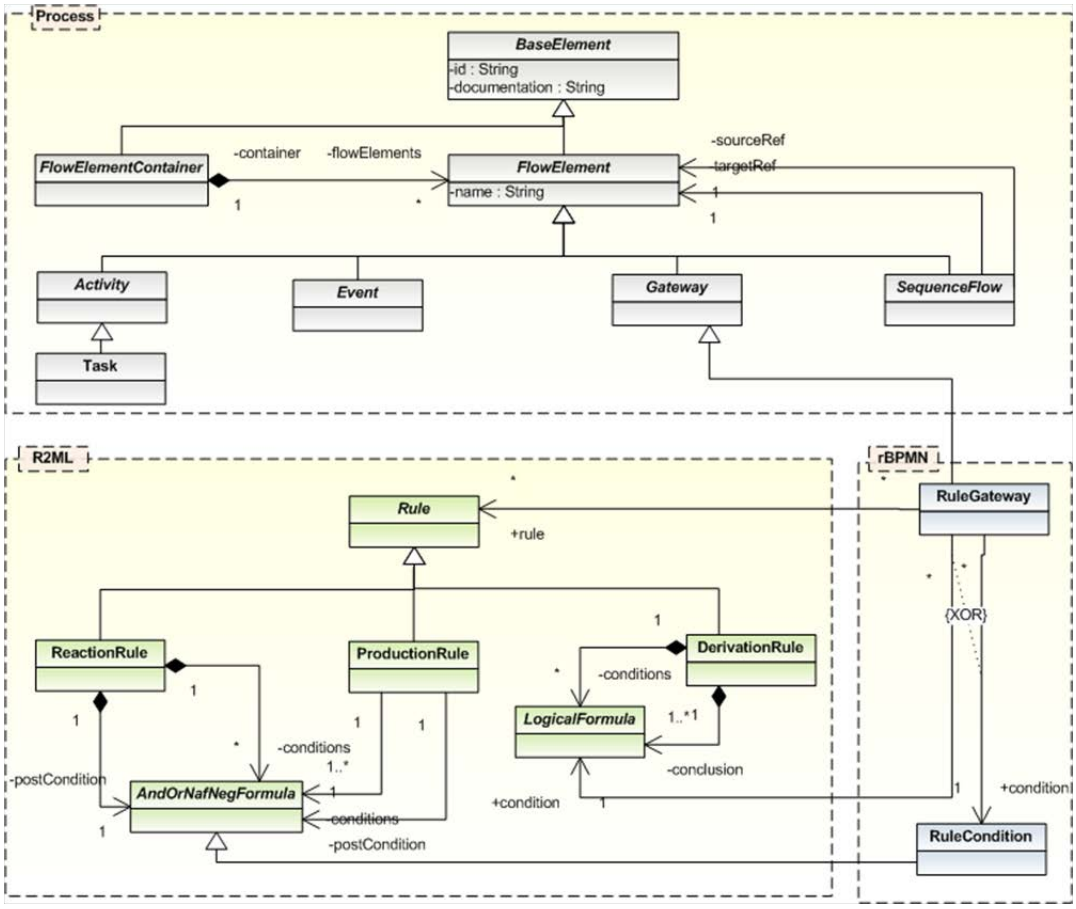


Figure 2.4: rBPMN meta-model [2].

orchestrated with the managed business process. This approach increases the flexibility at run-time while the process efficiency decreases since services are composed on the fly based on a domain ontology.

2.2.2 Variability-based approaches

Rule-based management/adaptation is not the only mechanism to provide flexibility in a process-aware information system.

The service based business process can also be adapted by explicit variability modeling [46].

Variability represents the key concepts of product-line technology which can be used to make service-oriented applications more flexible. It allows among others runtime flexibility.

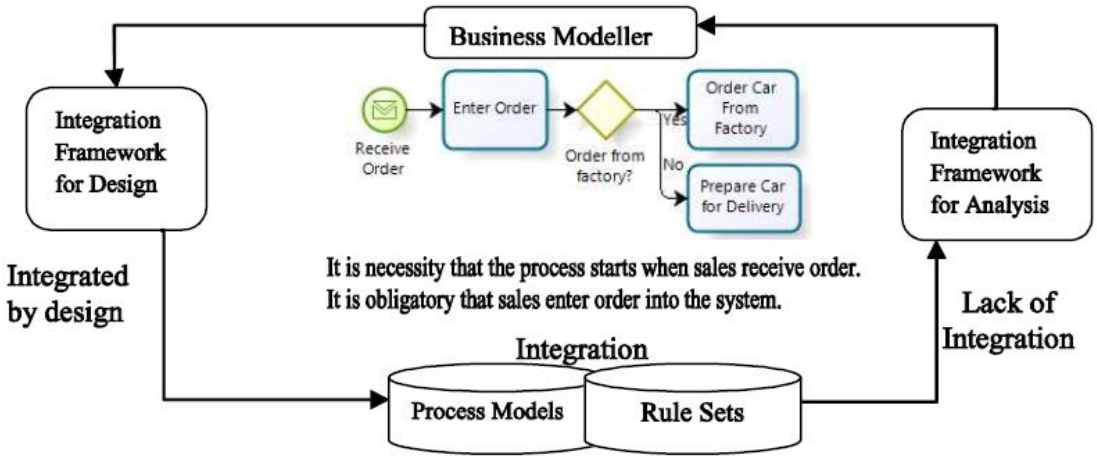


Figure 2.5: Integration Framework for design and analysis [3].

Authors in [61] distinguish two categories of variability in service based systems: variability inside a service and variability in the service based architecture (i.e. the composition of services / service based business processes).

The first category focuses on the variability inside a service with services as reconfigurable units that can be adapted for different contexts. Going into more details with variability inside a service, Galster et al. identify two main types: (1) variability in parameters required by a service and (2) variability in parameter values. The first type consists in varying the type of data sent at service invocation. For example, data sent to a service might be a single variable or an array of variables. This type of variation is usually expressed in Web Service Definition Language (WSDL) documents. The second type consists in varying the value of a parameter used at invocation. For example, the discount rate of a product differs from one sales season to another.

The second category, variability in the service based architecture, consists of three main types: logic variability, variability in the web service flow, and composition variability.

Authors in [5], present an adaptation of BPEL language called VxBPEL. They emphasize on the lack of flexibility and variability when deploying BPEL processes. Thus, they propose to extend BPEL language by adding `Variation Points` and `Variants` (Figure. 2.8). The former represents the places where the process can be configured, while the latter defines the alternative steps of the process that can be used. According to the running example, the VxBPEL fragment shown in Figure 2.8 could contain a `Variation Points name="Discount Rate"`. This variation point shows variability in the way dicount rate is determined. Variants for this variation

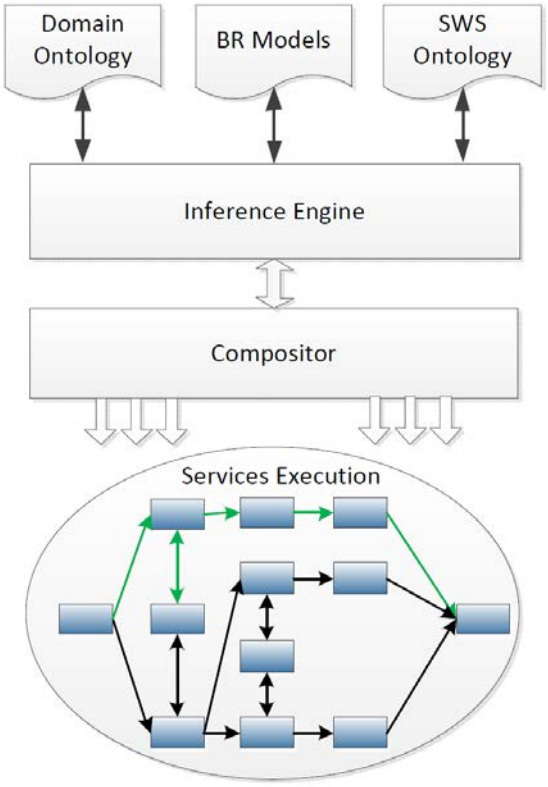


Figure 2.6: Architecture for integrating BRs and SWSs [4].

point are `Variants name="Fisrt Discount"` and `Variants name="Second Discount"`. Based on the selected variant, either a service for First discount is envoked or the second Discount service is processing. In this work, the variability is focused on BP aspects written in VxBPEL language. The designers should consider this extension and add their variation when designing the BP. VxBPEL supports service replacement, different service parameters, and changing system composition.

Other approaches, such as [18, 5], address management issue by process variants. When modeling process and their variants, one has to decide which control flow alternatives are variant-specific and which ones are common for all process variants. However, these process variants ought to be configured at configuration time which leads to a static instance of the process model at run-time.

2.2.3 Synthesis

Table 2.1 summarizes the surveyed approaches and derives the following synthesis. The defined criteria are rated as high level support (++), support (+), partial sup-

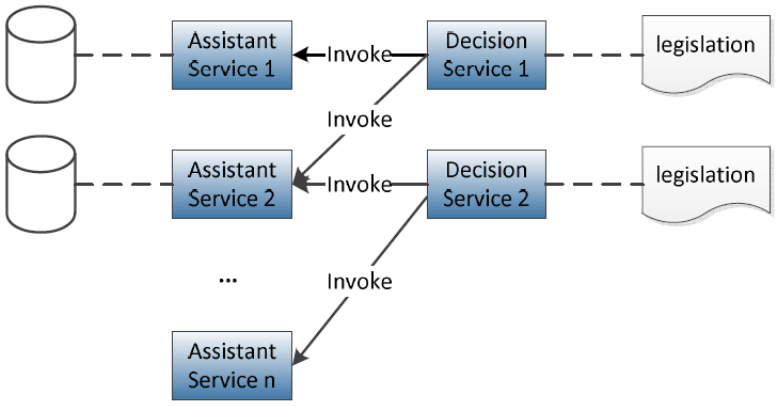


Figure 2.7: Relationship between decision services and assistant services [4].

port (+/-) and no support(-). We notice that efficiency and flexibility almost in all approaches in the table are antagonistic. In fact, when increasing the flexibility, the efficiency decreases accordingly and vice versa. Furthermore, none of approaches presented in [17, 18, 4] utilizes BPEL or any other process standard, which renders them difficult to use.

For this sake, we recall the following challenges addressed in this thesis:

- How to conciliate between imperative and declarative techniques in an integrated hybrid approach while aligning operational and business perspectives?
- How to develop a hybrid management approach that preserves industry standards to describe and systems to manage SBPs?
- How to minimize designers efforts?

Conclusion

As business environment changes keep increasing, enterprises are always seeking for a balanced solution to manage their BPs. However, most research works have focused on efficiency or flexibility using either imperative or declarative techniques. Therefore, different approaches [18, 5, 11, 4] try to integrate these two techniques in a joint approach by separating business logic (described by Business rules) and process logic (described by imperative BP). In this chapter, we presented these different business environment aware management approaches. We showed that existing approaches are efficient or flexible. We also presented the differences between current approaches and our targeted approach.

```
<vxbpel:VariationPoint name="VP1">
  <vxbpel:Variants>
    <vxbpel:Variant name="default">
      <vxbpel:VPBpelCode>
        <invoke .../>
      </vxbpel:VPBpelCode>
    </vxbpel:Variant>
    <vxbpel:Variant name="alternative1">
      <vxbpel:VPBpelCode>
        <sequence ...>
          ...
        </sequence>
      </vxbpel:VPBpelCode>
    </vxbpel:Variant>
  </vxbpel:Variants>
</vxbpel:VariationPoint>
```

Figure 2.8: Definition of a variation point in VxBPEL [5].

In the next chapter 3, we introduce an overview of our approch consisting in generating a management process connected to the managed process to be monitored and configured. Besides, we present a semantic model which facilitates the management process generation.

Approches	**Management phases and mechanisms**			**Flexibility**	**Efficiency**	**Preserving standards**
	Design time	**Deployment time**	**Run time**			
Rule-based configuration						
Charfi et al (2009)	*AO4BPEL *Modify BPEL engine		weaveing overhead	+	-	-
Cheng et al. (2011)			XPDL: Mapping overhead between BPMN & SBVR	+	-	+
Milanovic et al (2011)	proposing rBPMN: BPMN & R2ML			+	+	-
Ouyang et al. (2011)	Control rule for BPEL			-	+	+
Gong and Janssen (2011)	Describing BR, semantic services		Creating dynamic BP	++	-	+
Variability-based configuration						
Gottschalk et al (2008)	Specifying variants and configure them		static processes	-	+	-
Koning et al (2009)	proposing VxBPEL, variation points, variants			+	-	+

Table 2.1: Review of existing business environment aware management approaches

Part II

Contributions

Chapter 3

A hybrid approach for business environment-aware management of service-based business processes

Introduction

This chapter presents our contributions for business environment-aware management of service-based business processes (SBPs). The main idea is to manage SBPs against business environment changes, while conciliating imperative and declarative approaches, preserving business process standards and minimizing designers efforts. Only a SBP and its corresponding domain ontology are the input of our contributions. We do not ask process designers to re-model their processes. Hence, our approach is based essentially on semantic descriptions.

In this chapter, we firsltly present our approach overview. Then, we detail our propsed semantic model. The work of this chapter has been published in [63].

3.1 Approach overview

In our research work, we focus on SBPs where some activities are realized by services. We consider that the management of a composition of services that offer management operations (for their monitoring and configuration) can be realized through the composition of the offered management operations [63]. The enactments of management operations are triggered by events that are captured from the business environment. The integration of management operations and business environment events forms a business process called the management process.

Indeed, for a given managed process, our approach generates the corresponding management process. Figure 3.1 illustrates the purchase order process and its corresponding generated management process.

In fact, to take into account the business environment changes into the managed process, we use service properties that can be adjusted. Service properties allow for the configuration of an implementation with externally set values. The value for a service property is supplied to the implementation of the service each time the implementation is executed. In particular, the internal value of a property can be altered at any time through business management operations which enable its monitoring and configuration. The monitoring step reads properties while the configuration one updates them if necessary. When altering the property value, the corresponding service changes its behavior. For example, the service "Compute initial price" (Figure 3.1) has a property named "DiscountRate" which can change its behavior by a setter operation when a sales promotion is triggered. Besides, during a festive season the execution path can vary when choosing suppliers in response to the higher demand of clothes. In fact, the service "Choose supplier" changes its behavior by updating its property "CriteriaList".

As we already mentioned, when changing a service behavior, the corresponding business process is reconfigured and its behavior is accordingly modified.

Thus, the first step towards the automation of the management operations composition is to identify the semantic concepts of service properties from the managed business process. The issue is how to modify these properties and to follow which order to compose management operations. To deal with this issue, we consider three statements:

- **Statement 1:** Events represent the glue between business environments and SBPs. Events may trigger the update of service properties. Hence, there is an Environment-Service relationship.

- **Statement 2:** Service properties may depend on each others. Accordingly, modifying a service property may engender changes of others depending on it. Consequently, there is a Service-Service relationship.

- **Statement 3:** The structure of the managed business process may guide the order of management operations that modify properties.

Based on these statements, we propose an algorithm for generating the management process. This latter is based on three main phases: (1) constructing sub-processes based on Environment-Service relationships, (2) constructing sub-processes based on Service-Service relationships, (3) connecting the resulting sub-processes into the management process based on the structure of the managed business process.

As shown in Figure 3.1, the first phase builds sub-processes illustrated with dashed rectangles. The second one generates sub-processes represented in dashed ellipses. The third phase adds operators and connections shown in bold.

In order to facilitate the management process generation, there is a need for (1) an appropriate semantic model of business environments, business processes and relationships between them as well as (2) a dependency analysis of the managed business process. The semantic model represents an upper management ontology, which correlated with a domain ontology, depicts a declarative description of the company management strategy against dynamic change of the business environment. It involves Environment-Service and Service-Service dependencies. In addition, the dependency analysis aims at describing explicitly control dependencies between services of the managed business process in order to deduce the composition of their corresponding management operations [64].

More details for the proposed semantic model and the dependency analysis of the managed process are given respectively in Section 3.2 and Chapter 4.

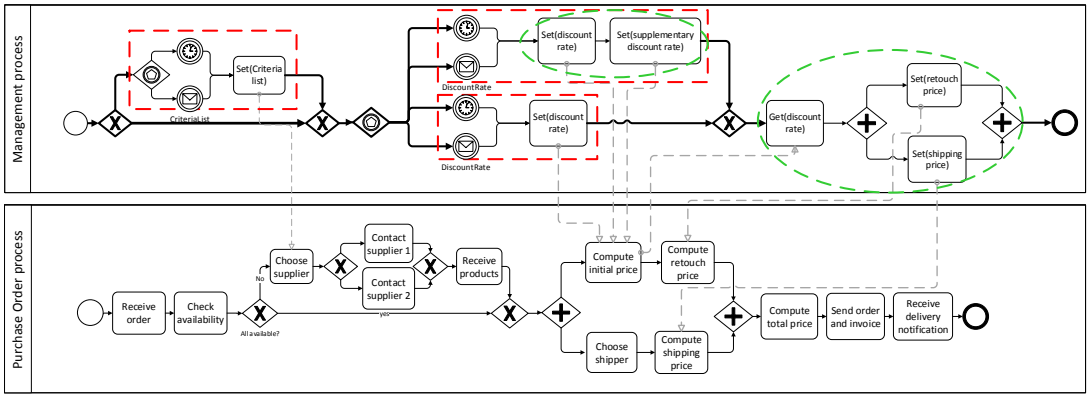


Figure 3.1: Purchase order process with its corresponding management process.

3.2 Semantic modeling of service-based business processes and business environment

The management process generation requires semantic descriptions of business environment, SBPs and relationships holding between them.

Indeed, as shown in Figure 3.2, there are three main actors in BEAM: business environment, business process and services. The business process has a service composition which consists of activities and gateways. Activities are realized by services. Services may have service properties and management operations. Services interact with the business environment. This latter engenders **events** that trigger **management operations** which act in turn on **service properties**. These three concepts represented in grey ellipses in Figure 3.2 describe, at a high level of abstraction, the

main concepts of the management ontology.

In the following subsections, we start by depicting the upper management ontology (section 3.2.1). Then, we describe the SBP by giving a semantic service model (section 3.2.2).

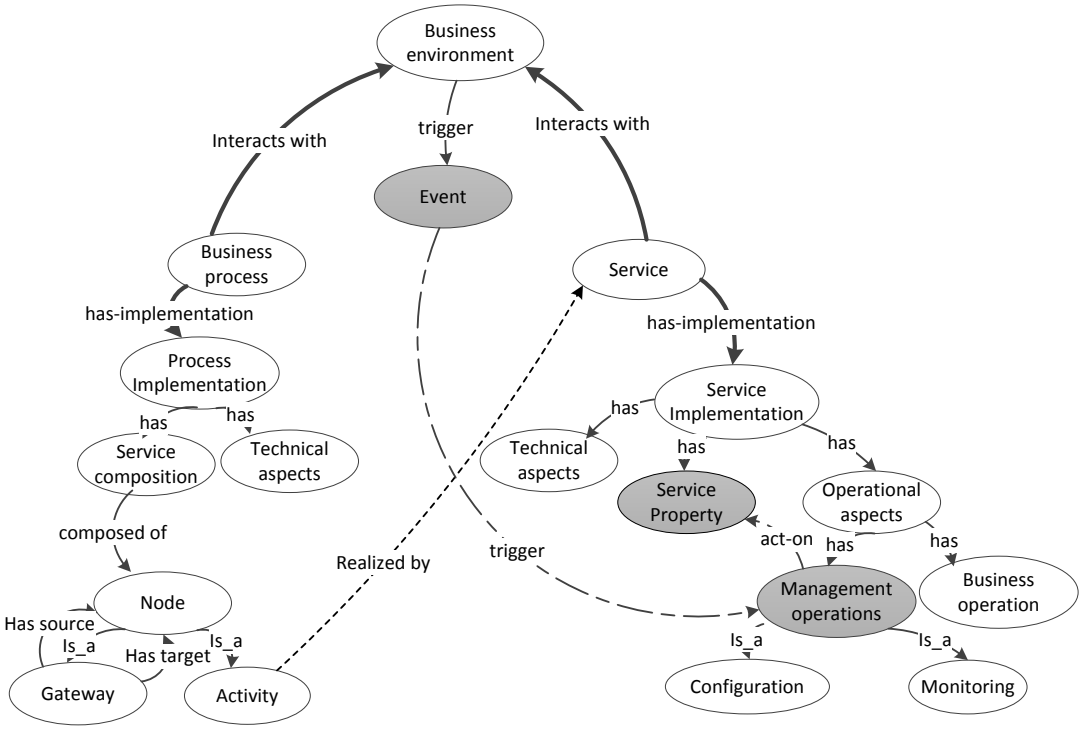


Figure 3.2: BEAM meta-model.

3.2.1 Upper management ontology

Based on the BEAM meta-model (Figure 3.2), we define an upper management ontology (UMO) correlated with a domain ontology (Figure 3.3). Top-level ontologies or Upper ontologies are models of the common objects that are generally applicable across a wide range of domain ontologies. It contains a core glossary in whose terms objects can be described. A domain ontology (or domain-specific ontology) models a specific business domain, or part of the world (e.g. e-commerce domain, medical domain, etc.). It represents the particular meanings of terms as they apply to that domain.

The UMO correlated with the domain ontology, is used for annotating semantic services and facilitating the management process generation as well. The UMO main

concepts are *events*, *properties* and *management operations*. They represent an ontological modeling of the relationships between business environment, processes and services. They are described against a domain ontology defined by domain experts. The UMO includes also two main relationships:

- **Environment-Service relationship (Event-based relationship):** Events trigger Actions (management operations) which act on services properties (e.g. the "Festive Season" event triggers "SetCriteriaList" operation which updates the "CriteriaList" property).
- **Service-Service relationship (Data-based relationship):** Each service property has service properties that may depend on it (e.g. "ShippingPrice" and "RetouchPrice" service properties depend on "DiscountRate" property).

In the following, we detail the main concepts of the upper management ontology as well as their relationships.

3.2.1.1 Events

Events play a prominent role in BEAM, since they are the glue between situations in the real world and SBPs. Thus, in the following, we detail event semantics and definitions based on the expressiveness of BPMN 2.0 [23]. Events are used to model something happening in the process lifetime (e.g. festive seasons, sales promotions). They affect the flow of the process by catching a trigger or throwing a result. Event definitions represent the semantics of events. In BPMN 2.0, there are 10 event definitions among them we use: Message, Signal, Timer and Conditional.

For instance, refering to Figure 3.3, the event "FirstSalesPromotion" is composed of two atomic events having respectively timer and message event definition. The timer event "PromotionTimeDate" is used to detect that the promotion time date is reached. While, the message event "DiscountRateMessage" is used to define a discount rate message in order to catch the discount rate information.

Alike, the event "FestiveSeason" consists of two atomic events having message and timer event definitions. The timer event "FestiveTimeDate" catches the beginning of the festive season. The message event "CriteriaListMessage" catches the criteria list to choose the adequate supplier.

3.2.1.2 Service Properties

A single business service operation can have different behaviors depending on the business context (e.g. the service "choose supplier" selects supplier 1 or supplier 2 according to the existence or not of a festive season (Figure 3.1)). The more business contexts increases, the more the variability of the service operation is crucial. When

specifying service operations, it is necessary to define the various contexts of use to model variabilities. In this work, we define service properties which represent the variability of services composing business processes. Once set, the operation is able to take account of its business context to adapt its behavior.

Regarding the Service-Service relationship, services properties may or not depend on each other . For example, both "ShippingPrice" and "RetouchPrice" service properties depend on the "DiscountRate" service property. Whereas, "ShippingPrice" and "RetouchPrice" do not depend on each other (Figure 3.3). In case of dependency, we can distinguish three types of relationships namely: sequence, mutuality and exclusivity. For instance, a "SupplementaryDiscountRate" property is applied sequencially after the "DiscountRate" property (Figure 3.3). A "DiscountRate" property and a "LoyalDiscount" property are exclusive and can not be combined in a public sales promotion. These relationships can be identified from the ontology through RDF containers such as RDF:Seq, RDF:Bag and RDF:Alt.

3.2.1.3 Management operations

Management operations represent actions triggered by business environment events to alter or read service properties. They are means to perform the configuration and the monitoring of services and business processes in turn. The management operations are given by the service provider since they are related to the business of the service. However, we can help the service provider to generate these operations through semantic annotations. Nevertheless, in this thesis, we are limited to setter and getter operations generated automatically.

The relationships with business operations represent the management operation types:

- **Transformation (Configuration):** The management operation can configure a service by updating the property (setter: set operation) (e.g. setDiscountRate, setCriteriaList (Figure 3.3 and Figure 3.1)).

- **Consultation (Monitoring):** The management operation can also monitor the service property (getter: get operation (e.g. getDiscountRate (Figure 3.3 and Figure 3.1)).

3.2.2 Semantic modeling of service-based business processes

Services represent the building block of SBPs. Hence, we start by presenting the service model. Then, we introduce the business process model.

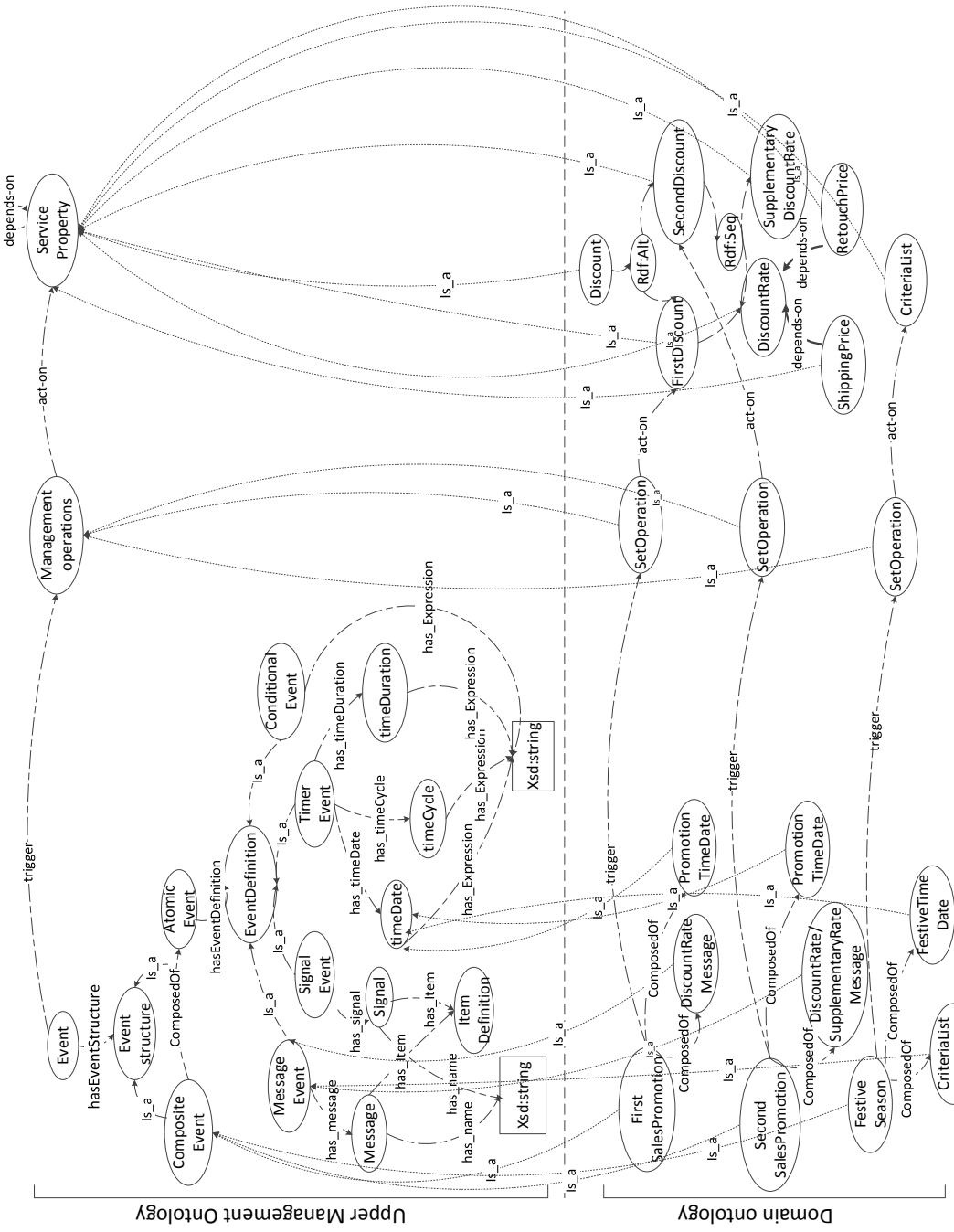


Figure 3.3: Upper management ontology.

3.2.2.1 Semantic Service model

A service has an implementation including its property, its technical aspects, and operational aspects (Figure 3.2). In this work, a service S is mainly characterized by its property p, which, being adjusted, changes the service behavior. A service property has a name, a value, and is annotated with a concept from the domain ontology. The technical aspects TA represent the set of: Inputs I, Outputs O, Pre-conditions Pre, Post-conditions $Post$, Assumptions A and Effects E. The operational aspects OA comprise the business operation F_p and its corresponding management operations M. We distinguish two types of management operations: monitoring operations M_m and configuration operations M_c. Furthermore, among management operations, we generate automatically a setter and a getter for each service property. The other management operations are set by the service provider. Thereby, changing the service behaviour can be made by changing the value of its property p through management operations which changes the behaviour of the business operation F_p (e.g. $F_{p=v_1}...$). Accordingly, a service is defined as follows:

Definition 1. Semantic Service Model. *We refer to each service by a tuple* $S = (p, TA, OA)$ *such that:*

- $p = (name, value, concept)$ *is the service property*
- $TA = (I, O, Pre, Post, A, E)$ *are technical aspects* where:
  - I: set of Inputs
  - O: set of Outputs
  - Pre: set of Pre-conditions
  - Post: set of Post-conditions
  - A: set of Assumptions
  - E: set of Effects
- $OA = (F_p, M)$ *connotes operational aspects*
  - F_p *represents the business operation of* S
    $F_p : A \times I \times Pre \longrightarrow^p O \times Post \times E$
  - $M = (M_m, M_c)$ *is the set of managing operations*

3.2.2.2 Service-based business process model

A SBP is represented by a set of activities, gateways and possibly events. Certain activities are realized by semantic services. The service composition of a SBP may be described using a business process standard (e.g. Event-driven Process chains (EPC),

Business Process Execution Language (BPEL), Business Process Modeling Notation (BPMN), etc). Hence, we define a SBP as a process graph (Definition 2). Definition 2 is inspired from the business process graph definition given in [65].

Definition 2. Process graph. *Let Γ_1 be a set of node types. Let Θ_1 be a set of node labels. A process graph PG is represented by a tuple $PG = (V_1, E_1, \tau_1, \theta_1)$ where:*

- *V_1 is a set of vertices*
- *E_1 is a set of edges modeling the control flow of the business process*
- *$\tau_1 : V_1 \rightarrow \Gamma_1$ is a function that maps vertices to types.*
- *$\theta_1 : V_1 \rightarrow \Theta_1$ is a function mapping vertices to labels comprising <(vertex type,vertex name), unique number in the business process>*

Each vertex is annotated with a pair indicating the vertex type, name and its corresponding unique number in the business process. As stated earlier, the available types of vertices depend on the adopted business process standard notation. In this thesis, we consider the BPMN notation which distinguishes between activities ('a'), gateways ('g'), and events ('e'). There are also different types of BPMN gateways and events.

Based on Definition 2, Figure. 3.4 shows the process graph of the purchase order process (Figure 1.3). The activity name, the gateway type and the event type represent possible vertex labels (e.g. <('a', 'receive order'),**1**>, <('g','AND-split'),**11**>, <('e','start event'),**0**>).

Conclusion

In this chapter, we presented a novel hybrid approach for business environment-aware management of service-based business processes. Our approach consists in generating automatically a management process to monitor and configure a given managed process. Indeed, automated composition of management operations and business events is one of the most promising challenges in our approach for BEAM of SBPs. The management process generation is performed thanks to a semantic model of SBPs and business environments as well as a dependency analysis of the managed process.

The semantic model involves an upper management ontology, describing relationships between SBPs and business environments. This ontology aligns business processes, services and business environments. In order to evaluate the expressiveness and the semantic richness of our upper management ontology, we compare it to existing business rules languages with respect to a representation theory. Details about this qualitative evaluation are given in Chapter 7.

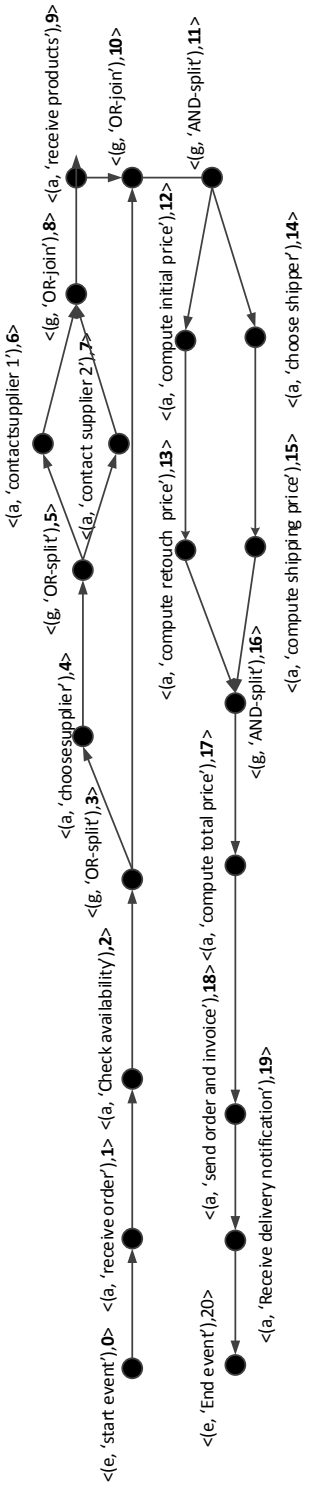


Figure 3.4: The process graph of the purchase order process.

The dependency analysis explicits the structure of the managed process. This latter is based on identifying control and data dependencies to facilitate the organization of the whole management process. In the next chapter, we present the dependency analysis of a given managed process.

Chapter 4

Analysis of the structure of the managed business process

Introduction

Control and data dependencies represent prominent information that supports business process management. However, sequencing constraints described by the control structures befog the true source of dependencies. Therefore, they are often not explicitly presented and are rather implicitly contained in the business process description. A description of a SBP is written in a business process standard language such as Business Process Execution Language (BPEL) [66], XML Process Definition Language (XPDL) [28], etc.

Nowadays, there has been an increasing trend toward the direct execution of business processes modeled in Business Process Modeling Notation (BPMN) [23] on engines such as jBPM [67] and Activiti [6]. Indeed, more than 70 tools support BPMN 2 (see www.bpmn.org). The purpose of BPMN 2 is twofold: (1) facilitating the communication and decision making between domain analysts by enhancing the expressiveness against the business environment (models can be as precise as required by the business context), (2) executing these models directly without mapping problems. The first purpose targets business analysts since real world business processes are able to be modeled directly using directed graphs.

However, there is neither tailored support nor implemented tool for dependency analysis of such processes. In fact, most research works [68, 69, 70, 71, 72, 73, 74] define dependency information for BPEL programs. Indeed, BPEL is a structured, more programming-like language having the same kind of logic and control structures [75]. Nevertheless, business analysts have to deal with the real world, which might be not only unstructured but highly parallel. The fact is that there are parallel unstructured SBPs that cannot be expressed directly into a parallel structured ones [76, 66, 77].

Hence, in general, data and control dependencies are dimensions solely for programming (or programming-like) langages. However, we argue that the unstructured and highly parallel real world processes written in BPMN render them inadequate

which requires some adaptations. Thereby, in this chapter, we adapted and tailored their dependency analysis strategy according to BPMN 2 requirements [78].

In order to explicitly describe the structure of a SBP written in BPMN, we analyze its dependencies (section 4.1) and model them as a dependency graph (section 4.2).

The work of this chapter has been published in [64].

4.1 Dependency analysis

Dependency analysis identifies execution-order constraints between activities and services in a business process program. Broadly speaking, an activity A_2 depends on A_1 if A_1 must be executed before A_2. There are two main classes of dependencies: control dependencies and data dependencies. In order to analyze control and data dependencies in business process programs, we build on the process graph definition (Definition 2) where nodes are labeled distinguishing node types.

In the following subsections, we conduct the example of the purchase order process written in BPMN 2 (Figure 1.3) in order to illustrate the control and data dependencies.

4.1.1 Control dependency analysis

The control dependency is a situation in which a program's instruction executes if the previous instruction evaluates in a way that allows its execution [78]. For example, the activity "choose supplier" is executed if some of the products are not available in stock.

For each program, a control flow graph (CFG) is usually generated in order to facilitate the dependency analysis. A CFG is a graphical representation of all paths that might be traversed through a program during its execution [78]. However, such CFG needs extensions for business process languages. Indeed, in BPMN programs (similarly in BPEL programs as discussed in [68]) there are parallel executions as well as synchronized executions which can obfuscate the true control dependency that we call here "common control dependency". In particular, BPMN (real world) processes present unstructured and highly parallel situations. Therefore, we propose to model business process as process graphs (Definition 2) where nodes are labeled distinguishing between gateway types.

Intuitively, given a PG, a node w is commonly control-dependent on a node u if node u determines whether w is executed. Generally, the common control dependency is defined in terms of post-dominance [78]. In fact, combining control flow and dominance information produces control dependence information.

Definition 3. Post-Dominance [78]. *Given a process graph PG, node B is said to post-dominate node A if every path from A to exit contains B.*

For example, in Figure 3.4 and Figure 1.3, node <('a',"Compute total price"),17> post-dominates <('a',"Check availabilty"),2>. However, <('a',"receive products"),9> doesn't postdominate <('a',"Check availability"),2>.

Definition 4. Common Control dependency [78]. *Let PG be a process graph. Let X and Y be nodes in PG. Y is control dependent on X iff*

- *There exists a directed path P from X to Y with any Z in P post-dominated by Y*
- *X is not post-dominated by Y*

For example, Y =<('a',"receive products"),9> is control dependent on X =<('g',"OR-split"),3> where the condition is the availability or not of the products order. A direcrt path P from X to Y contains: <('a',"choose supplier"),4>, <('g',"OR-split"),5>, <('a',"contact supplier 1"),6>, <('g',"OR-join"),8>. Each Z in P is postdominated by Y. X=<('g',"OR-split"),3> is not postdominated by <('a',"receive products"),9> as illustrated in Definition 4.

Based on Definition 2, we can identify the parallel executions which represent sub-graphs between ('g', "AND-split") and ('g', "AND-join") nodes in BPMN.

Example of Parallel subgraph: {<('a',"compute initial price"),12>, <('a',"compute retouch price"),13>, <('a',"choose shipper"),14>, <('a',"compute shipping price"),15>, <('a',"compute initial price"),12>,
<('a',"compute retouch price"),13>), (<('a',"choose shipper"),14>, <('a',"compute shipping price"),15>}.

The control dependency graph (CDG) represents an explicit description of control dependencies between activities as well as their types. We distinguish three types of control dependencies: "common control-dependency", "parallel dependency" and "synchronized-dependency". As claimed in [68], the "synchronized-dependency" connects the first node of a parallel branch with the first node of the other. Similarly, it connects the last nodes of parallel branches. Thereby, a CDG is defined as follows:

Definition 5. Control dependency graph. *Let Ψ_2 a set of edge types. Let Θ_2 be a set of node labels and Ω_2 a set of edge labels. A CDG is a labeled typed directed graph $CDG = (V_2, E_2, \psi_2, \theta_2, \omega_2)$ with:*

- *V_2 represents the nodes of PG graph*
- *E_2 represent a set of edges*
- *$\psi_2 : E_2 \rightarrow \Psi_2$ is a function that maps edges to types: common-control dependency, parallel dependency, synchronized dependency*
- *$\theta_2 : V_2 \rightarrow \Theta_2$ is a function that maps vertices to labels (numbering function)*

- $\omega_2 : E_2 \rightarrow \Omega_2$ *is a function that maps edges to labels*

Coming back to the purchase order process, its control dependencies are coherently represented in Figure 4.1.

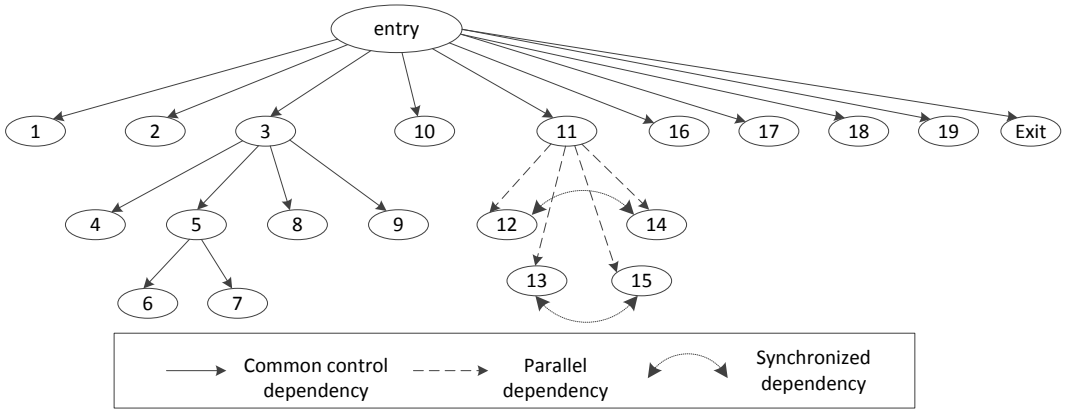


Figure 4.1: Control dependency graph of the purchase order process (1: Receive order, 2: Check availability, 3: OR-split, 4: Choose supplier, 5: OR-split, 6: Contact supplier 1, 7: Contact supplier 2, 8: OR-join, 9: Receive products, 10: OR-join, 11: AND-split, 12: Compute initial price, 13: Compute retouch price, 14: Choose shipper, 15: Compute shipping price, 16: AND-join, 17: Compute total price, 18: Send order and invoice, 19: Receive delivery notification)

4.1.2 Data dependency analysis

Data dependency analysis identifies the potential for a value returned from a service to affect the computation in another. It is used to represent the relevant data flow relationships of a business process program. Indeed, it arises between two activities and/or services such that the first one is a data producer and the second is a data consumer. This type of data dependency is called Definition-Use (Def-Use). It represents the dominant data dependency type in process programming. In this section, we use the Def-Use graph in order to abstract data dependencies for business process standards. Hence, we start by reviewing the concept of Def-Use relation as discussed in [68, 78, 79]. Then, we illustrate data dependencies with respect to the BPMN process illustrated in Figure 1.3.

Definition 6. Variable Definition [68]. *The assignment of a value to a variable x represents its definition: $Def(x)$*

e.g. The node <('a',"compute initial price"),12> outputs the variable *initialprice*: $Def(initialprice)$

Definition 7. Variable Use [68]. *The use of a variable x represents the use of its value: $Use(x)$*

e.g. the node <('a',"compute retouch price"),13> uses the variable *initialprice*: $Use(initialprice)$

Definition 8. Def-Use graph [79, 70]. *The Def-Use graph is a labeled directed graph $DUG = (V, E, \theta)$ such that:*

- *V is the set of process graph vertices (services)*
- *E is the set of edges*
- *$\theta : V \rightarrow (n, l)$ is a function that maps vertices to ordered pair (n,l) where:*
  - *l: a label illustrating the Def and/or Use of variables*
  - *n: represents the corresponding node number in PG as defined in Definition 2*

The Def-Use graph of the purchase order process is depicted in Figure 4.2. (e.g. The node <('a',"compute initial price"),12> in PG has its corresponding number $n = 12$ in DUG. It uses variable $unitPrice$ and defines variable $initialPrice$. It is then labeled with $l = Use(unitPrice), Def(initialPrice)$.

Based on the Def-Use graph, the data dependency of two given nodes is defined as follows:

Definition 9. Data dependency [68]. *Let v_u, $v_d \in DUG$. v_u is data dependent on v_d iff:*

- *there exists a variable x such that v_d contains $Def(x)$ and v_u contains $Use(x)$*
- *there doesn't exist variable redefinition of x within the path from v_d to v_u*

For instance, node 13 is data dependent on 12. Indeed, there exists a variable $initialPrice$ such that node 12 contains $Def(initialPrice)$ and node 13 contains $Use(initialPrice)$ (see Figure 4.2).

The data dependency relationships are recorded in the data dependency graph defined as follows:

Definition 10. Data dependency graph. *Let Ψ_3 a set of edge types. Let Θ_3 be a set of node labels and Ω_3 a set of edge labels. A DDG is a labeled typed directed graph $DDG = (V_3, E_3, \psi_3, \theta_3, \omega_3)$ with:*

- *$V_3 \subset V_1$ represents a subset of PG nodes having data dependencies*
- *E_3 represent a set of edges*

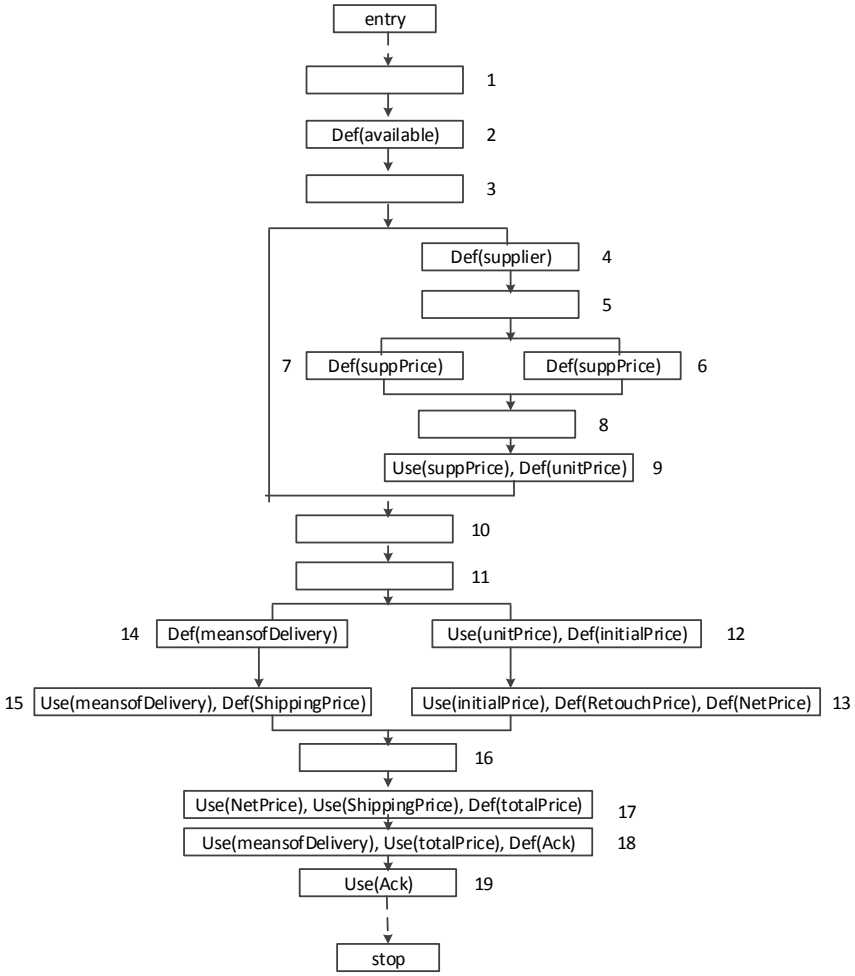


Figure 4.2: Def-Use graph of the purchase order process.

- $\psi_3 : E_3 \rightarrow \Psi_3$ *is a function that maps edges to the type: data dependency*
- $\theta_3 : V_3 \rightarrow \Theta_3$ *is a function that maps vertices to labels (numbering function)*
- $\omega_3 : E_3 \rightarrow \Omega_3$ *is a function that maps edges to labels*

Based on the *DUG* represented in Figure 4.2, the *DDG* of the purchase order process is described in Figure 4.3.

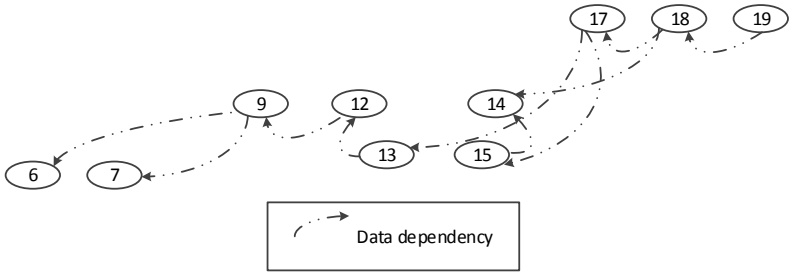


Figure 4.3: Data dependency graph.

4.2 Dependency graph generation

The results of data and control dependency analysis can be recorded in a directed labeled graph that we call Dependency graph. If an edge of control dependencies and/or data dependencies leads from one vertex to another in a dependency graph, then there is a dependency between the activities represented by the vertices. The dependency graph of the purchase order process is shown in Figure 4.4.

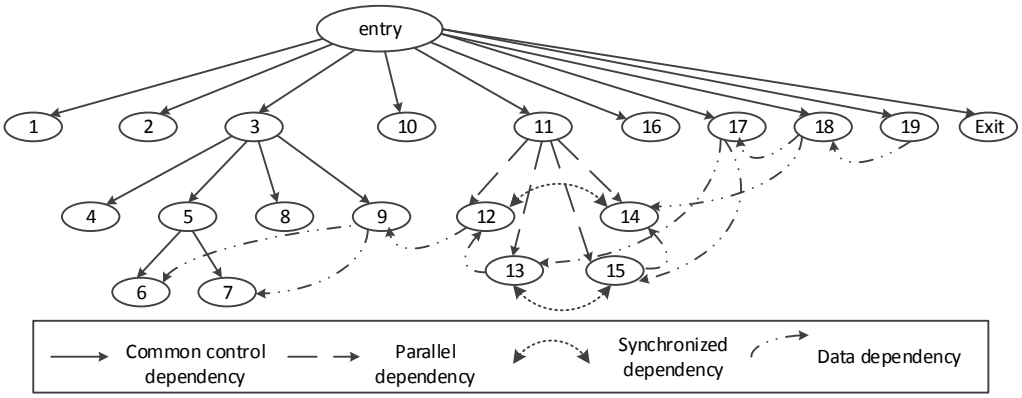


Figure 4.4: Dependency graph of the purchase order process (1: Receive order, 2: Check availability, 3: OR-split, 4: Choose supplier, 5: OR-split, 6: Contact supplier 1, 7: Contact supplier 2, 8: OR-join, 9: Receive products, 10: OR-join, 11: AND-split, 12: Compute initial price, 13: Compute retouch price, 14: Choose shipper, 15: Compute shipping price, 16: AND-join, 17: Compute total price, 18: Send order and invoice, 19: Receive delivery notification)

4.2.1 Control dependency graph generation

CDG is generated according to Algorithm 1 which takes as input the PG of a given business process. Indeed, given the PG of the purchase order process (Figure 3.4), Algorithm 1 outputs the CDG illustrated in Figure 4.1.

This algorithm is inspired from [78, 68]. In the following, we detail its different parts consisting in identifying (1) common control dependency (Line 1-6)(2) parallel dependency (Line 7) and (3) synchronized dependency (Line 26-32).

As discussed in section 5, the first part is based on post-dominance. It comprises in turn three steps which we detail in the following subsections: (a) computing post-dominators (Line 1), (b) generating PDT, (c) computing S, L and Marked (Line 3-6). In order to determine parallel dependencies, we search parallel sub-processes in PG (Line7). Based on parallel sub-graphs, synchronized dependencies are identified as stated in section 5.

The creation of CDG starts by adding the "entry" node (Line 8). Based on the resulting lists (marked nodes and their corresponding control dependent node), nodes that do not depend on any other node is added to CDG and linked with "entry" node (Line 9-12). Similarly, the rest of nodes are created and added in CDG while distinguishing between "parallel-dependency" and "common control dependency" (Line 13-24). Finally, edges labeled "synchronized-dependency" are added between: first nodes and last nodes of parallel branching (Line 26-32).

1. ***Computing post-dominators:*** The first step towards identifying common control dependencies consists in computing post-dominators of each node v in PG. Based on Definition 4, the post-dominators of a node v represent the intersection of all paths containing no cycles from v to exit node.

2. ***Generating post-dominators Tree:*** The second step creates the post-dominators Tree PDT. PDT involves all descendants of a node that are immediately post-dominated by this latter. As a consequence, the tree root is always the node "exit". In order to generate the PDT, we define Algorithm 2 taking as input the list of post-dominators of each node. In each iteration, we remove the last added node from *Postdom* list (Line 6,7). If the post-dominators of the corresponding node are already added to PDT (Line 8), then it will be in turn added to the vertex set of PDT and linked to the last erased node (Line 9-13). The running node will be removed from *nonDesigned* and added to *LastErased*.

3. ***Computing*** S, L, $Marked$ ***and*** CD***:*** After constructing the PDT, we identify the list of nodes $Marked$ that are control dependent on CD. Prior to identify these two sets, we compute S defined as the set of edges (A, B) in PG such that B is not an ancestor of A in PDT. Algorithm 4 details these steps. We start by identifying edges (i, j) in PG such that j is not an ancestor of i in PDT (Line

Algorithm 1 *GeneratingControlDependencyGraph*

Require: Process Graph PG
Ensure: Control Dependence Graph CDG
1: $Postdom \longleftarrow ComputePostdoms(PG)$
2: $PDT \longleftarrow GeneratePDT(Postdom, V_1(PG))$
3: $S \longleftarrow ConstructSLMarkedCD.S$
4: $L \longleftarrow ConstructSLMarkedCD.L$
5: $CD \longleftarrow ConstructSLMarkedCD.CD$
6: $Marked \longleftarrow ConstructSLMarkedCD.Marked$
7: List $SGs \longleftarrow SearchSubgraphs(PG, ('g', "AND - Split"), ('g', "AND - Join"))$
8: $V_2(CDG) \longleftarrow V_2(CDG) \cup \{"entry"\}$
9: $V_2(CDG) \longleftarrow V_1(PG) \setminus \bigcup \{marked(i) \setminus CD(i)\}$
10: **for all** $v \in V_2(CDG)$ **do**
11: $E_2(CDG) \longleftarrow E_2(CDG) \cup \{("entry", v)\}$
12: **end for**
13: **for all** $i \in CD$ **do**
14: **if** $CD(i) \in V_2(CDG)$ **then**
15: **for all** $j \in Marked(i)$ **do**
16: $V_2(CDG) \longleftarrow V_2(CDG) \cup \{j\}$
17: $E_2(CDG) \longleftarrow E_2(CDG) \cup \{(CD(i), j)\}$
18: **if** $(j \in V(SG) \mid SG \subset SGs)$ **then**
19: $CDG.\omega_2((CD(i), j)) = "parallel - dependency"$
20: **else**
21: $CDG.\omega_2((CD(i), j)) = "commoncontrol - dependency"$
22: **end if**
23: **end for**
24: **end if**
25: **end for**
26: **for all** $SG \in SGs$ **do**
27: $\{P_1, P_2\} \longleftarrow Searchpaths(SG, ('g', "AND - Split"), ('g', "AND - Join"))$
28: $E_2(CDG) \longleftarrow (P_1(0), P_2(0))$
29: $CDG.\omega_2((P_1(0), P_2(0))) = "synchronized - dependency"$
30: $E_2(CDG) \longleftarrow (P_1(P_1.length), P_2(P_2.length))$
31: $CDG.\omega_2((P_1(P_1.length - 1), P_2(P_2.length - 1))) = "synchronized - dependency"$
32: **end for**

Algorithm 2 *GeneratingPDTGraph*

Require: List $Postdom$, PG
Ensure: PDT
1: $SetnonDesingned \longleftarrow V(PG) \setminus \{exit\}$
2: $LastErased \longleftarrow \{exit\}$
3: **while** $nonDesingned \neq \varnothing$ **do**
4: **for all** $i \in LastErased$ **do**
5: **for all** $j \in nonDesingned$ **do**
6: **if** $(i \in Postdom(j))$ **then**
7: $Postdom(j) \longleftarrow Postdom(j) \setminus \{i\}$
8: **if** $(sizeof(Postdom(j)) = 0)$ **then**
9: $nonDesingned \longleftarrow nonDesingned \setminus \{j\}$
10: $V(PDT) \longleftarrow V(PDT) \cup \{(i), (j)\}$
11: $E(PDT) \longleftarrow E(PDT) \cup \{(i, j)\}$
12: $NV \longleftarrow NV \cup \{j\}$
13: **end if**
14: **end if**
15: **end for**
16: **end for**
17: $LastErased \longleftarrow NV$
18: **end while**

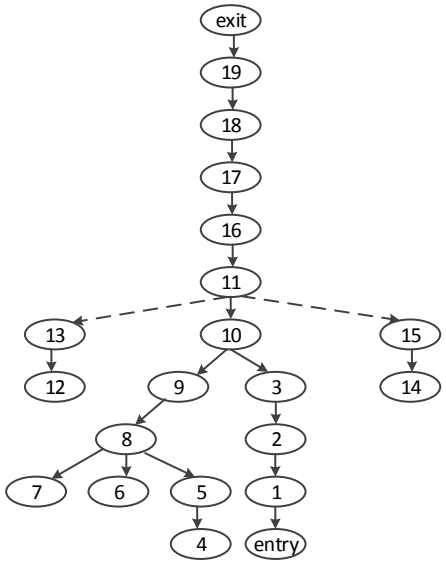


Figure 4.5: Post dominators tree of the purchase order process (1: Receive order, 2: Check availability, 3: OR-split, 4: Choose supplier, 5: OR-split, 6: Contact supplier 1, 7: Contact supplier 2, 8: OR-join, 9: Receive products, 10: OR-join, 11: AND-split, 12: Compute initial price, 13: Compute retouch price, 14: Choose shipper, 15: Compute shipping price, 16: AND-join, 17: Compute total price, 18: Send order and invoice, 19: Receive delivery notification)

4-9). Then, for each couple (A, B) in S (Line 12,13), we determine L the least common ancestor of A and B in PDT (Line 14). By traversing the PDT from B to L, each visited node is marked. L is marked only if $L = A$ (Line 13-17). In iteration i, all marked nodes are saved in $Marked(i)$ (Line 18). Nodes of $Marked(i)$ are control dependent on $CD(i)$ (Line 19).

4.2.2 Data dependency graph generation

The data dependency graph generation is based on both the process graph (Figure 3.4) and the Def-Use graph (Figure 4.2). The DDG is created according to Algorithm 4 based on the data dependency definition (Definition 10) presented in section 4.1.2. For each process variable x (Line 1), we find nodes in DUG such that their labels contains $Def(x)$ (Line 2). By the same way, we fetch nodes in DUG where labels contains $Use(x)$ (Line 3). Nodes labeled respectively $Def(x)$ and $Use(x)$ are added in DDG if the corresponding number of the first is smaller than the corresponding one of the second (Line 5,9).

Algorithm 3 *ConstructSLMarkedCD*

Require: PDT, PG.
Ensure: $S = \bigcup_{i=1}^{m} \{S_i\}, Marked = \bigcup_{i=1}^{m} \{marked_i\}, CD = \bigcup_{i=1}^{m} \{CDon_i\}, L = \bigcup_{i=1}^{m} \{L_i\}$

for all $i \in NodeSet$ **do**
  for all $j \in PG.adjacentTo(i)$ **do**
    if $(\not Ancestor(pdt, j, i))$ **then**
      $S \longleftarrow (i, j)$
    end if
  end for
end for
for $(i = 0 \rightarrow sizeof(S))$ **do**
  $A \longleftarrow S(i)(0)$
  $B \longleftarrow S(i)(1)$
  $L(i) \longleftarrow Least(A, B)$
  $Bool\ inclus$
  if $(L(i) = A)$ **then**
    $inclus \longleftarrow true$
  else
    $inclus \longleftarrow false$
  end if
  $Marked(i) \longleftarrow mark(inclus, L(i), B)$
  $CD(i) \longleftarrow A$
end for

Algorithm 4 *GeneratingDataDependencyGraph*

Require: List $ProcessVariables$, DUG
Ensure: DDG

for all $x \in ProcessVariables$ **do**
  Find $\{v_1\} \in V(DUG)$ such that $v_1.\theta.l$ contains $Def(x)$
  Find $\{v_2\} \in V(DUG)$ such that $v_2.\theta.l$ contains $Use(x)$
end for
for all $i \in \{v_2\}$ **do**
  Find $j \in \{v_1\}$ such that $v_1.\theta.n < v_2.\theta.n$ and $v_1.\theta.n = Max(v.\theta.n)|v \in \{v_1\}$
  $V_2(DDG) \longleftarrow V_2(DDG) \cup \{v_1, v_2\}$
  $E_2(DDG) \longleftarrow E_2(DDG) \cup \{(v_1, v_2)\}$
end for

Conclusion

Dependency analysis of SBPs provides relevant information for managing SBPs. In this chapter, we presented control and data dependencies for BPMN processes which describe explicitly their structure. BPMN 2.0, as a twofold-purpose language, can be used to both model and execute these SBPs. However, there is no tailored support for dependency analysis of such processes. Hence, we adopted and adapted definitions and algorithms of dependency analysis of programming languages in order to generate the dependency graph of a given SBP.

In order to validate these algorithms, we designed and implemented DAT, a dependency analysis tool for SBPs written in BPMN. Details of this framework are given in Chapter 6,

Given a managed process, DAT generates its corresponding dependency graph which explicits its structure. This dependecny graph together with the semantic model facilitates the management process generation. In the next chapter, we present how to generate this management process.

Chapter 5

Management process generation

Introduction

In this chapter, we show how to generate a management process to handle a SBP during its execution. We recall that properties of services that composed the management process frequently change due to business environment events. When a new event occurs, the adequate properties should be updated. Therefore, the management process consists in a composition of management operations that read and/or alter services' properties. To do this, we define a getter and a setter for each property (see Section 3.2.1.3).

The construction of the management process (composition of management operations) is performed using semantic descriptions over domain ontology (see Section 3.2) as well as the structure of the managed business process (see section 4.1). Thereby, the construction of the composition comprises three main phases: (1) constructing sub-processes based on the Environment-Service relationship, (2) constructing sub-processes based on the Service-Service relationship and (3) connecting generated sub-processes.

Algorithm 5 describes the statements executed for handling these phases which we detail in the following illustrated by the running example.

Properties externalize service behaviors. Thus, the first step towards the automation of the management operations composition is to capture the semantic concepts of services properties from the managed business process (Line 1). Each service property can have possible events that trigger the update of its value (Line 3). Thus, *ConstructESR* method (Line4) is called with p as parameter for building sub-processes relating configuration operations with events (Algorithm 6 in Section 5.1). Configuring a service property may engender the update of other properties related to it. Besides, the service property can have other relationships with other service properties. *ConstructSSR* (Line 5) is called in turn to build a sub-process connecting management operations with each other (Algorithm 7 in Section 5.2). Finally, *ConnectSP* (Line 7) is performed to connect resulting subprocesses based on both the structure of the managed SBP and semantic relationships between properties (Algorithm 10 in Section 5.3).

Doing so, we are based on the following BPMN patterns [80]:

- Basic control-flow patterns
  - **Sequence:** the ability to depict a sequence of activities
  - **Parallel split:** the divergence of a branch into two or more parallel branches executed concurrently.
  - **Synchronization:** the convergence of multiple parallel branches into a single thread of control thus synchronising multiple threads.
  - **Exclusive choice:** a decision point in a process process where one of several branches is chosen.
  - **Simple merge:** a point in a process where two or more alternative branches come together without synchronisation.
- State-based pattern
  - **Deffered choice:** a divergence point in a process where one of several possible branches should be activated by the environment. In BPMN, this pattern is supported via an event-based exclusive gateway followed by either intermediate events using message-based triggers or receive tasks.

Algorithm 5 *GeneratingManagementProcess*

Require: Process Graph PG
Ensure: Managing Graph MG
1: List $P \longleftarrow \{S.p.concept, S \in V_1(PG)\}$
2: **for all** Properties $p \in P$ **do**
3: List $L_1 \longleftarrow$ *FindEvents(p)*
4: $MG \longleftarrow$ ConstructESR(p, MG, L_1)
5: $MG \longleftarrow$ ConstructSSR(p, MG)
6: **end for**
7: $MG \longleftarrow$ ConnectSP(PG, MG)
8: **return** MG

5.1 Constructing sub-processes based on Environment-Service relationship

In this first phase, the issue is to alter a service property based on the Environment-Service relationship introduced in Section 3.2.1. Indeed, when an event occurs the corresponding service property is updated according to dependencies between business environment events and services.

In accordance with the running example, when a "FestiveSeason" happens, the criteria for choosing suppliers change. Subsequently, the property named "CriteriaList" is altered. As stated in Section 3.2.1, the event "FestiveSeason" is composed of atomic events having event definitions: CriteriaListMessage and FestiveTimeDate.

In order to create sub-processes aiming at modifying a service property, Algorithm 6 is performed. These subprocesses relate a service management operation with possible events that can trigger it.

Figure 5.1(a) is the resulting sub-process for p="CriteriaList". Similarly, with p="DiscountRate" the sub-process described in Figure 5.1(b) is generated. Indeed, when a "Sales promotion" happens, there is a decrease in clothes prices (FirstDiscount). Subsequently, the property named "DiscountRate" is altered. As stated in section 3.2.1, the event "FirstSalesPromotion" is composed of atomic events having event definitions: DiscountRateMessage and PromotionTimeDate.

The list of possible events as well as their definitions (input of Algorithm 6) result from calling the procedure *FindEvents(p)* that executes the following SPARQL query: **SPARQL Query 1:** "SELECT ?atomicEvent ?eventdefinition WHERE { ?event :trigger ?action. ?action :act-on ?property. ?property rdf:type :*p*. ?event :hasEventstructure ?events. ?events :composedOf ?atomicEvent. ?atomicEvent :hasEventDefinition ?definition. ?definition rdf:type ?eventdefinition.}".

When an event occurs, the service property p will be altered automatically using a set operation. A vertex ("*a*","*set*(*p*)") is added to the vertex-set of the managing graph *MG* (Line 2). If the list of possible events that can modify the property comprises only one event, we add this event to the set of vertices of *MG* graph (Line 4). A single edge between the event and the "set" operation is also added (Line 5). Otherwise, a node of gateway type labeled "*Event-based XOR*" is added (Line 7). Then, a node for each event and edges relating it to the gateway as well as the set operation are identified (Line 9, 10, 11).

In case there are no event related to p (Line 14), the events related to its *superServiceProperty* are identified (Line 15,16). Algorithm 6 is then recalled taking as input the list of these events (Line 18).

5.2 Constructing sub-processes based on Service-Service relationship

A service property may depend on others. Hence, updating a service property may engender the modification of others depending on it. Therefore, in this second phase, the concern is to properly identify the semantic relationship holding between service properties.

For example, the service properties named "ShippingPrice" and "RetouchPrice" depend on "DiscountRate" property (Figure 3.3). Thus, if "DiscountRate" is up-

Algorithm 6 *ConstructESR(ServiceProperty p, Managing Graph MG, List L_1)*

Require: Managing Graph MG
Ensure: Managing Graph MG
1: **if** $L_1 \neq \emptyset$ **then**
2: $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}a\text{"}, \text{"}set(p)\text{"})\}$
3: **if** $L_1 = \{l_1\}$ **then**
4: $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}e\text{"}, \text{"}l_1\text{"})\}$
5: $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}e\text{"}, \text{"}l_1\text{"}), (\text{"}a\text{"}, \text{"}set(p)\text{"}))\}$
6: **else**
7: $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}g\text{"}, \text{"}Event - basedXOR\text{"})\}$
8: **for all** $l_1 \in L_1$ **do**
9: $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}e\text{"}, \text{"}l_1\text{"})\}$
10: $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}g\text{"}, \text{"}Event - basedXOR\text{"}), (\text{"}e\text{"}, \text{"}l_1\text{"}))\}$
11: $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}e\text{"}, \text{"}l_1\text{"}), (\text{"}a\text{"}, \text{"}set(p)\text{"}))\}$
12: **end for**
13: **end if**
14: **else**
15: String $SuperServiceProperty \longleftarrow FindSuperClassServiceProperty(p)$
16: $SuperList \longleftarrow FindEvents(SuperServiceProperty)$
17: **if** $SuperList \neq \emptyset$ **then**
18: $ConstructESR(p, MG, SuperList)$;
19: **end if**
20: **end if**
21: **return** MG

dated, both "ShippingPrice" and "Retouch price" should be updated. The corresponding resulting subprocess is depicted in Figure 5.1(c).

In order to generate this subprocess, Algorithm 7 explores the different dependency relationships between concepts of services' properties from the domain ontology. Two services properties have a relationship if they are related with "depends-on" relationship in the domain ontology. A SPARQL query is then sent to the domain ontology to enquire for the sources of the property p :

SPARQL Query 2: "SELECT ?sourceType WHERE ?source :depends-on ?a. ?a rdf:type :*p*. ?source rdf:type ?sourceType. ?sourceType rdfs:subClassOf :ServiceProperty.".

The result of this query is performed by calling the procedure *ServiceSourceOfDependsOn(p)* (Algorithm 3, Line 1). If p has properties that depend on it (Line 2), then the $get(p)$ operation is automatically invoked (Line 3). As a result, a setter for each property depending on p is defined (Line 4-6). If there is only one property, then a simple edge links its setter with $get(p)$ (Line 7-8). Otherwise, the adequate gateway relating properties setters with $get(p)$ is identified using Algorithm 8. In this latter algorithm, we rely on dependencies between services in the managed BP represented in the control dependency graph (see section 3.2.2.2). For example, the services "Compute retouch price" and "Compute shipping price" are synchronized according to the dependency graph of the purchase order process (Figure 4.1). Therefore, a gateway

labeled ('g','AND-Split) is added. As for a well structured BP, when starting with a gateway type, we finish by the same one (Line 14-17).

The relationship between services properties could be also clearly identified through the ontology via RDF containers (see section 3.2.1.2). Indeed, sequence, exclusivity and mutuality are described respectively via RDF:Seq, RDF:Alt, RDF:Bag. For example, the property "SecondDiscount" is composed of a sequence of "DiscountRate" and "SupplementaryDiscountRate". By calling $ContainersRelationships(p)$, we execute SPARQL Query 3 in order to identify all containers in which p represents a member (Line 20). We identify the container type and the mother service property "MotherConcept" which includes container members among them p (Line 23, 24). The containers members other than the current property p are identified by executing SPARQL Query 4 (Line 25). For each member concept, a new vertex representing an activity calling the management operation set(l) (l=member concept) is added (Line 26-28). According to the container type, we identify the connector type by executing Algorithm 9. If the container type is RDF:Seq, the connector type will be a sequence flow. The BPMN sequence pattern is then used in order to link the management operations (Line 31-33). Otherwise, the corresponding gateway is added and the links to the management operations are established (Line 37-39).

SPARQL Query 3: "SELECT ?containerType ?MotherIndividual ?MotherConcept WHERE {?container rdfs:member ?member. ?member rdf:type :*p*. ?container rdf:type ?containerType. ?MotherIndividual rdf:object ?container. ?MotherIndividual rdf:predicate ?MotherConcept.}".

SPARQL Query 4: "SELECT ?memberType WHERE {?container rdfs:member ?member. ?member rdf:type ?memberType. ?container rdf:type ?containerType. ?MotherIndividual rdf:object ?container. ?MotherIndividual rdf:predicate :*p*.}".

5.3 Connecting sub-processes

So far, a set of sub-processes are created. Indeed, for each property sub-processes based on the Environment-Service and/or Service-Service relationship are built. How to connect them? How to determine their order?

Resuming with the running example, till now, four sub-processes are built (see Figure 5.1). In order to connect them aiming to generate the whole management process (Figure 3.1), in this phase, we add necessary links and gateways based on both the upper management ontology and the explicit dependency description of the managed BP.

As a first step, we start by linking Event-based sub-processes according to the relationships between services properties. Hence, SPARQL Query 3 and 4 are respectively executed in Algorithm 10 to identify (1) the RDF container which determine relationships between these service properties (Line 4) and (2) container members (Line 5).

Algorithm 7 *ConstructSSR(ServiceProperty p, Managing Graph MG)*

Require: Managing Graph MG
Ensure: Managing Graph MG

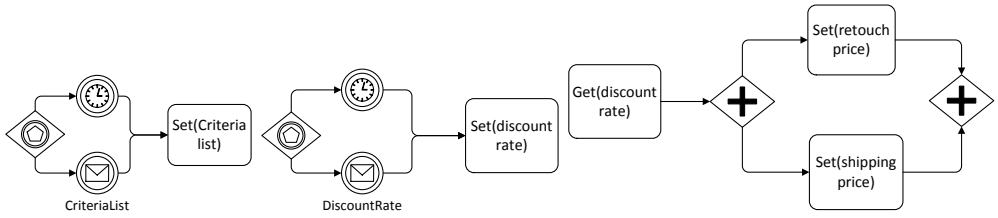
List $L_2 \longleftarrow$ *ServiceSourceOfDepends-On(p)*
if $L_2 \neq \emptyset$ **then**
  $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}a\text{"}, \text{"}get(p)\text{"})\}$
  for all $l \in L_2$ **do**
    $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}a\text{"}, \text{"}set(l)\text{"})\}$
  end for
  if $L_2 = \{l_2\}$ **then**
    $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}a\text{"}, \text{"}get(p)\text{"}), (\text{"}a\text{"}, \text{"}set(l_2)\text{"}))\}$
  else
    String GatewayType=*ChooseGateway(L_2, p)*
    $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}g\text{"}, GatewayType)\}$
    $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}a\text{"}, \text{"}get(p)\text{"}), (\text{"}g\text{"}, GatewayType))\}$
    $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}g\text{"}, GatewayType)\}$
    for all $l_2 \in L_2$ **do**
      $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}g\text{"}, GatewayType), (\text{"}a\text{"}, \text{"}set(l_2)\text{"}))\}$
      $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}a\text{"}, \text{"}set(l_2)\text{"}), (\text{"}g\text{"}, GatewayType))\}$
    end for
  end if
end if
$CR \longleftarrow$ *ContainersRelationships(p)*
if $CR \neq \emptyset$ **then**
  for all $cr \in CR$ **do**
    $Stringcontainer \longleftarrow cr.getElement().getKey()$
    $StringcontainerServiceProperty \longleftarrow cr.getElement().getValue()$;
    List $L = MemebersOfContainerServiceProperty(container)$ {Execute Query 4}
    for all $l \in L$ **do**
      $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}a\text{"}, \text{"}set(l)\text{"})\}$
    end for
    String ConnectorType=*ChooseConnector(container)*
    if $ConnectorType = \text{"}Sequenceflow\text{"}$ **then**
      for all $l \in L$ **do**
        for all $l_1 \in L$ **do**
          $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}a\text{"}, \text{"}set(l)\text{"}), (\text{"}a\text{"}, \text{"}set(l_1)\text{"}))\}$
        end for
      end for
    else
      $V_3(MG) \longleftarrow V_3(MG) \cup \{(\text{"}g\text{"}, ConnectorType)\}$
      for all $l \in L$ **do**
        $E_3(MG) \longleftarrow E_3(MG) \cup \{((\text{"}g\text{"}, GatewayType), (\text{"}a\text{"}, \text{"}set(l)\text{"}))\}$
      end for
    end if
  end for
end if
return MG

Algorithm 8 ***ChooseGateway(List*** L_2***, ServiceProperty*** p***)***

Require: Process graph PG
Ensure: String GatewayType
1: **for all** $l \in L_2$ **do**
2: **for all** $h \in L_2$ **do**
3: Let $S_1 \in V_1(PG)$ such that $S_1.p.concept = l$
4: Let $S_2 \in V_1(PG)$ such that $S_2.p.concept = h$
5: Let $e \in V_1(PG)$ such that $e = (S_1, S_2)$
6: **if** $(\omega_1(e) = "synchronized - dependency")$ **then**
7: GatewayType="AND-Split"
8: **end if**
9: **if** $(\omega_1(e) = "commoncontrol - dependency")$ **then**
10: GatewayType="OR-Split"
11: **end if**
12: **end for**
13: **end for**

Algorithm 9 ***ChooseConnector(String container)***

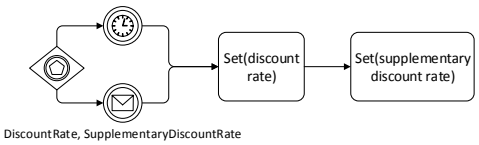
Require: String container
Ensure: String ConnectorType
1: **if** $(container = RDF : Alt)$ **then**
2: ConnectorType="OR-Split"
3: **end if**
4: **if** $(container = RDF : Bag)$ **then**
5: ConnectorType="AND-Split"
6: **end if**
7: **if** $(container = RDF : Seq)$ **then**
8: ConnectorType="Sequence flow"
9: **end if**



(a) Result of phase 1 for p="CriteriaList"

(b) Result of phase 1 for p="DiscountRate"

(c) Result of phase 2 for p="DiscountRate"



(d) Result of phase 2 for p="SupplementaryDiscountRate"

Figure 5.1: Result of phase 1 and phase 2

Finally, the corresponding sub-processes are linked by choosing the adequate gateway (Line 7-9). For example, both sub-processes in Figure 5.1(b) and Figure 5.1(d) represent two discount strategies depicted in the domain ontology. These strategies, represented by the service properties "FisrtDiscount" and "SecondDiscount", are applied exclusively.

Figure 5.2 shows the resulting sub-process.

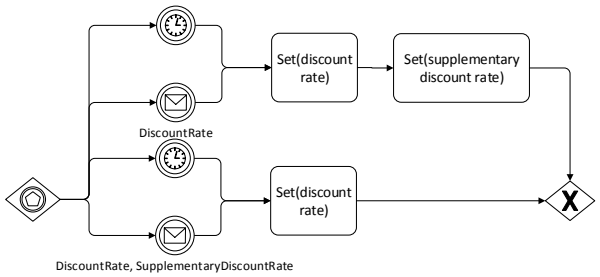


Figure 5.2: Result of connecting subprocesses based on the upper management ontology.

Afterwards, we link the new list of sub-processes based on the control dependency analysis of the managed BP. To do so, we adopted the following phases: (1) identifying management process ends, (2) capturing their correspondings in the managed BP, (3) determining control dependencies for each activity in order to add corresponding gateways and (4) organizing results based on the control flow of the managed BP.

Formalisation of these phases is given in Algorithm 10. The first phase consists in finding nodes having no targets (set operations) and nodes having no sources (get operations) (Line 13). The second phase identifies nodes corresponding to these activities having p as property in the process graph (Line 14). Afterwards, the control dependency of each node is determined (Line 15). Control dependencies for each node are then compared in order to identify the existence or not of control dependency between sub-processes (Line 16-26). Then, according to control-flow relations between activities in the process graph, the sub-processes are organized and control flow edges are added to the managing graph (Line 28). Finally, nodes having no sources are linked to the start event, and those having no targets are connected to the end event (Line 31-35).

Resuming with the running example, in the first phase, the corresponding activities "Choose supplier" and "Compute initial price" of the management operations (Set(CriteriaList), Set(DiscountRate), Get(DiscountRate)) are identified. Coming back to the control dependency graph (Figure 4.1), "choose supplier" is control dependent on node 3. Thus, an "OR-split" gateway is added to the management process. By checking the non dependencies between both activities "Choose supplier"

and "Compute initial price", an "Or-join" gateway is added to the management process. Then, based on the corresponding unique numbers, we determine the order of the management operations and we add links. Figure 3.1 depicts the resulting management process.

Conclusion

In this chapter, we answer the following reseach question raised in Section 3.1: How to modify service properties and which order to follow to compose management operations? To do so, we defined three main phases for automatically compose management operations based on the semantic model and the dependency analysis of the managed process. The first phase consists in building sub-rocesses based on Environment-Service relationships. The second phase builds subprocesses based on the Service-Service relationships. Finally, the third phase connects the resulting sub-processes based on both the structure of the managed process and semantic description of service properties and events.

To validate our work, we implemented a framework that monitors and configures a given managed process. We also performed experiments using not only the running example but also BPMN control flow patterns. Details on implementation and experiments are presented in Chapter 6 and Chapter 8, respectively.

Algorithm 10 *ConnectSP(Process Graph PG, Managing Graph MG)*

Require: Managing Graph MG, Process Graph PG
Ensure: Managing Graph MG

HashMap $Subprocesses$ <serviceproperty, subprocess>
for all s in $Subprocess$ **do**
  String $serviceproperty \longleftarrow s.getKey()$
  String $containerType = FindContainerRelationships(serviceproperty)$
  List $L = MemebersOfContainerServiceProperty(container)$
  for all serviceprop in L **do**
    $Subprocesssubprocess \longleftarrow s.get(serviceprop)$
    String connector=chooseConnector(containerType)
    $V_3(MG) \longleftarrow V_3(MG) \cup \{('g', connector)\}$
  end for
end for
for all $v \in V_3(MG)$ **do**
  if $(S(v) = \emptyset \wedge MG.\tau_3(v) =' a') \vee (T(v) = \emptyset \wedge MG.\tau_3(v) =' a')$ **then**
    Find v_1 in $V_1(PG)$ such that $v_1.p.concept = MG.\theta_3(v)$
    $v_2 \longleftarrow searchControldependencies(CDG, v_1)$
    if $PG.\omega_1((v_1, v_2)) =$ "$commoncontrol - dependency$" **then**
      $V_3(MG) \longleftarrow V_3(MG) \cup \{('g', "OR - Split")\}$
      $E_3(MG) \longleftarrow E_3(MG) \cup \{(('g', "OR - Split"), PG.\theta_1(v_1)\}$
    end if
    $Map \longleftarrow Map \cup (v_1, v_2)$
  end if
end for
if $\bigcap\{Map(i)\} = \varnothing$ **then**
  $V_3(MG) \longleftarrow V_3(MG) \cup \{('g', "OR - Join")\}$
  $E_3(MG) \longleftarrow E_3(MG) \cup \{PG.\theta_1(v_1), ('g', "OR - Join")\}$
  $E_3(MG) \longleftarrow E_3(MG) \cup \{('g', "OR - Split"), ('g', "OR - Join")\}$
end if
$precedence(PG, n1, n2)$
$V_3(MG) \longleftarrow V_3(MG) \cup \{(("e", "Startevent"), ("e", "Endevent"))\}$
for all $v \in V_3(MG)$ **do**
  if $S(v) = \emptyset$ **then**
    $E_3(MG) \longleftarrow E_3(MG) \cup \{(("e", "Startevent"), MG.\theta_3(v))\}$
  end if
  if $T(v) = \emptyset$ **then**
    $E_3(MG) \longleftarrow E_3(MG) \cup \{(MG.\theta_3(v), ("e", "Endevent"))\}$
  end if
end for
return MG

Part III

Implementation and evaluation

Chapter 6

Implementation

Introduction

To test the feasibility of our approach, we implemented a Business Environment-Aware Management Framework BEAM4SBP [1]. In the following subsections, we start by presenting an overview of the BEAM4SBP architecture. Then, we present the Dependency Analysis Tool (DAT), plug-in of BEAM4SBP framework. Afterwards, we present the integration of BEAM4SBP into the Activiti process engine.

6.1 The BEAM4SBP framework

BEAM4SBP is a java library that enables to generate a management process connected to the managed business process allowing for its monitoring and configuration. Given the purchase order process (Figure. 1.3) and the purchase order ontology (Figure. 3.3), BEAM4SBP outputs the expected management process to connect to the purchase order process (Figure. 3.1). In the following, we present the framework's architecture.

As shown in Figure. 6.1, the architecture of BEAM4SBP is composed of four main components: (1) *Process Graph Generator* (2) *Dependency Graph Generator* (3) *Jena module* (4) *Managing Graph Generator*.

The first component involves a *BPMN Parser* developed to enable the process graph creation and takes as input a valid BPMN process (Section 3.2). It is developed using EMF [81] to generate a Java Model for the BPMN 2.0 specification described in XSD files. Indeed, EMF can easily generate a parser for any language specification given its meta-model. Based on the BPMN java model, we extract events and tasks. The extracted information are used to create the process graph (Definition. 2, Chapter 3) based on the Jgrapht java library [82].

This latter represents the input of the *Dependency Graph Generator* component which outputs its corresponding dependency graph (Figure. 4.4) allowing to get the

[1] BEAM4SBP can be found at `http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Framework`.

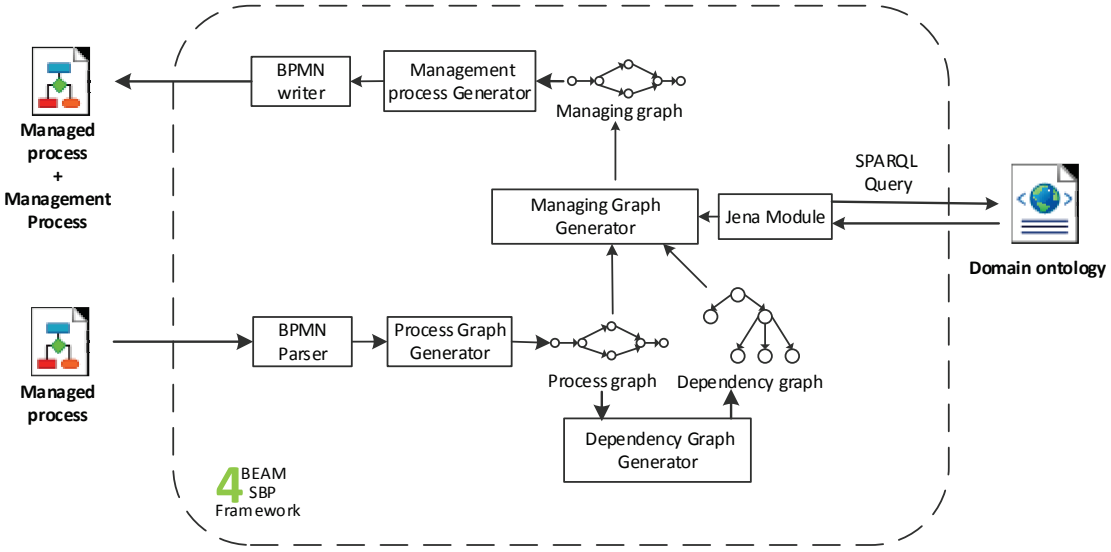


Figure 6.1: Business environment-aware management framework for SBPs.

explicit dependencies between activities. This component is also based on the Jgrapht library. It is implemented as a plug-in of BEAM4SBP (see Section 6.3).

The resulting dependency graph and the *Jena module* represent the inputs of the Algorithm 5 implemented by the *Managing Graph Generator*. This latter module outputs a managing graph representing the management process model (Section 3.2). The *Jena Module* implements the procedures *FindEvents* and *FindSourceofDependsOn* called, respectively, in Algorithm 6 and Algorithm 7.

The resulting managing graph is translated into a process description with flow nodes by the *Management Process Generator* module. Finally, the *BPMN Writer* outputs a BPMN file connecting the management process to the managed business process (Figure 6.2).

6.2 The Dependency Analysis Tool (DAT)

In this section, we present the Dependency Analysis Tool (DAT) [2] which implements the Dependency graph Generator component depicted in Figure 6.1. We start by giving an overview of the DAT's architecture (Section 6.2.1). Then, we present its main functionalities (Section 6.2.2).

[2]DAT can be found at `http://www-inf.int-evry.fr/SIMBAD/tools/DAT`.

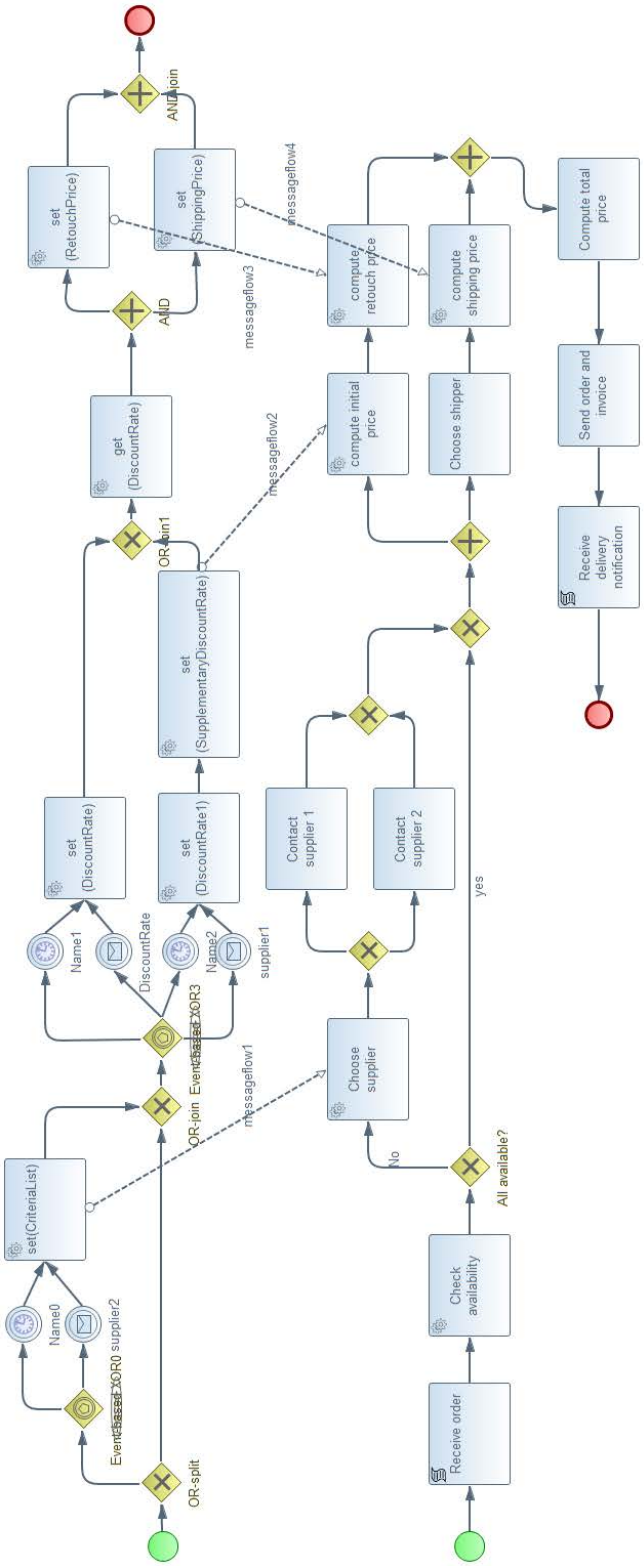


Figure 6.2: The generated BPMN file.

6.2.1 Architecture overview

DAT is a java library aiming at generating a dependency graph for a given business process. Given the BPMN business process, the DAT outputs the dependency graph shown in Figure. 4.4. In the following, we present the framework's architecture. The architecture of DAT comprises six main modules: (1) Parser, (2) Process Graph Generator, (3) Def-use Graph Generator, (4) Data Dependency Graph Generator, (5) Control Dependency Graph Generator, (6) Dependency Graph Generator.

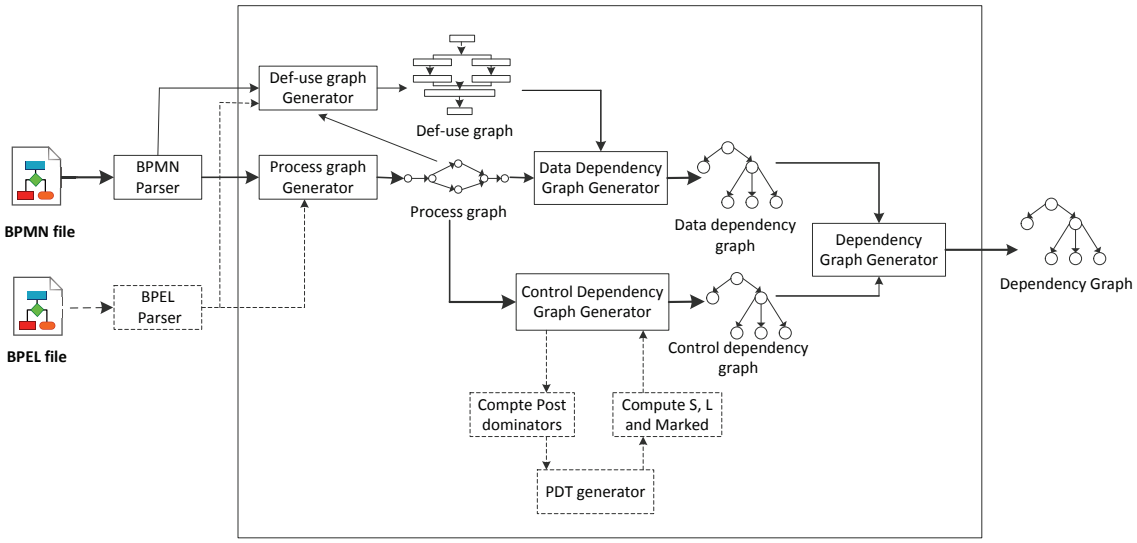


Figure 6.3: Dependency Analysis Tool (DAT) Architecture.

The first module implements a Parser taking as input a well formed executable business process. The extracted information are used to create the process graph as well as the Def-use graph based on the Jgrapht java library [82]. In this work, we developed a BPMN parser. BPMN processes are syntactically verified under Activiti 5.12 process engine [6]. The BPMN file purchaseorderprocess.bpmn20.xml as well as a tutorial for creating executable BPMN processes are provided in the tab DOWNLOADS of the DAT's web site.

The *Process Graph Generation* module generates the process graph based on the parsing results. In case of BPMN processes, the extracted information are events, tasks, gateways, etc. The process graph involves event, gateway and task nodes (e.g. ('e', "startevent"), ('e', "AND-split")) (see Figure. 3.4).

The process graph represents the input of the *Control dependency graph Generator* module and outputs a control dependency graph allowing to get the explicit dependencies between activities. This module implements Algorithm 1 and is also

based on the Jgrapht library. The resulting dependency graph involves labelled edges representing the edge type: "Common control dependency", "Parallel dependency", "Synchronized dependency". We extended the Jgrapht library by defining labelled edges inspiring from weighted edges. The identification of common control dependencies is done with three modules *Compte Post dominators*, *PDT generator* and *Compute S, L and Marked* that implements Algorithm 2 and Algorithm 5.

The third module outputs the Def-use graph which is created based on the extracted input/output of activities as well as the process graph. The created Def-use Graph correlated with the process graph represents the input of the *Data Dependency Graph Generator* module. This latter implements Algorithm 4.

6.2.2 DAT's functionalities

The framework provides two main functionalies aiming at identifying (1) data and control dependencies of one activity against all existing ones and particularly (2) dependencies existing between two given activities.

Dependencies of an activity: Given the dependency graph, control and data dependencies relationships of one activity can be determined. In fact the framework takes as input an activity and outputs its different data and control dependencies. This functionality can be used in slicing programs [68] as well as managing SBPs [83].

Dependencies between two given activities: Introducing two activities, DAT provides control and data dependencies between them. This functionality can be used in modeling and managing business processes [83].

6.3 Integrating BEAM4SBP into Activiti

BEAM4SBP is generic and may be integrated with any process engine developed in Java (e.g. Activiti, jBPM, etc.). In order to test the resulting management process and the configuration at run-time of the managed process, we integrated the BEAM4SBP framework as a plug-in in the *Activiti* business process engine [6].

In the subsequent subsections, we present the *Activiti* engine, the integration as well as a scenario of the process configuration.

6.3.1 Activiti Engine

Activiti [6] is a framework that provides an environment for designing, implementing, deploying and running processes described in BPMN 2.0. It is an open source project distributed under the Apache license. *Activiti* is written in java; hence its core is a super-fast and rock-solid BPMN 2.0 process engine. The project of *Activiti* is funded by Alfresco and established by jBPM founder Tom Baeyens [84].

Activiti [37] is a framework that provides an environment for designing, implementing and running processes described in BPMN 2.0. It is an open source project distributed under the Apache license. One of its strengths is that it is written in java; hence its core is a super-fast and rock-solid BPMN 2.0 process engine. The project of Activiti is funded by Alfresco and established by jBPM founder Tom Baeyens [36].

Figure 2 shows a general overview of the Activiti components which make the whole framework. (The figure is adopted from [41])

Figure 6.4 shows a general overview of the *Activiti* components which make the whole framework.

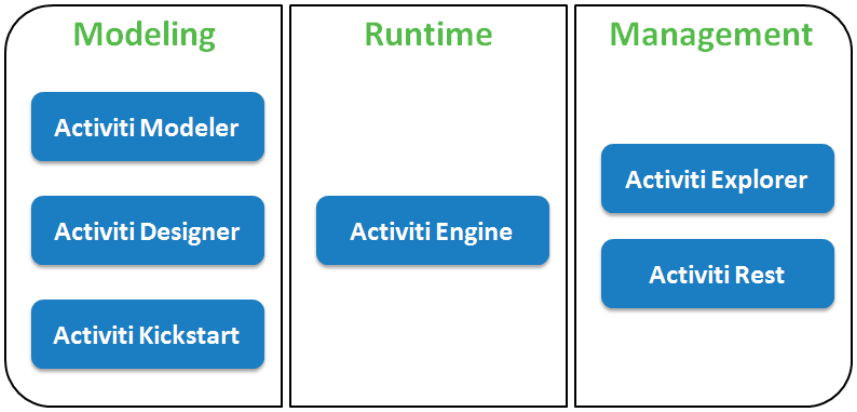


Figure 6.4: *Activiti* components [6].

Figure 2: Activiti components

As shown in Figure 6.4, *Activiti* is divided into three different parts. Activiti Modeler, Activiti Designer and Activiti Kickstart compose the modeling part applied to design BPMN 2.0 processes. The management part consists of Activiti Explorer and Activiti Rest. While, the runtime part comprises the Activiti Engine which is the core component of the *Activiti* project. It performs the process engine functions [84] such as deploying the business processes, starting instances, creating and executing workflow tasks.

As shown in Figure 2, Activiti is divided into different components. Activiti Modeler, Activiti Designer and Activiti Kickstart are applied to model BPMN 2.0 processes. For the management part, it is represented by Activiti Explorer and Activiti Rest. In the middle, in the runtime part, there is Activiti Engine which is the core component of the Activiti project. It performs the process engine functions [36] such as deploying the business processes, starting instances, creating and executing workflow tasks.

Our BEAM4SBP framework will be accessible via Activiti Explorer as it provides an easy-to-use web interface. Activiti Explorer is a web-based application that can be simply used for various tasks in conjunction with Activiti Engine. It is developed to both technical and non-technical users. On one hand, they can model BPMN 2.0 processes through Activiti Modeler. On the other hand, they can deploy and run business processes thanks to the core component Activiti Engine.

In our work, the BEAM4SBP is accessible via Activiti Explorer as it provides an easy-to-use web interface. By definition, Activiti Explorer is a web-based application that can be simply used for various tasks in conjunction with Activiti Engine [36]. It is destined to both technical and non-technical persons. From its interface, we can model a BPMN 2.0 process through the Activiti Modeler. We can also, deploy and run business processes thanks to the Activiti Engine.

Technically, the Activiti Explorer is a war file that can be deployed in a web server such as Apache Tomcat [85]. Then, it is accessible through the following address:

```
http://localhost:8080/activiti-explorer
```

Technically, the Activiti Explorer is a war file that should be deployed in a web server. So, we select the Apache Tomcat [48] and it becomes accessible by using a web browser through: http://localhost:8080/activiti-explorer.

6.3.2 The integration

First and foremost, we conducted a deep study of the source code of Activiti Explorer. Then, we started the files to be modified. The major challenge we encountered was keeping the proper functioning of Activiti Explorer since some files have been either changed or added.

In order to integrate BEAM4SBP into Activiti, we need the following software to be installed:

- Eclipse [86]: is an integrated development environment (IDE). It is an extensible plugin system for adapting the applications.

- The plugin m2eclipse [87]: manages Maven projects, executes Maven builds via the Eclipse interface, and interacts with Maven repositories. It makes the development easier with Eclipse IDE.

- Source code of Activiti: is downloadable from the following address: `https://github.com/Activiti/Activiti/releases`

- Apache Maven [88]: is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

- Apache Ant [89]: is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications.

- Apache Tomcat [85]: is an open source web server and servlet container developed by the Apache Software Foundation (ASF). It implements the Java Servlet and JavaServer Pages technologies.

Details of the integration are described in a technical report for both developers and users [90].

After having accomplished the integration, we generate a new war file for Activiti Explorer called "activiti-explorerM.war" including the BEAM4SBP framework. It can be downloaded from the following address: `http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Integration/download.html`.

The user has to copy this file into the webapps directory of Tomcat server and run the startup.bat from the bin folder. When Tomcat is started, Activiti Explorer becomes accessible via this new URL: `http://localhost:8080/activiti-explorerM`. Then, the user logs in with Kermit/Kermit.

After clicking on the "Manage" menu, the user should select the third sub-menu item "Upload with Business Management" from the "Deployments" menu bar (See Figure 6.5).

For example, supposing the file "PurchaseOrderProcess.bpmn20.xml" is uploaded. The purchase Order process is then deployed as shown in Figure 6.6.

At this step, instead of deploying this process lonely, the selected file will be the input of the plug-in BEAM4SBP in order to generate the management process. The output file of the plug-in is called "sortie.bpmn20.xml" and includes the initial

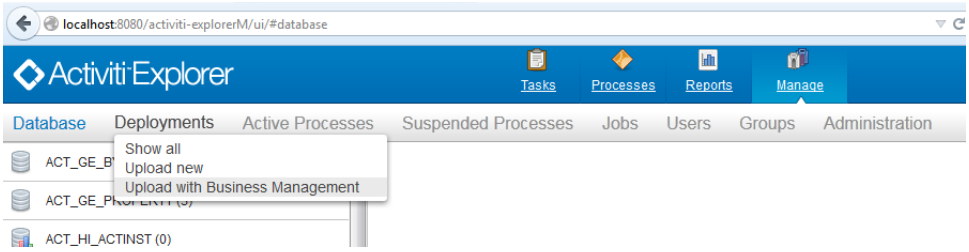


Figure 6.5: Upload with Business Management.

business process together with the management process. Then, this output file will be deployed via Activiti Engine.

As depicted in Figure 6.7, in the "Process Definitions" side, there are two process "id" which represent, respectively, the managed process and the management process.

Finally, the user can run these processes through "Start Process" in the "Processes" menu.

For more details, we developed a web site including informations about the BEAM4SBP, the required tools and some instructions of how to use it. Also, we provide a demonstration video which could be found via this address: `http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Integration/demo.html`

6.3.3 A scenario of the process configuration

The generated management process includes timer events and catch message events (Figure. 3.1). However, while deploying this process into *Activiti*, we noticed that its current versions have not supported catch message events, yet. Hence, we replace the catch message event by a catch signal event and a service task reading e-mail in order to take into account the new values of the service properties. We modeled the business environment as a SBP deployed in turn in the *Activiti* engine. The business environment consists of a service task sending e-mails and signals to alert and throw data to the management process.

The business environment is then deployed and started sending signals and e-mails. The management process catches the corresponding signal triggering a business environment event and reads an email containing the value of the event. A demo and a real test can be found, respectively, in [91] and [92].

Conclusion

This chapter was dedicated to the implementation and validation of our approach of business environment-aware management. We started by implementing the BEAM4SBP framework executing the management process generation algorithms presented in

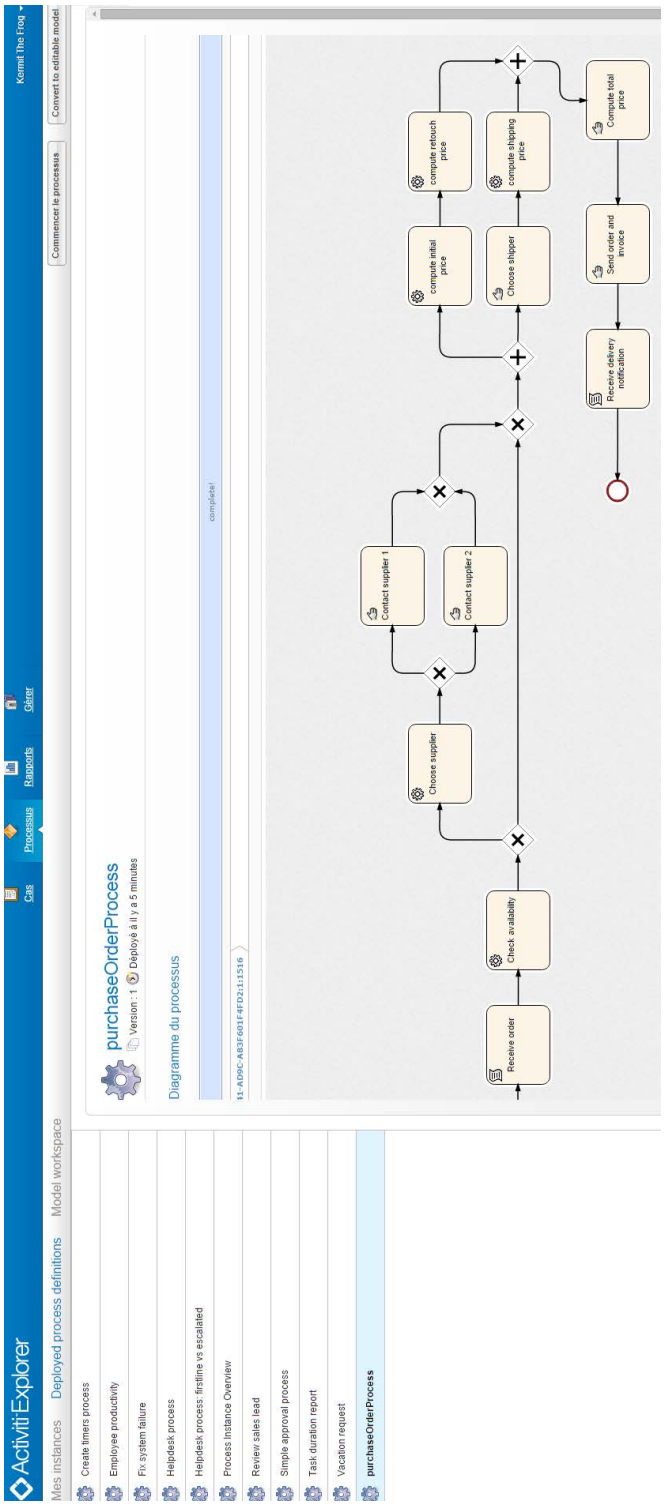


Figure 6.6: Purchase order process.

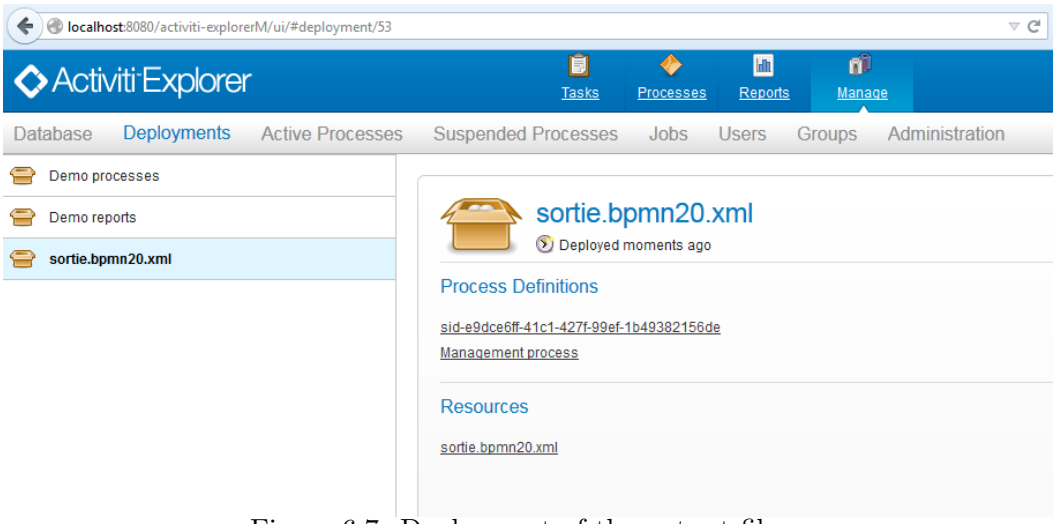


Figure 6.7: Deployment of the output file.

Chapter 5. Then, we implemented the DAT (Dependency Analysis Tool) plug-in for BEAM4SBP. DAT outputs the dependency graph of the managed process. It implements algorithms introduced in Chapter 4. The whole framework is then integrated into the Activiti business process engine. As a result, the BEAM4SBP framework becomes accessible via the Activiti Explorer interface.

Doing so, we prepared the test bed for testing our approach. In order to assess the efficiency and flexibility of our approach compared to the existing BEAM approaches, we will prepare their specific execution environments in Chapter 8. In the next chapter, we will assess these BEAM aproaches qualitatively.

Chapter 7

Qualitative assessment

Introduction

In Chapter 3, we presented our approach for business environement aware management. We used an upper management ontology. In this chapter, we aim to assess qualitatively our approach compared to the existing BEAM approaches presented in Chapter 2. Indeed, we assess the expressiveness of each approach with respect to the real world interactions. Each approach is based on a modeling technique (e.g. business processes (BPs), business rules (BRs), etc.). The expressiveness evaluation of a modeling technique is handled based on a representation theory. In the following sections, we present the adopted representation theory. Then, we detail the adopted evaluation methodology.

7.1 The BWW representation theory

According to zur Muehlen and Indulska [39], a representation theory can be used as a benchmark to make predictions about the capabilities of a grammar to provide complete and clear representations of real world interactions. In this study [39], authors studied the following theories: Chisholm's ontology [93], the Enterprise ontology [94] and the Bunge-Wand-Weber ontology [95, 96].

They argued that the Bunge-Wand-Weber (BWW) is more adapted for Information System (IS) domain. Indeed, the use of BWW is motivated and justified by: (1) its considerable level of maturity [97], (2) its set of constructs that are considered as necessary and able to accurately describe the structure and behavior of the real world, and (3) its success in the evaluation of over thirty analysis projects that spanned various representational grammars [98]. Indeed, many of the most popular modeling techniques have now been discussed in the light of their comparability with respect to the BWW models [98]. Including but not limited to, we mention: Unified Modeling Language (UML)([99]), Merise [97], Petri Nets [100], ebXML Business Process Specification Schema (ebXML BPSS) [98], Web Service Choreography Interface (WSCI) [98], etc.

As for us, we use the BWW representation model (BWW model for short) to evaluate the expressiveness of modeling techniques used in the mentioned BEAM approaches (see Section 2.1). The BWW model defines a set of constructs which can be clustered into four main groups represented by bold rectangles in Figure 7.1: things and their properties, states of things, events and transformations occurring on things, and systems structured around things.

Our objective is not to evaluate a management approach by comparing it to BWW independently of other approaches. Our objective is rather to compare three management approaches that use different modeling languages. To do so, we should map the concepts of these languages into a common ontology. Since the concepts of the considered approaches are not close enough to each other, we have to use a rather generic ontology to allow such mapping and make the comparison possible and feasible. In addition to that, BWW was used successfully for this purpose and particularly to compare BPMN and rule-based languages [39]. Besides, results of many BWW-based studies indicate that the model is good enough to be considered as basis to study the representational capabilities of conceptual modeling languages (for example, see [101, 102, 103, 104] among others).

7.2 The adopted evaluation methodology

In order to determine the expressive power of BEAM approaches, our evaluation strategy consists of three phases. The first phase is the representational analysis of each modeling technique used in the imperative and declarative BEAM approaches presented in Chapter 2, Section 2.1. It consists in comparing the constructs of the BWW model with the constructs of the considered modeling languages. The second phase compares between the representation analyses of the adopted modeling techniques (i.e. BPs, BRs and the Upper management ontology (UMO)). Based on the obtained results, the third phase establishes an overlap analysis consisting in evaluating the conbination of modeling techniques used in declarative and imperative BEAM approaches. Hence, we evaluate the expressiveness of hybrid BEAM approaches introduced in Section 2.1.

7.2.1 Representational analysis

The aim of this step is to provide a rigorous evaluation of the expressiveness of a modeling technique (e.g. BPMN, SBVR, UMO, etc.), used in imperative or declarative approaches, with respect to the BWW model. To achieve this goal, we follow the reference methodology proposed in [105] providing strategic guidelines. These guidelines make more efficient the analysis procedure and increase objectivity.

As stated by Wand and Weber [95], two main evaluation criteria may occur after

a representational anaysis, ontological completeness and ontological clarity. On one hand, the ontological completeness is violated if there exists a construct deficit. The construct deficit happens once one construct in the BWW model does not have any correspondence construct in the evaluated modeling technique. It represents one to zero mapping (1:0). On the other hand, the ontological clarity is determined by identifying the overload, redundancy and excess construct rates. The construct overload occurs when two or more distinct constructs of the BWW model can be compared to the same construct of the evaluated modeling technique. In other words, one construct of the modeling technique can be used to represent many constructs of the BWW model. It can be described by (m:1) relationship mapping. The construct redundancy results when one construct of the BWW model can be compared to two or more distinct constructs of the evaluated modeling technique. This means that many constructs of the modeling technique can be used to represent one construct of the BWW model. It depicts (1:m) relationship mapping. The construct excess results when the BWW model has no constructs that can be compared to some constructs of the evaluated modeling technique. It represents zero to one mapping (0:1).

In order to compare a modeling language to BWW model, the Rosemann's methodology [105] consists in representing the modeling language and BWW model in meta-models using a common meta-language (e.g. UML). This is done in order to overcome the lack of understandability, lack of comparability, lack of guidance, lack of objectivity, etc. Hence, by using a common meta-language, we are able to easily make comparison between models constructs while increasing the rigor, the overall objectivity and the level of detail of the representational analysis.

Figure 7.1 shows a meta-model of the BWW model [7]. It expresses the complete representation model in the UML class diagram notation.

Based on both meta-models of BWW and the evaluated modeling technique, two mappings must be created [95]:

- **Representation mapping:** maps each construct in the BWW model to its corresponding construct in the evaluated modeling technique.
- **Interpretation mapping:** requires mapping each construct in the evaluated modeling technique to its corresponding construct in the BWW model.

In order to evaluate the declarative and imperative approaches presented in Section 2.1, we consider 7 corresponding modeling languages: 5 business rule modeling languages (R2ML, SRML, PRR, SBVR, SWRL), the BPMN business process modeling language and the upper management ontology (UMO).

In the following, we exemplify the representational analysis by considering UMO as the evaluated modeling technique. We create a meta-model of the UMO using the UML class diagram to facilitate the mapping (Figure 7.2). This meta-model is then mapped with the BWW meta-model shown in Figure 7.1.

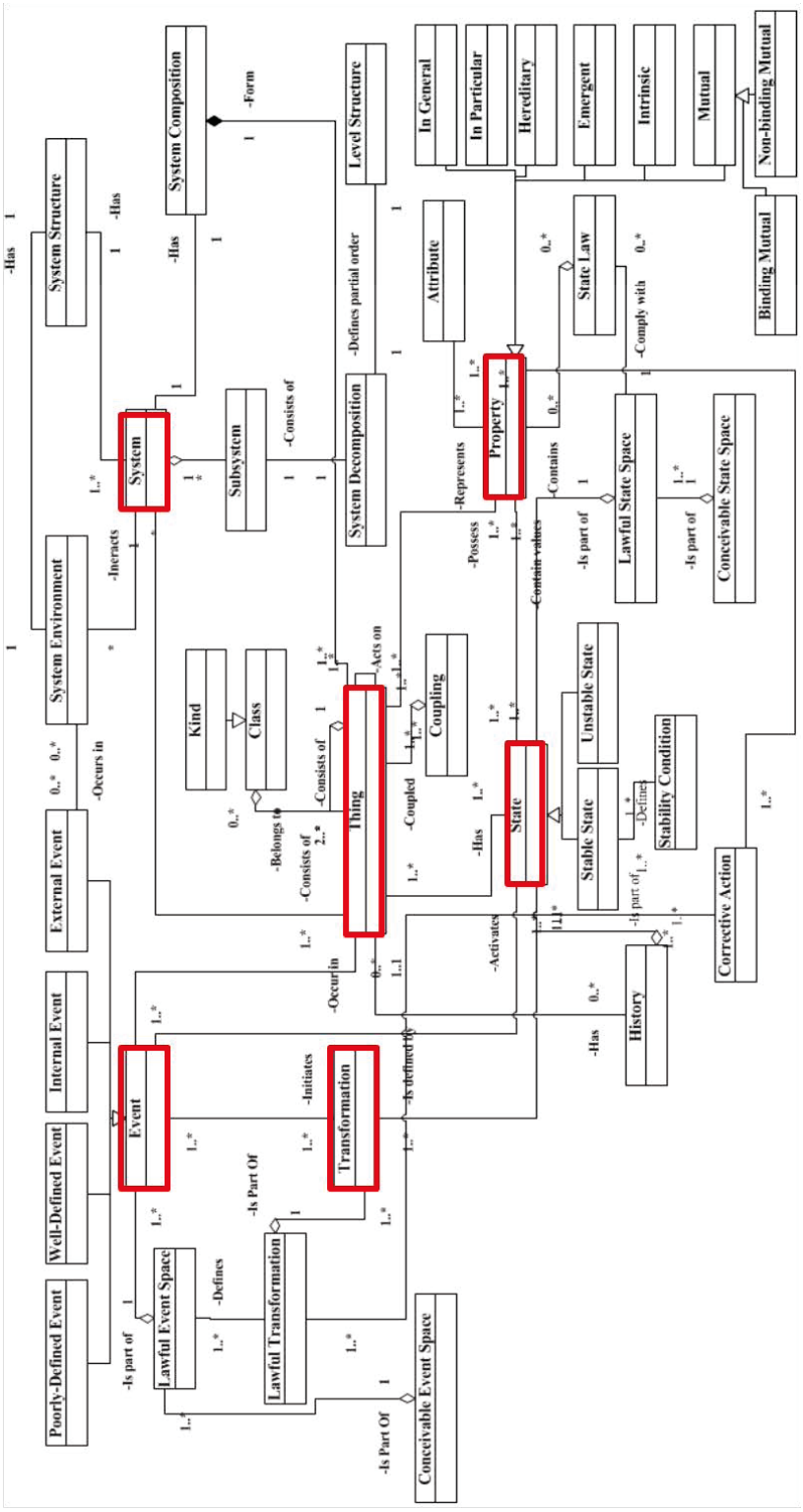


Figure 7.1: BWW meta-model [7].

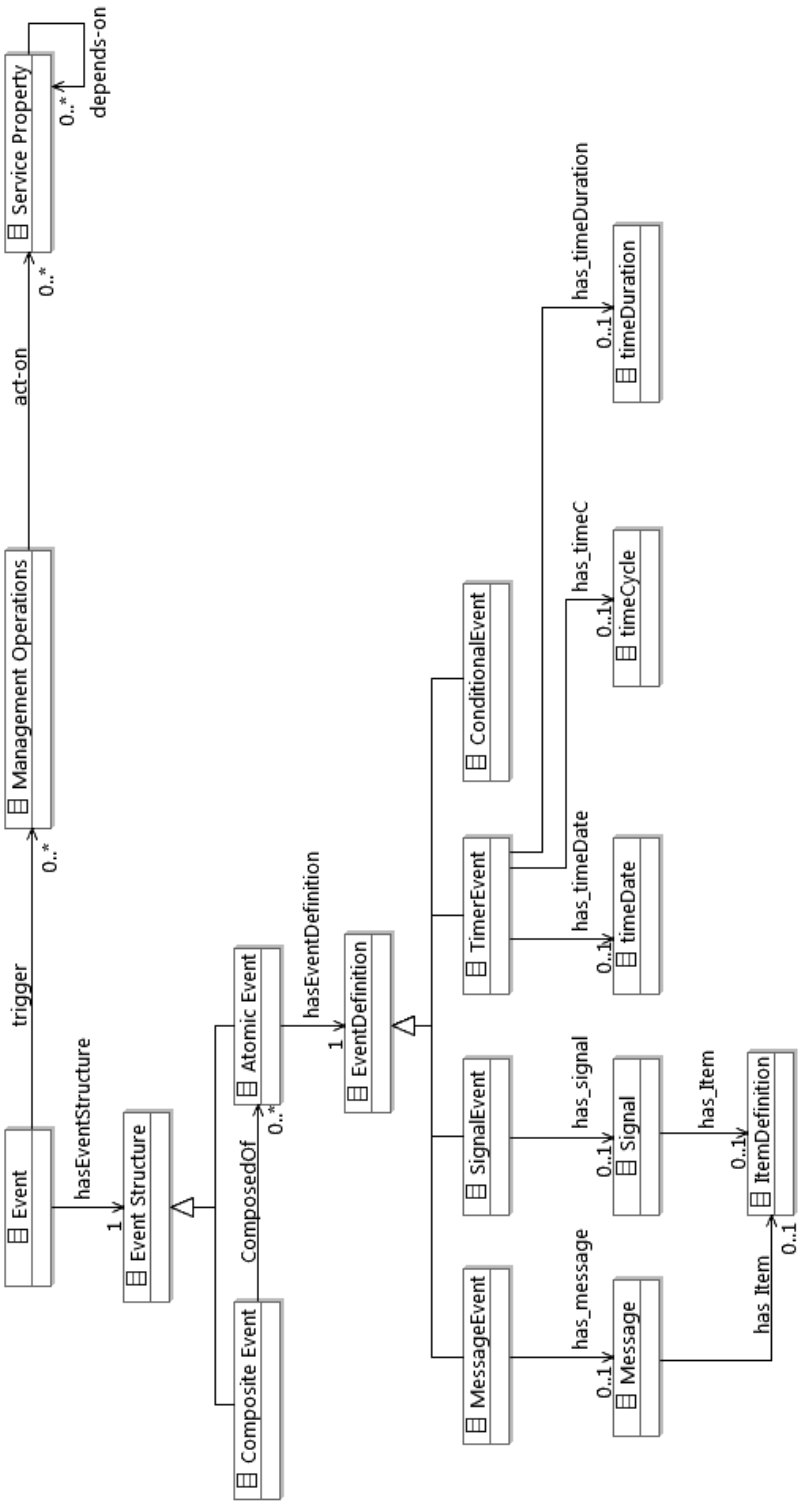


Figure 7.2: UMO meta-model.

According to the two mappings carried out above, Table 7.1 summarizes our results of ontological completeness and clarity for UMO expressed as rates. The results of the mappings are as follows: the construct deficit of the UMO is 57.15% which means that its completeness represents 42.85%. Indeed, 15 constructs among 28 of BWW are in UMO. These constructs are enumerated in the grey column of Table 7.2. The correspondence between BWW model constructs and the constructs in UMO are rated as supported (+) or no supported (–). For the ontological clarity, despite the high value of the construct excess (64.7%, i.e. 18 constructs in UMO over 28 do not exist in BWW meta-model), it becomes balanced thanks to the low value of construct redundancy (11.76%, i.e. 3 costructs in BWW are represented by many constructs in UMO meta-model) and overload (35.29%, 9 constructs among 28 of BWW are compared to one construct in UMO). Overall, these results show that the UMO has an acceptable ontological expressiveness.

By the same way, the representational analyses of the considered modeling techniques are handled. The mapping result is represented in Table 7.2. The correspondence between BWW model constructs and the constructs of the evaluated technique are rated as supported (+) or no supported (–). In the following subsection, we compare these representational analyses in order to identify the BEAM approach with the highest expressive power.

	Ontological Completeness	**Ontological Clarity**		
Deficiency	**Deficit**	**Redundancy**	**Overload**	**Excess**
Rate	57.15%	11.76%	35.29%	64.7%

Table 7.1: Results of ontological completeness and clarity of the UMO.

7.2.2 Comparison of representational analyses of adopted modeling techniques

In this section, we compare the representational analysis of adopted modeling techniques (i.e. BPs, BRs and UMO). The comparison is based on previous representation analyses conducted by different researchers. Most notably, we select the representational analyses of SRML 2001, PRR v1.0 and SBVR 2006 from [39]. The techniques R2ML v0.5, SWRL v1.0 and BPMN v2.0 are adopted from [106]. Then, we compare the findings of UMO to those of business rule languages and business process languages. This comparison is handled with respect to the BWW model (see Table 7.2).

Based on the results given in Table 7.2, BPMN v2.0 offers the highest completeness value with a score of 19/28 such that 28 represents the number of constructs of BWW (67.85%). In the second place, the UMO's score is equal to 12/28 (42.85%) followed

by R2ML v0.5 and SRML 2001 with the same score of 10/28 (35.71%). After them, PRR v1.0 and SBVR 2006 cover only of 7/28 (25%), and finally SWRL v1.0 with the lowest score (21.42%) with respect to the analyzed languages.

Generally, all the representational analyses depicted in Table 7.2 on the following page have scores which are so far away from the total which imply their incompleteness. For this reason, we resort to the next phase consisting in the overlap analysis (see Section 7.2.3).

	R2ML v0.5	**SRML 2001**	**PRR v1.0**	**SBVR 2006**	**SWRL v1.0**	**UMO**	**BPMN v2.0**
THING	+	+	–	+	+	+	+
PROPERTY	+	+	+	+	+	+	+
CLASS	+	–	+	+	+	+	+
KIND	–	–	–	–	–	–	+
STATE	+	+	–	–	–	+	–
CONCEIVABLE STATE SPACE	–	+	–	–	+	+	–
LAWFUL STATE SPACE	–	+	+	–	–	–	–
STATE LAW	+	+	+	+	–	–	–
STABLE STATE	–	–	–	–	–	+	–
UNSTABLE STATE	–	–	–	–	–	+	–
HISTORY	–	–	–	–	–	–	+
EVENT	+	+	–	–	–	+	+
CONCEIVABLE EVENT SPACE	+	–	–	–	–	+	–
LAWFUL EVENT SPACE	+	–	–	–	–	–	–
EXTERNAL EVENT	–	–	–	–	–	+	+
INTERNAL EVENT	–	–	–	–	–	+	+
WELL DEFINED EVENT	–	–	–	–	–	–	+
POORLY DEFINED EVENT	–	–	–	–	–	–	+
TRANSFORMATION	+	+	–	–	+	+	+
LAWFUL TRANSFORMATION	+	+	+	–	+	–	+
COUPLING	–	–	–	–	–	–	+
SYSTEM	–	+	+	+	–	–	+
SYSTEM ENVIRONMENT	–	–	–	–	–	–	+
SYSTEM COMPOSITION	–	–	+	+	–	–	+
SYSTEM DECOMPOSITION	–	–	–	+	–	–	+
SYSTEM STRUCTURE	–	–	–	–	–	–	–
SUBSYSTEM	–	–	–	–	–	–	+
LEVEL STRUCTURE	–	–	–	–	–	–	+
SCORE /28	10	10	7	7	6	15	19
COMPLETENESS RATE	35.71%	35.71%	25%	25%	21.42%	42.85%	67.85%

Table 7.2: Summary of the representation analyses of R2ML, SRML, PRR, SBVR, SWRL, UMO and BPMN

7.2.3 Overlap Analysis

In case none of the studied languages provides a complete representation capability of the real world, the overlap analysis is performed [106]. In the previous section, we argued that declarative and imperative approaches are not sufficient to express the real world interactions. In this phase, our goal is to evaluate the expressiveness of hybrid approaches presented in Section 2.1. We aim to identify the best combination of business process modeling language, business rule modeling language and the upper management ontology. We select BPMN v2.0 as a business process modeling language. We combine BPMN with the 5 business rule modeling languages as well as UMO presented in Table 7.2.

According to [107], two important characteristics are useful for evaluating these combinations, with respect to the BWW model:

- **Maximum Ontological Completeness (MOC):** represents the maximum ontological expressiveness afforded by a combination of modeling languagees.

- **Minimum Ontological Overlap (MOO):** occurs when a construct of BWW model has a correspondence in both modeling languages that make the combination.

MOC and MOO are computed based on symmetric difference, intersection and relative compliment as defined in [39].

BPMN v2.0 +	**R2ML v0.5**	**SRML 2001**	**PRR v1.0**	**SBVR 2006**	**SWRL v1.0**	**UMO**
MOC	82.14%	82.14%	75%	71.42%	71.42%	85.71%
MOO	21.42%	21.42%	17.85%	21.42%	17.85%	25%

Table 7.3: Comparison of overlap analyses results

Based on the results shown in Table 7.3, the combination of BPMN v2.0 with UMO offers the highest ontological completeness with a score of 24 from 28; that is 85.71%. Even though its MOO is the highest value, it remains quite comparable with respect to the other combinations. Indeed, MOC is proportional to MOO.

Figure. 7.3 shows a synthetic histogram of the best modeling techniques in each BEAM approach, according to their scores of completeness with respect to the BWW model. The results shown herein are extracted from Tables 7.2 and 7.3.

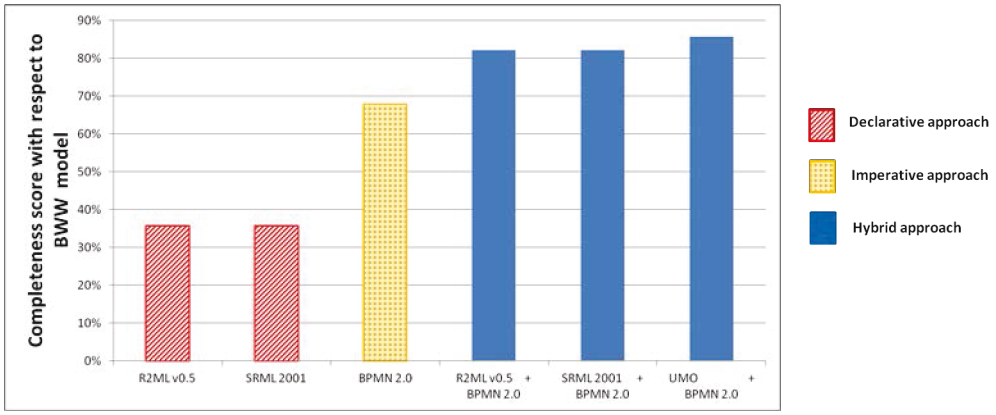


Figure 7.3: Completeness score of the assessed BEAM approaches.

According to the results shown in Figure. 7.3, the hybrid approaches offer the highest scores. Then, the imperative approach followed by the declarative approach.

Conclusion

In this chapter, we conducted a qualitative evaluation comparing the expressivenes of existing BEAM approaches. More precisely, we compare our approach to hybrid approaches integrating business processes and business rules. Doing so, we answered the following research question: which are more expressive the business rules or our upper management ontology?

In the next chapter, we assessed our approach quantitatively with respect to the presented BEAM approaches by experimenting their efficiency and studying their flexibility.

Chapter 8

Quantitative assessment

Introduction

As companies are becoming much more involved in business environments, they hope to become more aware of business impact of their decisions. Hence, choosing an adequate BEAM approach to address critical business opportunities and challenges facing companies remains arguably an awkward task. Indeed, with more competitive markets (i.e. subdued demand, competitive pricing pressures and dynamic business environment changes) companies must seek the efficiencies needed to protect them. Flexibility is also fundamental to deliver high levels of customer service and satisfaction. The problem with business process flexibility is that the changes required are often unknown until the need for that change arises. Besides, companies aim to use existing business process standards in order to be feasible in practical use.

Thereby, this chapter is dedicated to assess the efficiency and flexibility of our approach compared to three other approaches: imperative, declarative and hybrid approaches based on integrating rules in BPs. Although there are existing research works in the area of variability in processes [18, 5, 11, 4, 25, 2, 3] that may be assessed and we believe on their importance, the BEAM approaches discussed in this chapter are complex enough in themselves to deserve separate treatment.

Our experimentation work was achieved in three stages: (8.1) preparing the test-bed, (8.2) testing the efficiency of BEAM approaches and (8.3) studying their flexibility.

8.1 Preparing the test-bed

In order to assess the BEAM approaches, a business process engine and a rule engine are required. We are aware that the choice of these engines can influence the efficiency values. However, we believe that this influence remains limited since it does not radically change the ratio of comparisons between approaches. We opted for *Activiti* process engine and *Drools* rule engine since they are available, open source and widely used by companies [84, 108].

On one hand, *Activiti* [6] is a framework that provides an environment for designing, implementing and running processes described in BPMN 2.0. It is an open source project distributed under the Apache license. On the other hand, *Drools* is a Business Rules Management System (BRMS) based on forward and backward chaining inference engine. It uses an optimized implementation of the Rete algorithm called ReteOO (Rete algorithm for Object-Oriented systems).

Each BEAM approach requires a specific environment to be tested.

The suitable environment to test the imperative approach is a process engine. Hence, we select the *Activiti* engine. We start by writing the business process in BPMN then we deploy and execute it in Activiti. Then, we add to this process, management actions and events in order to re-depoy and re-execute it.

For the declarative approach which is purely based on rules, we need a rule engine. We choose the well-known *Drools* rule engine [109]. The project of Drools is an open source written in Java. We use precisely Drools Workbench [110], a web application composed of all Drools related editors, screens and services. It is equivalent to the old Guvnor [111]. Drools Workbench includes Drools Expert (business rule engine) and Drools Fusion (Complex Event processing features) representing an environment to author, test and deploy rules. As shown in Figure 8.1, the rule engine applies the rules to the facts. Hence, it takes as inputs: facts, a fact model and a set of rules. The facts are the data records to be processed. The fact model tells the rule engine how to interpret the facts. It contains data records as well as their setters and getters. The rules use these facts to tell the rule engine what actions to take when certain conditions are met. Hence, we start by creating the JAR file containing the fact model consisting of Java classes used by rules. The JAR file is then uploaded into the Drools Workbench. Afterwards, we create a set of rules written based on the DRL (Drools Rule Language). Finally, we run the scenario by introducing facts and expected rules. For example, for our running example we introduce values of unitPrice and quantity. We also expect that the discount rule will be executed at least once.

Finally, we test two different hybrid approaches: (i) The first one consists in integrating business rules with business processes. Therefore, we need two engines for running rules and processes at the same time. For this purpose, we integrate *Drools* with *Activiti*. Doing so, we create a maven project in eclipse including the business process written in BPMN, the fact model and the rules related to the business rule tasks. Then, we test based on the Junit. (ii) The second one represents our approach. We have already prepared our environment for testing our hybrid approach by integrating the plug-in BEAM4SBP into Activiti (see Chapter 6).

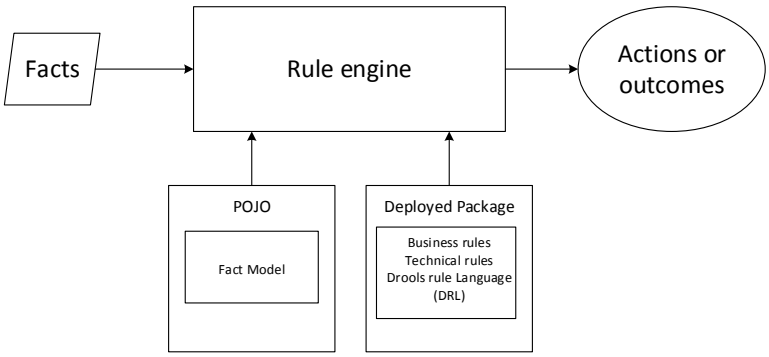


Figure 8.1: The test environment for declarative approaches.

8.2 Testing efficiency

We characterize a given management approach as efficient when the management overhead of the management on the process performance is limited. Consequently, comparing the efficiency of two or more management approaches consists in comparing their management overheads.

Hence, for each test we perform the following steps:

1. **Step 1**: Calculate two different execution times (t_1 and t_2): t_1 is the execution time of the initial BP without including the management time and t_2 represents the execution time of the whole process including the management time.

2. **Step 2**: Compute the *Overhead* using the values of t_1 and t_2

$$Overhead = t_2 - t_1$$

3. **Step 3**: Express the *Overhead* in form of percentage ($Overhead_p$) relatively to the whole execution time t_2. The goal of this step is to provide a clear idea about the rate of the time taken for the management.

$$Overhead_p = \frac{Overhead \times 100}{t_2}$$

In the following, we start by testing efficiency value of each BEAM approach using the running example. Then, we compute efficiency values using control-flow BPMN patterns.

8.2.1 Testing using the running example

The running example presented in Section 1.2 is implemented and experimented in the four considered BEAM approaches.

Figure 8.3 depicts a synthetic histogram representing the average value of $overhead_p$ (called $Overhead_{AVG}$) through running 20 tests while varying the input "quantity" value (we selected 5 values: 10, 30, 50, 70, 80, 90 (Figure 8.2)). It shows that the value of the management spent in the imperative approach is equal to 9.35%, while, the management in the declarative approach takes 45.10%. This value shows the time-consuming problem with the declarative approach. The hybrid approaches in which the configuration is described based on rules has an $overhead_p$ equal to 39.89%, and the hybrid approach with a configuration based on management process has an $overhead_p$ equal to 20.08%. The difference between these two latter values is significant since the configuration of the second hybrid approach is described imperatively. This leads to a better value of efficiency.

8.2.2 Testing using BPMN patterns

An obvious approach of experimentation consists in developping a set of real BPs of a large public dataset (e.g. the dataset shared by the Business Integration Technologies (BIT) research group at the IBM Zurich Research Laboratory [1]). However, the results based on these experimentations could be confusing since they depend on the manner of implementation of these processes according to the four experimented approaches. Hence, the efficiency value can come from the implementation of these processes rather then the evaluated BEAM approaches.

As for us, we consider another way of testing these approaches based on the following assumption: business processes can be built based on workflow patterns [80]. The result of comparision of BEAM approaches on these patterns based on their behavior could be very close to the result of comparison of these approaches on a set of processes which are compositions of patterns.

We have conducted 12 tests. Indeed, we experimented the three basic patterns ((1) sequence, (2) parallel split + synchronization and (3) exclusive choice + simple merge) on the implementations of the four considered BEAM approaches based on the purchase order ontology. In fact, these BPMN patterns are commonly used specially with structured workflows [112]. Figure 8.4 shows results of these experimentations. More details are given in the following web site: `http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Experimentation/ontology1.html`

Figure 8.4 illustrates the same shape as Figure 8.3. Indeed, the declarative approach represents always the higher overhead value: for the sequence pattern 70%, for the parallel split and synchronization pattern 67.74%, for the exclusive choice and simple merge 60.86%. The hybrid approach in which the configuration is described based on rule represents the overhead values 39%, 40.65%, 36.86% respectively for the sequence pattern, the parallel pattern and the exclusive choice pattern. Our hybrid

[1] http://www.research.ibm.com/labs/zurich/csc/

Figure 8.2: Histograms of BEAM approaches.

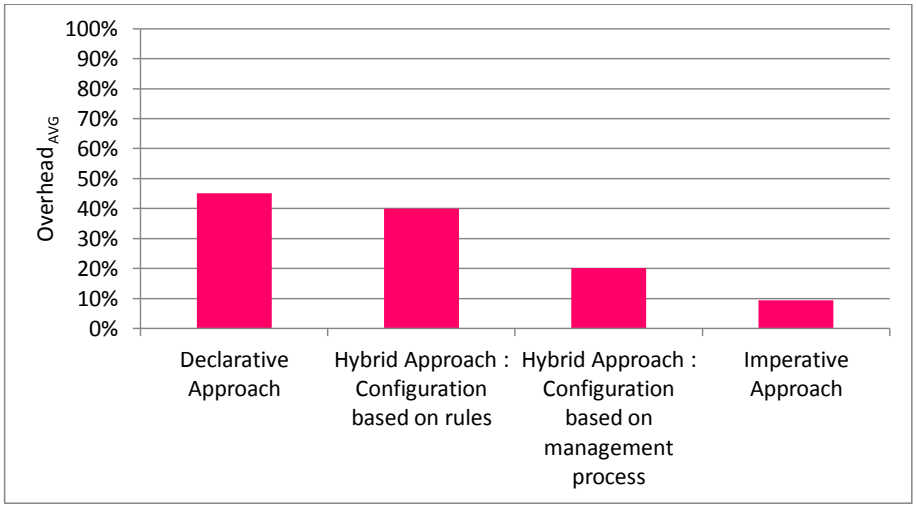


Figure 8.3: Histograms of BEAM approaches.

approach where the configuration is described by a management process has overhead values of: 33.31% for the sequence pattern, 33.55% for the parallel split pattern and 33.39% for the exclusive choice pattern. Finally, the impearative approach represent the best overhead values. In fact, for the sequence pattern we obtained 4%, while for the parallel split pattern and exclusive choice pattern we get respectively 5.42% and 5.78%.

In addition to that, we conducted 12 tests based on the three basic BPMN patterns mentioned above. These tests are based on the pathology ontology. More details are given in the following web site: `http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Experimentation/ontology2.html`. Hence, we notice that both resulting histograms based on the purchase order ontology and the pathology ontology have similar shapes.

As a result, the lower the $Overhead_p$ value is, the more efficient the approach will be. Therefore, as expected, the imperative approach is the most efficient one and the declarative approach is the most time-consuming approach. The hybrid approaches are classified between these two extremities. We recall that our hybrid approach, describing its configuration based on the management process, represents a good value of efficiency thanks to modeling the management process imperatively.

8.3 Studying flexibility

Flexibility is the second quantitative criterion upon which we assess the BEAM approaches. It is the ability to modify the process model at design, deployment and

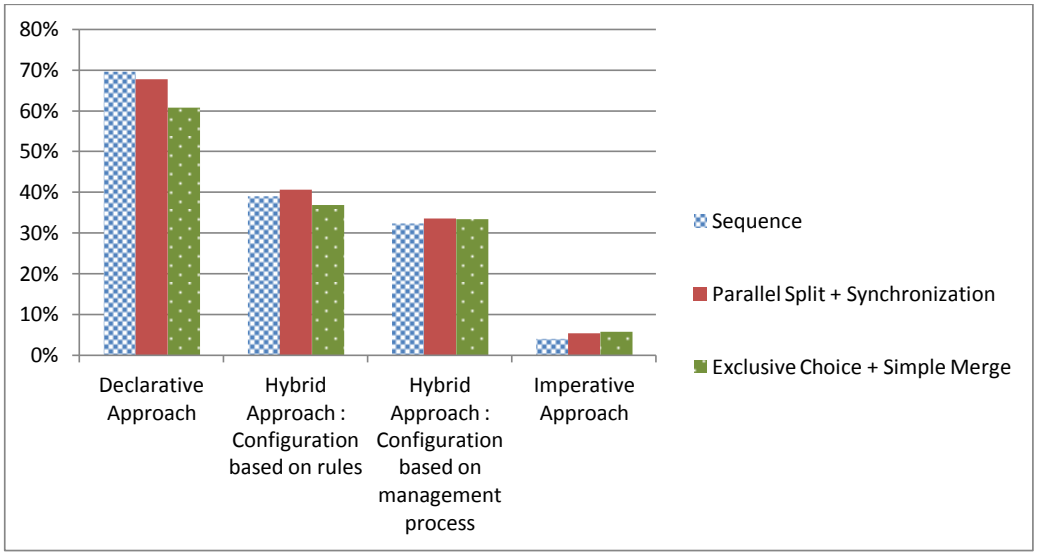


Figure 8.4: Histograms of BEAM approaches with respect to BPMN control flow patterns.

run time. While implementing the four assessed BEAM approaches, we argue the ascertainments summarized in Table 8.1.

Table 8.1: Flexibility results.

	Declarative approach	**Hybrid approach**		**Imperative approach**
		CBR	**CBMP**	
Design time	+	+	+	+
Deployment time	+	+/-	+/-	–
Run time	+	+/-	+/-	–

The management in declarative approaches is made at design time, deployment time or run time as it is based on inference on rules. Indeed, a business rule may be added, modified or deleted at any time. Hence, declarative approaches have a high level of flexibility.

As for the hybrid approaches where the configuration is described based on rules (CBR), the management is implemented using rules that are integrated in some activities (called business rule tasks) in the managed process. Then the management consists in adding, modifying or deleting business rules in business rule tasks. After

the design, this ability of modifying the management is limited to business rule tasks. The management can affect neither non business rule tasks nor control flow in the managed process.

Regarding the hybrid approach in which the configuration is based on a management process (CBMP), the management consists in annotating BP using the upper management ontology (UMO) and business (domain) ontology. Beside the design time, in this approach, the management process is generated at deployment time. At the run-time, it allow different scenarios of management (the different management paths in the management process). Nevertheless, this way of management is not as flexible as the one of declarative approaches since the management of this current approach consists in a limited set of management paths. In fact, declarative approaches can deal with much more possibilities of rules.

In imperative approaches, the management consists in over-specifying BPs at design time. When designed a managed process cannot be changes neither at deployment time nor at run-time.

8.4 Guidelines for choosing the adequate BEAM approach

Both efficiency experimentations and flexibility studies show that the declarative approach represents a high level of flexibility but it is very time consuming. On the contrary, the imperative approach as intended is very efficient, however, it is very rigid. Hence, hybrid approaches prove good opportunities in practical if conciliating between flexibility and efficiency is a need.

We think that the choice of a BEAM approach should be based on the dependency of the business process behavior, on one hand, and the frequency of changes in the business environment, on the other hand.

The frequenter the change is, the higher the motivation to use a declarative approach. For example, a declarative approach would be adequate to manage social-based business processes. In such processes, human intervenes frequently to enact and change the behavior of processes based on the behavior of other human activities. Control flow and behaviors of activities of social business processes can be changed frequently, which needs to adopt a declarative approach.

Contrary, an imperative approach is adopted when the business process tends to be insensitive to the business environment change. Business processes of embedded systems are a good example of processes where it is convenient to adopt an imperative BEAM approach.

A hybrid approach, in which the configuration is described based on rules, is used when the behavior of some activities could be modified, whereas the control flow remains unchanged. It is the case of administrative processes where the control flow is defined by law or decree and behaviors of activities depend on citizen situations.

Last but not least, a hybrid approach where the configuration is described by a management process can be adopted when the management of control flow and activities of business processes depends on business environment. It is the case of hospital service processes where activities and control flow depend on the situation of other hospital services and patients situations considered as business environments.

Conclusion

In this chapter, we experimented the efficiency and flexibility of our approach compared to the imperative, declarative and hybrid BEAM approaches. To realize our experimentations, we started by preparing the test bed of each BEAM approach. The comparision of these approaches are based on a running example and control flow BPMN patterns.

Hence, our quantitative evaluation offers to designers or system administrators, guidelines that may help them to decide about the adequate modeling approach to adopt for a given context or case. Indeed, we showed that there is not a best approach that might be applied in any case and we provided guidelines to choose the suitable approach for each case/context.

Conclusion and future work

Conclusion

In this thesis, we proposed a novel hybrid approach for managing SBPs against highly dynamic business environments. This approach conciliates between imperative and declarative approaches while addressing the following issues: preserving standards for describing SBPs and minimizing the designer efforts. Our approach consists in generating, at deployment time, a management process for a managed BP connected to it allowing its monitoring and configuration. The management process generation is performed thanks to a semantic module and a control dependency analysis module. The first module involves an upper management ontology describing relationship between SBPs and business environments. While, the second module describes an explicit description of the managed BP by identifying control dependencies to facilitate the organization of the whole management process. We also tested the feasibility of our approach and experimented its efficiency and flexibility.

Indeed, meeting the market requirements within a highly dynamic business environment remains a challenge for companies. Hence, choosing the adequate business environment management approach represents a challenging issue. Therefore, we handled, throughout this thesis, qualitative and quantitative evaluations of existing types of business environment-aware management approaches. The qualitative evaluation revealed the expressiveness of each approach based on the BWW representation theory. As to the quantitative evaluation, we experimented the efficiency and studied the flexibility based on basic control-flow patterns. Furthermore, we verify the feasibility of existing approaches in practical use by preserving or not the business process standards. As a result, through this thesis, we argue that declarative approaches are very time consuming but are very flexible. Thus, declarative approaches are suitable for highly frequent management changes (e.g. social based processes). Imperative approaches, by contrast, are proved very efficient, however, they are too rigid due to over-specifying processes at design time. It is then nearly infeasible to change the management rules at run-time. Hence, the imperative approaches are suitable when the business process is insensitive to the business environment (e.g. processes of embedded systems). The evaluation, both qualitative and quantitative, prove good opportunities for hybrid approaches in practical use. On one hand, the hybrid approach where the configuration is described based on rules, is used in case some activities change their behaviour by opting for business rule tasks (e.g. administrative processes). On the other hand, the hybrid approach in which the configuration

is based on management process, is used if the behaviour of some activities as well as the process control-flow vary (e.g. hospital service processes).

Future work

The upper management ontology details events and service properties. The management operations are not yet detailed. Indeed, at this level, our approach involves only getters and setters as managing operations. Thus, we plan to specify a composition for managing operations given by the service provider. As for the experimentation, we aim at testing other BPMN control-flow patterns. Besides, we foresee to test these patterns based on a large ontology. As for long-term perspective, we aim at using our approach for managing multi-tenant BPs. For a given modeled BPs having one tenant, there is a dedicated management.

Bibliography

[1] Mohamed Boukhebouze. *Gestion de changement et vérification formelle de processus métier : une approche orientée règles*. PhD thesis, L'Institut National des Sciences Appliquées de Lyon, 2010.

[2] Milan Milanovic and Dragan Gasevic. Towards a language for rule-enhanced business process modeling. In *IEEE International Enterprise Distributed Object Computing Conference*, pages 64–73, 2009.

[3] Ran Cheng, Shazia Wasim Sadiq, and Marta Indulska. Framework for business process and rule integration: A case of bpmn and sbvr. In *International Conference on Business Information Systems*, pages 13–24, 2011.

[4] Yiwei Gong and Marijn Janssen. Creating dynamic business processes using semantic web services and business rules. In *ICEGOV*, pages 249–258, 2011.

[5] Michiel Koning, Chang ai Sun, Marco Sinnema, and Paris Avgeriou. Vxbpel: Supporting variability for web services in bpel. *Information & Software Technology*, 51(2):258–269, 2009.

[6] Activiti. `http://www.activiti.org/`.

[7] Ludmila Penicina. Choosing a BPMN 2.0 Compatible Upper Ontology. In *ThinkMind // eKNOW 2013, The Fifth International Conference on Information, Process, and Knowledge Management*, pages 89–96, 2013.

[8] Patrick Feldbacher, Peter Suppan, Christina Schweiger, and Robert Singer. Business process management: A survey among small and medium sized enterprises. In *S-BPM ONE - Learning by Doing - Doing by Learning - International Conference, S-BPM ONE 2011*, pages 296–312, 2011.

[9] Rafael Aschoff and Andrea Zisman. Qos-driven proactive adaptation of service composition. In *ICSOC*, pages 421–435, 2011.

[10] Claudio Bartolini and Cesare Stefanelli. Business-driven it management. In *Integrated Network Management*, pages 964–969, 2011.

[11] Bang Ouyang, Farong Zhong, and Huan Liu. An eca-based control-rule formalism for the bpel process modularization. *Procedia Environmental Sciences*, 11(1):511–517, 2011.

[12] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business process management: A survey. In *International Conference Business Process Management*, pages 1–12, 2003.

[13] Dimitris Karagiannis. Business process management: A holistic management approach. In *Information Systems: Methods, Models, and Applications - 4th International United Information Systems Conference*, pages 1–12, 2012.

[14] Zibin Zheng. *QoS management of web services*. PhD thesis, The Chinese University of Hong Kong, 2011.

[15] Branimir Wetzstein, Zhilei Ma, and Frank Leymann. Towards measuring key performance indicators of semantic business processes. In *BIS*, pages 227–238, 2008.

[16] Adela del Río-Ortega, Manuel Resinas, and Antonio Ruiz Cortés. Defining process performance indicators: An ontological approach. In *OTM Conferences (1)*, pages 555–572, 2010.

[17] Mohamed Boukhebouze, Youssef Amghar, Aïcha-Nabila Benharkat, and Zakaria Maamar. A rule-based modeling for the description of flexible and self-healing business processes. In *ADBIS*, pages 15–27, 2009.

[18] Florian Gottschalk, Wil M. P. van der Aalst, Monique H. Jansen-Vullers, and Marcello La Rosa. Configurable workflow models. *Int. J. Cooperative Inf. Syst.*, 17(2):177–221, 2008.

[19] Carlo Ghezzi and Sam Guinea. Run-time monitoring in service-oriented architectures. In Luciano Baresi and Elisabetta Di Nitto, editors, *Test and Analysis of Web Services*, pages 237–264. Springer, 2007.

[20] Fabio Casati, Eric Shan, Umeshwar Dayal, and Ming-Chien Shan. Business-oriented management of web services. *Commun. ACM*, 46(10):55–60, 2003.

[21] Konstantinos Bratanis, Dimitris Dranidis, and Anthony J. H. Simons. Towards run-time monitoring of web services conformance to business-level agreements. In *TAIC PART*, pages 203–206, 2010.

[22] Hans Weigand, Willem-Jan van den Heuvel, and Marcel Hiel. Business policy compliance in service-oriented systems. *Inf. Syst.*, 36(4):791–807, 2011.

[23] Business process model and notation 2.0. http://www.omg.org/spec/BPMN/2.0/.

[24] Bpel: Who needs it anyway?, 2009. http://www.bpm.com/bpel-who-needs-it.html.

[25] Anis Charfi, Tom Dinkelaker, and Mira Mezini. A plug-in architecture for self-adaptive web service compositions. In *ICWS*, pages 35–42, 2009.

[26] The Workflow Management Coalition. Workflow management coalition terminology & glossary, Février 1999.

[27] Riadh Ben Halima, Emna Fki, Khalil Drira, and Mohamed Jmaiel. A large-scale monitoring and measurement campaign for web services-based applications. *Concurrency and Computation: Practice and Experience*, 22(10):1207–1222, 2010.

[28] XPDL, 2009. http://www.xpdl.org/.

[29] OMG. Business Process Modeling Notation, Final Adopted Specification, 2014. `http://www.omg.org/spec/BPMN/2.0/PDF/formal-11-01-03.pdf`.

[30] Nick Russell, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Petia Wohed. On the suitability of UML 2.0 activity diagrams for business process modelling. In *Conceptual Modelling 2006, Third Asia-Pacific Conference on Conceptual Modelling (APCCM 2005), Hobart, Tasmania, Australia, January 16-19 2006*, pages 95–104, 2006.

[31] Workflow Management Coalition Specification. Process definition interface – xml process definition language. Technical report, August 2012.

[32] Wil M. P. van der Aalst, Mathias Weske, and Dolf Grünbauer. Case handling: a new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, 2005.

[33] Marlon Dumas, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. *The FLOWer Case-Handling Approach: Beyond Workflow Management*, pages 363–395. John Wiley & Sons, Inc., 2005.

[34] Maja Pesic and Wil M. P. van der Aalst. A Declarative Approach for Flexible Business Processes Management. In *Proceedings of the 2006 International Conference on Business Process Management Workshops*, BPM'06, pages 169–180. Springer, 2006.

[35] Stijn Goedertier and Jan Vanthienen. Designing compliant business processes with obligations and permissions. In *Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006, Proceedings*, pages 5–14, 2006.

[36] *The Event-driven Process Chain*, pages 105–125. Springer London, London, 2007.

[37] David Hay and Keri Anderson Healy. Defining Business Rules - What Are They Really?. `http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf`, 2000.

[38] Gerd Wagner. Rule Modeling and Markup. In *Proceedings of the First International Conference on Reasoning Web*, pages 251–274. Springer, 2005.

[39] Michael zur Muehlen and Marta Indulska. Modeling Languages for Business Processes and Business Rules: A Representational Analysis. *Information Systems*, 35(4):379–390, 2010.

[40] R2ML. The REWERSE I1 Rule Markup Language Working Group I1. `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=R2ML`.

[41] SRML. Simple Rule Markup Language. `http://xml.coverpages.org/srml.html`.

[42] OMG. Production Rule Representations. `http://www.omg.org/spec/PRR/1.0/PDF`.

[43] SBVR. Semantics of Business vocabulary and Rules. `http://www.omg.org/spec/SBVR/1.2/PDF/`.

[44] Chi P. T. Tran and Hanh Huu Hoang. A Combination of Business Rule and Modeling Languages for Semantic Business Processes Modeling. In *KES-AMSTA*, pages 444–453, 2013.

[45] Mike P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45, 2007.

[46] Raman Kazhamiakin, Salima Benbernou, Luciano Baresi, Pierluigi Plebani, Maike Uhlig, and Olivier Barais. Adaptation of service-based systems. In *Service Research Challenges and Solutions for the Future Internet - S-Cube - Towards Engineering, Managing and Adapting Service-Based Systems*, pages 117–156, 2010.

[47] Ta Nguyen Binh Duong, Xiaorong Li, and Rick Siow Mong Goh. A framework for dynamic resource provisioning and adaptation in iaas clouds. In *IEEE 3rd International Conference on Cloud Computing Technology and Science, CloudCom 2011, Athens, Greece, November 29 - December 1, 2011*, pages 312–319, 2011.

[48] J. Bi, Z. Zhu, R. Tian, and Q. Wang. Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 370–377, July 2010.

[49] R. Han, L. Guo, Y. Guo, and S. He. A deployment platform for dynamically scaling applications in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 506–510, Novembre 2011.

[50] Jon B. Weissman, Seonho Kim, and Darin England. A framework for dynamic service adaptation in the grid: Next generation software program progress report. In *19th International Parallel and Distributed Processing Symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CO, USA*, 2005.

[51] Mourad Amziani, Tarek Melliti, and Samir Tata. Formal modeling and evaluation of stateful service-based business process elasticity in the cloud. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*, pages 21–38, 2013.

[52] Kristof Hamann, Sonja Zaplata, and Winfried Lamersdorf. Process instance migration: Flexible execution of distributed business processes. In *First International Workshop on European Software Services and Systems Research - Results and Challenges, S-Cube 2012, Zurich, Switzerland, June 5, 2012*, pages 21–22, 2012.

[53] Geetika T. Lakshmanan, Paul T. Keyser, Aleksander Slominski, and Francisco Curbera. A business centric monitoring approach for heterogeneous service composites. In *IEEE SCC*, pages 671–678, 2011.

[54] Jacques Philippe Sauvé, Claudio Bartolini, and Antão Moura. Looking at business through a keyhole. In *Integrated Network Management Workshops*, pages 48–51, 2009.

[55] Branimir Wetzstein, Philipp Leitner, Florian Rosenberg, Schahram Dustdar, and Frank Leymann. Identifying influential factors of business process performance using dependency analysis. *Enterprise IS*, 5(1):79–98, 2011.

[56] Branimir Wetzstein, Philipp Leitner, Florian Rosenberg, Ivona Brandic, Schahram Dustdar, and Frank Leymann. Monitoring and analyzing influential factors of business process performance. In *EDOC*, pages 141–150, 2009.

[57] Raman Kazhamiakin, Branimir Wetzstein, Dimka Karastoyanova, Marco Pistore, and Frank Leymann. Adaptation of service-based applications based on process quality factor analysis. In *ICSOC/ServiceWave Workshops*, pages 395–404, 2009.

[58] Mohamed Boukhebouze, Youssef Amghar, Aïcha-Nabila Benharkat, and Zakaria Maamar. Towards an approach for estimating impact of changes on business processes. In *CEC*, 2009.

[59] Milan Milanovic, Dragan Gasevic, and Luis Rocha. Modeling flexible business processes with business rule patterns. In *IEEE International Enterprise Distributed Object Computing Conference*, 2011.

[60] Martijn Zoet, Johan Versendaal, Pascal Ravesteyn, and Richard J. Welke. Alignment of business process management and business rules. In *European Conference on Information Systems*, 2011.

[61] Matthias Galster and Paris Avgeriou. Variability in web services. In Rafael Capilla, Jan Bosch, and Kyo Chul Kang, editors, *Systems and Software Variability Management, Concepts, Tools and Experiences*, pages 269–278. Springer, 2013.

[62] Gary J. Nutt. The evolution towards flexible workflow systems. *Distributed Systems Engineering*, 3(4):276–, 1996.

[63] Olfa Bouchaala, Samir Tata, and Mohamed Jmaiel. A hybrid approach for business environment-aware management of service-based business processes. In *EC-Web*, pages 68–79, 2013.

[64] Olfa Bouchaala, Mohamed Yangui, Samir Tata, and Mohamed Jmaiel. Dat: Dependency analysis tool for service-based business processes. In *International Conference on Advanced Information Networking and Applications*, 2014.

[65] Remco M. Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In *BPM*, pages 48–63, 2009.

[66] Bpel: Who needs it anyway?, 2009. http://www.bpm.com/bpel-who-needs-it.html.

[67] jbpm. http://www.jboss.org/jbpm/.

[68] Chengying Mao. Slicing web service-based software. In *SOCA*, pages 1–8, 2009.

[69] ZhangBing Zhou, Sami Bhiri, and Manfred Hauswirth. Control and data dependencies in business processes based on semantic business activities. In *iiWAS*, pages 257–263, 2008.

[70] Qinyi Wu, Calton Pu, Akhil Sahai, and Roger S. Barga. Categorization and optimization of synchronization dependencies in business processes. In *ICDE*, pages 306–315, 2007.

[71] Ivaylo Spassov, Valentin Pavlov, Dessislava Petrova-Antonova, and Sylvia Ilieva. Ddat: Data dependency analysis tool for web service business processes. In *ICCSA (5)*, pages 232–243, 2011.

[72] Matthias Winkler, Thomas Springer, Edmundo David Trigos, and Alexander Schill. Analysing dependencies in service compositions. In *ICSOC/ServiceWave Workshops*, pages 123–133, 2009.

[73] Olaf Henniger and Hasan Ural. Test generation based on control and data dependencies within multi-process sdl specifications. In *SAM*, pages 189–202, 2000.

[74] Geert Monsieur, Monique Snoeck, and Wilfried Lemahieu. Managing data dependencies in service compositions. *Journal of Systems and Software*, 85(11):2604 – 2628, 2012.

[75] Chun Ouyang, Marlon Dumas, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Jan Mendling. From business process models to process-oriented software systems. *ACM Trans. Softw. Eng. Methodol.*, 19(1), 2009.

[76] Pierre Vigneras. Why bpel is not the holy grail for bpm, 2008. http://www.infoq.com/articles/bpelbpm.

[77] Luciano García-Bañuelos. Pattern identification and classification in the translation from bpmn to bpel. In *OTM Conferences (1)*, pages 436–444, 2008.

[78] Jeanne Ferrante, Karl J. Ottenstein, and Joe D. Warren. The program dependence graph and its use in optimization. *ACM Trans. Program. Lang. Syst.*, 9(3):319–349, 1987.

[79] Judith A. Stafford. *A Formal, Language-Independent, and Compositional Approach to Interprocedural Control Dependence Analysis*. PhD thesis, University of Colorado, 2000.

[80] Wil MP van der Aalst, Arthur HM ter Hofstede, and Nick Russell. Workflow patterns, 2011. http://www.workflowpatterns.com/.

[81] Eclipse modeling framework project. http://www.eclipse.org/modeling/emf/.

[82] Jgrapht. http://jgrapht.org/.

[83] Olfa Bouchaala, Samir Tata, and Mohamed Jmaiel. A Hybrid Approach for Business Environment-Aware Management of Service-Based Business Processes. In *E-Commerce and Web Technologies*, pages 68–79. Springer, 2013.

[84] Tijs Rademakers and Ron van Liempd. *Activiti in Action: Executable Business Processes in BPMN 2.0*. Manning Publications Company, 2012.

[85] Apache Tomcat. `http://tomcat.apache.org/`.

[86] Eclipse. `https://www.eclipse.org/`.

[87] Maven integration for Eclipse. `https://www.eclipse.org/m2e/`.

[88] Apache Maven. `http://maven.apache.org/`.

[89] Apache Ant. `http://ant.apache.org/`.

[90] Mouna Hadj Kacem and Olfa Bouchaala. Technical report for beam4sbp integration, 2015. http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Integration/TechnicalReport.

[91] Integration demo, 2015. http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Integration/demo.html.

[92] Integration real test, 2015. http://www-inf.int-evry.fr/SIMBAD/tools/BEAM4SBP0.2/Integration/download.html.

[93] Roderick M. Chisholm. The basic ontological categories. In *Language, Truth and Ontology*, volume 51, pages 1–13, 1992.

[94] Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. The enterprise ontology. *Knowledge Eng. Review*, 13(1):31–89, 1998.

[95] Y. Wand and R. Weber. On the ontological expressiveness of information systems analysis and design grammars. *Inf. Syst. J.*, 3(4):217–237, 1993.

[96] Yair Wand and Ron Weber. On the deep structure of information systems. *Inf. Syst. J.*, 5(3):203–223, 1995.

[97] Jan Recker, Michael Rosemann, Marta Indulska, and Peter F. Green. Business Process Modeling - A Comparative Analysis. *Journal of the Association for Information Systems*, 10(4), 2009.

[98] Peter Green, Michael Rosemann, Marta Indulska, and Chris Manning. Candidate Interoperability Standards: An Ontological Overlap Analysis. *Data & Knowledge Engineering*, 62(2):274–291, 2007.

[99] Andreas L. Opdahl and Brian Henderson-Sellers. Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model. *Software and System Modeling*, 1(1):43–67, 2002.

[100] Jan C Recker and Marta Indulska. An ontology-based evaluation of process modeling with petri nets. *IBIS-International Journal of Interoperability in Business Information Systems*, 2(1):45–64, 2007.

[101] Andreas L. Opdahl and Brian Henderson-Sellers. Ontological evaluation of the UML using the bunge-wand-weber model. *Software and System Modeling*, 1(1):43–67, 2002.

[102] Peter F. Green and Michael Rosemann. Perceived ontological weaknesses of process modeling techniques : Further evidence. In *10th European Conference on Information Systems, Information Systems and the Future of the Digital Economy*, pages 312–321, 2002.

[103] Jan Recker, Marta Indulska, Michael Rosemann, and Peter F. Green. How good is BPMN really? insights from theory and practice. In *14th European Conference on Information Systems*, pages 1582–1593, 2006.

[104] Jan Recker, Marta Indulska, Michael Rosemann, and Peter Green. Do process modelling techniques get better? a comparative ontological analysis of bpmn. In *16th Australasian Conference on Information Systems*, 2005.

[105] Michael Rosemann, Peter F. Green, and Marta Indulska. A reference methodology for conducting ontological analyses. In *ER*, pages 110–121, 2004.

[106] Vid Prezel, Dragan Gasevic, and Milan Milanovic. Representational Analysis of Business Process and Business Rule Languages. In *1st International Workshop on Business Models, Business Rules and Ontologies at 4th International Conference on Web Reasoning and Rule Systems*, 2010.

[107] Peter F. Green, Michael Rosemann, Marta Indulska, and Jan C. Recker. Complementary Use of Modeling Grammars. *Scandinavian Journal of Information Systems*, 23(1):59–86, 2011.

[108] Mark Proctor. Drools: A rule engine for complex event processing. In Andy Schürr, Dániel Varró, and Gergely Varró, editors, *Applications of Graph Transformations with Industrial Relevance*, volume 7233 of *Lecture Notes in Computer Science*, pages 2–2. Springer Berlin Heidelberg, 2012.

[109] Mark Proctor, Michael Neale, Michael Frandsen, Sam Griffith Jr., Edson Tirelli, Fernando Meyer, and Kris Verlaenen. Drools Documentation. Technical report, 2008.

[110] Drools workbench, 2014. http://docs.jboss.org/drools/release/6.2.0.CR4/drools-docs/html/wb.Workbench.html.

[111] Paul Browne. *JBoss Drools Business Rules*. Packt Publishing, 2009.

[112] Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Christoph Bussler. On structured workflow modelling. In *Advanced Information Systems Engineering, 12th International Conference CAiSE 2000, Stockholm, Sweden, June 5-9, 2000, Proceedings*, pages 431–445, 2000.

université PARIS-SACLAY

ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Gestion sensible au métier des processus métiers à base de services

Mots clés : processus métier, environnement métier, efficacité et flexibilité

Résumé : Face à un environnement métier très dynamique, les entreprises expriment un grand besoin de gestion de leurs processus métiers de point de vue métier. Il existe trois types d'approche de gestion sensible aux changements de l'environnement métier : à savoir les approches impératives, déclaratives et hybrides. Les approches déclaratives sont très flexibles. Cependant, elles sont très couteuses en termes de temps. Contrairement, les approches impératives sont très efficaces mais trop rigide. Les approches hybrides, à leur tour, essaient de concilier entre les approches impératives et déclaratives afin d'augmenter le niveau concurrentiel des entreprises. Néanmoins, elles nécessitent un effort d'alignement entre la logique métier et la logique du processus. En outre, nous constatons que certaines approches ne sont pas faisables en pratique puisqu'ils n'utilisent pas les standards des processus. De plus, l'efficacité et la flexibilité sont antagonistes. Par conséquent, dans cette thèse, nous nous intéressons à la gestion sensible au métier visant à : (1) concilier les techniques déclaratives et les techniques impératives en une approche hybride pour tirer profit de leurs avantages, (2) préserver les standards des processus, et (3) minimiser l'effort des concepteurs. Nous avons ainsi proposé une nouvelle approche hybride pour la gestion des processus métiers. Nous avons modélisé la gestion d'un processus métier par un processus de gestion connecté au premier qui permet de le superviser et le configurer. Ce processus de gestion est généré grâce à une modélisation sémantique des relations entre les processus métiers, les services et l'environnement métier. Nous avons également implémenté et évalué notre approche en comparaison avec les approches existantes de gestion sensible aux changements de l'environnement métier.

Title : Business-aware management of service-based business processes

Keywords : business processes, business environment, efficiency and flexibility

Abstract: Continuous business environment changes urge companies to adapt their processes from a business environment point of view. Indeed, companies struggle to find a balance between adapting their processes and keeping competitiveness. While the imperative nature of business processes is too rigid to adapt them at run-time, the declarative one of the purely rule based business processes is, however, very time consuming. Hybrid approaches in turn try to reconcile between these approaches aiming to reach the market requirements. Nevertheless, they also need an effort for aligning business logic and process logic. Therefore, in this thesis, we focus on business environment-aware management of service-based business processes aiming at conciliating imperative and declarative approaches. Our challenge is to develop a hybrid management approach that preserves industry standards to describe and to manage SBPs as well as minimizes designers' efforts. Based on a semantic modeling of business environment, business processes as well as their relationships, and a control dependency analysis of business processes, we are able to synthesize a controller, itself modeled as a process, connected to the business process to be monitored and configured at run-time. We also validated the feasibility of our management approach by implementing the framework Business Environment-Aware Management for Service-based Business processes (BEAM4SBP). Experimentations show the efficiency of our approach with respect to other BEAM approaches.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France