Guided configuration of industry reference models

Niculae, C.C.

Award date:
2011

Technische Universiteit
Eindhoven
University of Technology

Eindhoven University of Technology
Department of Mathematics and Computer Science

Guided Configuration of Industry Reference Models

in partial fulfilment of the requirements for the degree of
Master of Science in Business Information Systems

Cosmina Cristina Niculae

Supervisor:	prof.dr.ir. W.M.P. (Wil) van der Aalst (TU/e - W&I - AIS)
Tutors:	dr.ir. J.M.E.M. (Jan Martijn) van der Werf (TU/e - W&I - AIS)
	ing. R. (Rine) le Comte (To-Increase B.V.)
External member:	dr. G.H.L. (George) Fletcher (TU/e - W&I - DH)

Eindhoven, September 2011

Acknowledgements

This thesis is the result of my graduation project, which concludes the master program Business Information Systems within Eindhoven University of Technology. The project was carried out within To-Increase B.V. in collaboration with the Architecture of Information Systems Group of the department of Mathematics and Computer Science.

Writing a thesis is a learning process. This period served me not only to accumulate knowledge in the domain of configurable process models but also to gain new life experiences. During this project I had the opportunity to work with various people and I would like to thank all of them for a fruitful collaboration.

Thank you professor Wil van der Aalst for offering me the opportunity to carry out this project within To-Increase and for the valuable feedback provided.

Thank you Jan Martijn van der Werf for always making time for our meetings, for the confidence and interest you put in this project and for the valuable feedback. Your guidance was of real help and was appreciated.

Working within To-Increase was from the beginning a pleasure for me. Thank you Rine le Comte for all our interesting conversations which made the project progress and for the confidence you had in me. Also, thank you Eric van Hofwegen for sharing with me the knowledge over service flaws and IEM solution and for the nice jocks and good mood you always share with the others. Furthermore, I would like to address thanks to Phil Simpson. Our collaboration was very nice and your feedback on the service process was very useful.

Along this journey I experienced also some difficult periods. The unconditioned support of my parents and Andrei made me go further every time and I want to thank them especially.

Finally, I want to dedicate this thesis to Unchiu' Nicu who was the first showing me that an educated mind can conquer the world. I know you are still watching me from above.

Cosmina Cristina Niculae

Eindhoven,
September 2011

Abstract

Configurable process models are compact representations of *process families*, capturing both the similarities and differences of business processes and further allowing for the *individualization* of such processes in line with particular requirements. Such a representation of business processes can be adopted in the consultancy sector and especially in the ERP market, as ERP systems represent general solutions applicable for a range of industries and need further configuration before being implemented to particular organizations. Configurable process models can potentially bring several benefits when used in practice, such as faster delivery times in project implementations or standardization of business processes. They follow the *design by reuse* paradigm, eliminating the need to redo the business process models from scratch on each project implementation. As configurable processes are a relatively new line of research in the BPM field, they are not widely used yet.

In this thesis *we focus on deriving configurations for particular organizations.* We investigate a new methodology for business process configuration, which allows business consultants and domain experts to conduct the configuration phase. *The methodology decouples the low level variation points captured in the process, which are of a technical nature, from the high level configuration decisions, which are driven from the business side.*

The configuration methodology proposed in this thesis is based on the *business context* in which business processes operate. We structure the context on three layers, the *process layer*, the *organizational layer* and the *external layer*. The process layer captures the business concepts describing business processes (e.g. activities, resources, events, data). The organizational layer includes elements describing the structure and internal activities of organizations. Finally, the external layer describes the external environment that encompass organizations, including the relations that an organization has with other business entities on the market like suppliers, customers, representatives. For modelling context in a manner close to the natural language allowing automatic reasoning at the same time, we use *ontologies*.

We propose to configure a process model following a three step approach.

First, the business context is modelled with the help of ontologies. This results in an ontology model which we call *context model*. The context model is easily extensible. Such a model will be tailored to meet the needs of the consultancy company which engineers it, serving the organization to configure the business processes for the industry sectors in which it operates. Moreover, as a new business line is created for a new industry sector, the context model can be extended to encompass this new knowledge.

Second, for being able to derive configuration decisions based on the business context, a set of *configuration rules* are defined. These rules are able to reason over the business context of an organization and to infer a valuation over *domain facts*. Domain facts abstract from the variation points present in the configurable process model.

Last, the context model is instantiated for the business model of a particular organization,

resulting in an *organization model*. The configuration rules are applied over the knowledge captured in the organization model and configuration decisions over the business process are automatically inferred. An individualized business process results at the end of the process configuration phase.

In order to support the new methodology, we developed tooling which is integrated in the Synergia architecture. This allows for end-to-end support for process configuration, from capturing the configurable model in C-YAWL and until obtaining individualized variants of the process, in line with the particular requirements of stakeholders.

Finally we apply the proposed methodology to the real-life technical service process common within the manufacturing industry and validate our results in collaboration with business consultants. Our initial results show that the technical service process provides sufficient variations to be modeled as a configurable process and can be successfully configured using the novel methodology.

Contents

Chapter 1

Introduction

Business Process Management (BPM) is focusing on the continuous improvement of business processes within organizations in order to achieve business effectiveness and efficiency. A *business process* consists of a collection of related tasks, or activities, that need to be carried out in order to satisfy a particular customer need [3]. BPM considers business processes as corporate assets and aims to provide them with flexibility and to integrate them with technology. Business processes can be found within and across organizations and can be either private (taking place within an enterprise at the strategic, management or operational level) or public (involving external organizations) [30]. Business processes are formalized as *business process models* which are abstract models that visualize the behavior of business processes [18].

Within the BPM field, the domain of *Business Process Configuration* deals with capturing and individualizing multiple variants for the same business process in the form of a *configurable process model* [2, 46, 41].

Even if organizations implement very similarly processes as sales, insurance or technical support, variations exist due to specific requirements [18]. For example, within the manufacturing industry, the sales process can include a quotation phase in some organizations, if the price of the sold goods can vary, while other organizations sale products at a fixed price and do not need to send quotes to their customers. These different variations of the same business process within a particular domain form what we call a *process family*. The *domain* can vary from a particular functional area such as Customer Relationship Management or Financial Accounting to an entire industry sector - e.g. Food Industry [46].

Configurable process models are an integrated representation of a process family - i.e. a single process model combining multiple variations of a business process. Configurable process models are not models which can be directly deployed in organizations. Such integrated process models need first to be configured to meet specific requirements, before being implemented to an organization [2]. The configuration implies the selection of the desired options from the integrated set of process variants captured by the configurable process model. In this way a process variant containing only those parts which are required by a particular organization is obtained, while the undesired behavior is eliminated.

Configurable process models offer benefits like elimination of redundancies in a process family, standardization of processes in a given domain, reuse of proven best practices, etc.

Enterprise Systems (ES) support the execution of business processes in organizations. Vendors of ES solutions, especially of Enterprise Resource Planning (ERP) systems, aim to

capture the business processes that the software supports through the mean of *reference models*. These reference models should comprise all the variations of a process and should allow further the stakeholders to select a particular process individualization. In this way, the modelling efforts of building from scratch process models for each particular customer situation can be reduced [46]. Therefore, configurable process models are suitable for capturing such reference models.

The *process configuration life cycle* (Figure 1.1) illustrates the dynamics of configurable process models. This life cycle is usually implemented within organizations interested to use reference models (ES vendors, consultancy companies).

At *design time*, given the process family, a configurable process model is generated. The configurable process model captures the different variations between the initial family of processes through the mean of so-called *variation points*. During the *configuration time*, an analyst selects from the integrated process model an individual alternative, in line with the particular requirements of the organization willing to implement the process. This phase represents the *process configuration step*. The result of this phase is a so-called *individualized process model*. The decisions taken during configuration time have direct impact over the final structure of the individualized process model [46]. During *build time*, the individualized process model is used to implement an executable model for a specific information system at the customer side. Further, various instances of this model are run within the customer's organization during *run time*. The individualized process model obtained can be again inserted within the initial process family, and the cycle can be redone.

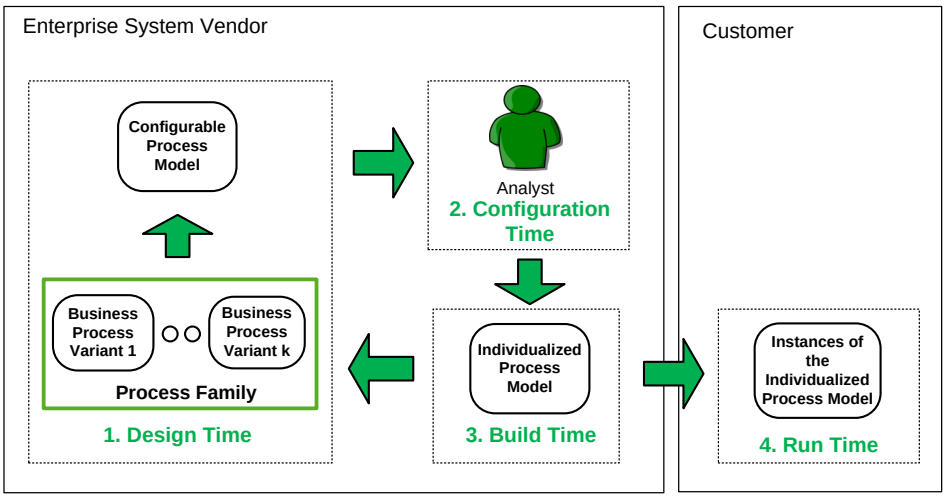

Figure 1.1: Process configuration life cycle

Selecting an individualized process model during the process configuration step is based on information from the domain in which the process will be implemented, which is called *domain variability*. In the same time, understanding the structure of the configurable process model and the mechanisms used to configure the variation points present in it is needed. It is legitimate to assume that the modellers who design configurable process models are familiar with the formalisms used to capture them. In the same time, domain experts who provide input to configure these models (e.g. business consultants, customers) are not familiar with these formalisms [33]. Therefore, finding a method which can guide domain experts in

configuring business processes by abstracting from the low level variations points in the process model is an important step in using configurable process models in industry.

The goal of this master thesis is *to provide a configuration methodology that decouples the domain knowledge from the process knowledge, allowing analysts to perform the process configuration step without having to understand the low level configuration elements present in the configurable process model.*

This chapter gives an overview of the context in which this master project was conducted, together with the research questions that triggered the current work. Section 1.1 describes the context in which the master thesis was carried out. Further, the problem description is provided in Section 1.2. The research objectives, questions and the methodology used to reach the results are presented in Sections 1.3, 1.4 and 1.5, respectively. This chapter concludes with an overview describing the structure of this document and its chapters, in Section 1.6.

1.1 Thesis Context

The research project was conducted within To-Increase B.V. To-Increase is an independent software vendor (ISV), developing and selling Microsoft Dynamics Enterprise Resource Planning (ERP) solutions through a partner network. Within the current research project, an interesting point for To-Increase regards how the process configuration principles can be applied in order to deliver ERP project implementations to customers. Another aspect takes into consideration how the process configuration cycle can be implemented within a consultancy company.

The motivation for this project comes therefore from the necessity to understand how configurable business processes can be applied in the context of ERP systems. More exactly, given a repository of configurable process models, the main topic of interest concerns the step of configuring these processes for a particular customer. By using configurable process models in ERP implementation projects, the "design by reuse" paradigm can be applied and the need to redo process models from scratch for each customer is eliminated [46].

In order to facilitate the use of configurable process models in industry, the goal is to decouple the low level process variations (process knowledge) from the business knowledge, allowing business consultants and domain experts to conduct the process configuration step without needing to understand the low level variations captured in the configurable process model. This constitutes also the motivation of the current research project.

The following subsections introduce more in detail the context in which this master's project has been carried out. First, a general overview of Enterprise Resource Planning systems is provided. Next, we provide an example of a business process, which can be supported by any ERP system, and which will be further used along this thesis, as a running example.

1.1.1 Enterprise Resource Planning Systems

Enterprise Resource Planning (ERP) systems are packaged software solutions aiming to integrate the complete range of business processes running inside an organization in a unique IT infrastructure. Many vendors are offering ERP solutions to the market, some of the most important according to a Gartner Worldwide ERP Report from June 2009 [24] being SAP, Oracle, the Sage Group and Microsoft. According to the same report, the ERP market in 2009 was estimated to $ 24,5 billion.

ERP software represents a standard solution which targets a varied range of industries, therefore when implementing such a solution at a customer, the standard software needs to be configured in order to support the specific requirements of that particular organization. Furthermore, pre-configured solutions for particular industries (e.g. manufacturing industry, construction industry, retail industry etc.) have been released [29]. Therefore, ERP systems form a good candidate for the use of configurable processes.

From a vendor perspective, the business processes incorporated in an ERP solution represent best practices, and are captured as business/industry reference models, i.e. conceptual models aiming to capture best practice processes for a certain domain or class of domains. Some examples of industry reference models include the Supply Chain Operations Reference-model (SCOR), the process reference model endorsed by the Supply Chain Council as a tool for supply chain management [11]. A well-known example of a reference model capturing the functionality of an ERP solution has been documented by the SAP vendor - the SAP R/3 collection of reference models [12]. Nevertheless, the quality of such models is rather problematic. For example, a close analysis of the SAP reference models has revealed that more than 5% of these models are flawed and present additional problems as poor consistency among related models [37, 14].

Representing complex solutions, ERP systems do not only bring advantages, have also an important number of problems that both vendors and consumers have tried to overcome. Some of the most important problems include *high implementation costs* - both in terms of time and money, *need of enterprise reengineering* - meaning that companies need to restructure and redesign their internal business processes in order to implement an ERP solution in the organization [13], *alignment with the specific requirements of an organization* - one of the most frequent cause of failure for an ERP implementation being the lack of aligning between the system and the business needs of the enterprise [13], as well as *need for important allocation of internal resources* [43].

Capturing industry reference models supported by ERP solutions as configurable process models represents an interesting line of research, as such an approach allows for reuse and adaptation of process models. Additional benefits as shortening the delivery time to customers in ERP project implementations and reducing the number of specialists involved in such projects can be potentially achieved by using this approach.

1.1.2 Running Example

In order to better describe the context that generates the research objectives for this project, let us consider a common sales process, as supported by a generic ERP software package. The sales process can present variations, depending on several factors (e.g. the domain in which the process is implemented, the internal procedures that the companies follow, the type of goods that are being sold, etc.). We consider two variations of this process, illustrated in Figure 1.2.

The process starts once a company selling goods receives a sales order from one of its customers. First a cost estimation for that specific good needs to be calculated and a quotation document needs to be sent to the customer. The cost estimation can be influenced by the materials used to manufacture that specific good (items), by the number of working hours necessary to manufacture and deliver the good to the customer, and by other type of expenses that are needed in order for the good to be successfully delivered (e.g. transportation costs, fixed costs, etc.). After the quotation document is sent to the customer, the response of the

customer is inspected. In case the customer agrees with the costs, it might be necessary for the good to be engineered and then the good is delivered to the customer. An additional step which synchronizes the actual costs of the product with the estimated costs introduced in the quotation phase is needed in case differences between these costs exist. The last step in the process consists in sending an invoice to the customer. In case the customer rejects the quotation, then the selling process ends immediately after the negative response in received.

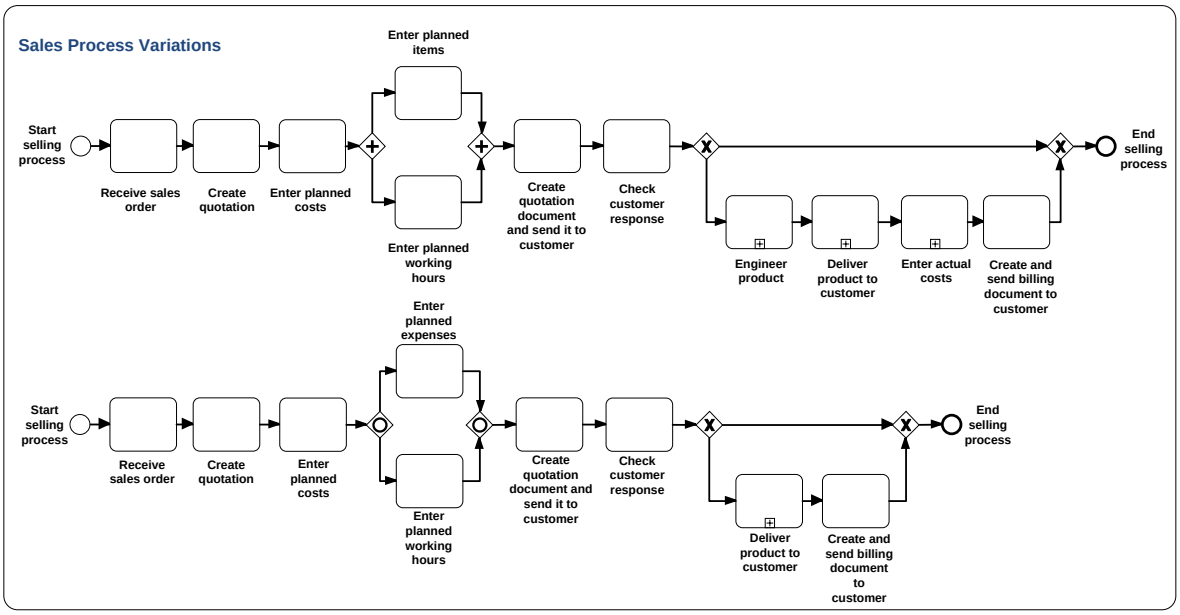

Figure 1.2: Process family for the sales process

Let us now closely inspect the two variations of the sales process illustrated in Figure 1.2.

The first difference regards the cost estimation step. We can see that in the first process (Figure 1.2, top), the planned items and the planned working hours are considered as important and introduced in the system simultaneously, whether in the second process (Figure 1.2, bottom), either the planned expenses, the planned working hours, or both are considered. This implies that a company implementing the second process can generate cost estimations based only on the necessary expenses, or only on the estimated number of working hours, or can consider both.

The next difference among the two process variations involves the product engineering step. In the first process (top), in case the customer accepts the quotation, the product is engineered and delivered. The second process (bottom) does not include the engineering phase, the product being directly delivered to the customer. This might imply that a company implementing the bottom process can deliver products already available in stock, whereas a company implementing the top process needs to engineer a product every time, according to the specifications of the customers.

Lastly, the costs synchronization step is considered in case of the top process, being absent in case of the bottom process.

These two processes represent the “sales process” process family. Considering this process family, the challenge for a consultancy company is to generate a configurable process model

able to capture the variations in both processes in an integrated representation. Variation points are used to represent these variations among the process family. Further, the resulting configurable business process needs to be configured to meet the requirements of particular customers, resulting in an individualized business process.

Next, we will describe the problems that arise in the given context, and the objectives settled for the current project.

1.2 Problem Description

Configurable process models contain a number of configuration alternatives in their structure, number that grows with the complexity of the processes. The process configuration step implies selecting a subset of such alternatives that are suitable for a particular situation and has as output an individualized process model. This step is desirable to be executed by those stakeholders that provide the input for configuring the process (e.g. business consultants, domain experts) and not by the business modellers, whose only role should be in designing and maintaining the collection of configurable business processes [18, 34]. Therefore, there appears the need of a configuration methodology which takes into consideration the business context in which the process aims to be implemented.

This gap between the configuration options in the process and the higher level configuration options of the domain has been partially addressed by La Rosa et al. [34]. In this approach, a higher level of abstraction over the process variation points is introduced by defining a set of domain facts which influence the configuration of several such variation points in the process. Moreover, a set of questions guides the user in providing values for these domain facts. A complete collection of tools grouped under the name Synergia Tool Set [32] is available and supports the entire process configuration cycle, including the process configuration step.

In line with these new research undergone in business process configuration, the management of To-Increase is interested in a business process configuration methodology that allows for the configuration step to be conducted by business consultants in collaboration with customers (domain experts), having as a result an individualized business process model conform with the needs of the customer's organization. Therefore, such an approach ideally abstracts from the variation points of the configurable process model.

When considering the discussion above, the main problem that initiated the current research project is *to provide a business process configuration methodology that is able to abstract from the low level, technical variations points captured in the configurable process model and to allow the configuration step to be driven by a particular business context.*

1.3 Research Objective

Taking into consideration the identified problem, the focus of the project is to provide a new methodology for conducting the process configuration step, starting from the business context of the organizations implementing the processes. This approach is based on research that has been recently conducted into business process configuration [2]. In providing such a method, our aim is to define a knowledge model incorporating the business elements which determine what configuration decisions to be taken over business processes. Further, we aim to model

this knowledge model, which is defined at a theoretical level, with the help of ontologies, bringing ontologies in this new area of research.

Having the knowledge model defined, and integrated in the new process configuration methodology, the next step is to provide a tool support for the new approach. A new tool able to derive configuration decisions over a business process based on the business context of an organization needs to be developed. The tool will be integrated in Synergia Tool Set.

The proposed methodology is validated on a real-life configurable business process supported by the Industrial Equipment Manufacturing (IEM) information system, an industry solution developed by To-Increase B.V. on top of the Microsoft Dynamics AX ERP solution.

Having all these considered, the objective of the project can be summarized as follows:

Develop a new methodology and toolset for the configuration of process models, taking into account the business context that determines the configuration decisions over processes. The new tool should be integrated within the already existing collection of systems implementing the process configuration life cycle, such that end-to-end configuration support over the life cycle is provided.

1.4 Research Questions

The objectives described in Section 1.3 raise a set of research questions that need to be answered during this master project in order to overcome the identified problems. Specifically, we address the following questions:

1. **Which are the desirable characteristics of a relevant knowledge model for the configuration of business processes?** Defining the characteristics that the knowledge model needs to meet, in order to have a role in configuring business processes, is the main issue addressed by this question.

2. **What elements need to be considered in engineering the knowledge model?** A universally applicable and comprehensive knowledge model is a difficult task to attain. Therefore, a set of guiding principles, aiming to help interested parties in engineering such a model, need to be considered.

3. **What mapping exists between the knowledge model and the variations points of a configurable business process?** The next point of interest is to determine how does the information captured in the knowledge model help in setting the configuration decisions for a particular business process. Do any additional constructs need to be defined in this purpose?

4. **How can the knowledge model be used during the process configuration step?** The role played by the knowledge model in process configuration, and the exact steps that need to be performed in order to configure a process model for a particular situation, are the two issues to be addressed by this question.

5. **Is the novel configuration methodology feasible in practice?** Testing whether the new business process configuration methodology can be used in practice, in the context of ERP systems, is an important step in the research.

1.5 Research Methodology

In order to address the research questions, the following steps are taken:

1. Conduct a literature review and a case study over existing process configuration techniques.
2. Identify the limitations of the current techniques, and determine the requirements for a new method for business process configuration.
3. Define an appropriate knowledge model, knowledge representation and additional constructs needed for the new method.
4. Conduct a literature review over knowledge models and derive a set of requirements for the engineering of knowledge models used for process configuration.
5. Implement a new tool that supports the approach.
6. Validate the novel methodology on a real-life configurable business process supported by the IEM solution.

1.6 Main Contributions and Outline

In this thesis we study the configuration of process models, by considering the business context in which an organization is operating, with the goal of decoupling the domain knowledge from the process knowledge in the configuration phase. In this way the configuration phase can be managed by domain experts which have no in depth knowledge over the technical representation of configurable process models. We propose to capture the business knowledge with the help of ontologies, introducing a new application area for ontologies, which are currently mainly used as a communication mean between various stakeholders in a domain.

The remainder of this thesis is structured as follows:

Chapter 2 provides an overview over the necessary background information used throughout the thesis. For this the theory behind configurable process models, the BPMN modeling language, as well as ontologies are introduced.

In Chapter 3 we present the process configuration framework, detailing the process configuration life cycle, the process configuration methodology and the supporting software architecture.

A novel methodology for process configuration is detailed in Chapter 4. Our main contributions are the use of context models in process configuration, the use of ontologies for modelling this context knowledge and the definition of a new configuration methodology based on context models.

Chapter 5 presents the supporting software architecture for the newly introduced methodology. Our main contributions here are the development of a new tool able to reason over a knowledge base in deriving configuration decisions over a business process and its integration within the available Synergia toolset for providing end-to-end configuration support.

The developed methodology is validated for the configuration of the technical service process, a business process supported by the IEM industry solution, in the case study presented in Chapter 6.

This thesis concludes with Chapter 7 which presents the work that has been done, conclusions and future work.

Chapter 2

Preliminaries

In this chapter, we introduce the background information used throughout this thesis.

First we introduce the concepts of business process modelling and we illustrate them on the Business Process Modeling Notation (BPMN) modelling language.

Next, we show how conventional business process modelling languages can be extended to support configurability. We have opted for (C-)YAWL as a modelling language for representing configurable process models, due to its support for configurability, its academic foundation, and its integration in the Synergia Tool Set. Therefore, we first present, in Section 2.2, the theory behind configurable processes. We start by illustrating the hiding and blocking principles on label transition systems. Then we show how these principles are applied to the C-YAWL configurable modelling language.

A mapping between elements in BPMN and elements in YAWL is presented in Section 2.3.

We also use ontologies in the current project in relation to the novel process configuration methodology. Thus, in Section 2.4, we describe the main elements and theory behind ontology models.

2.1 Business Process Modeling Notation

A *business process model* is a set of *activities* with *casual dependencies* which capture the behavior of business processes. In order to enable practitioners to model and visualize the process flow of business processes, several business process modelling languages were developed, which come with tool support. Most tools implementing business process modelling languages also allow for the annotation of the process model with additional information, such as resources or data, and for basic methods to analyse the correctness of the models [18]. We will discuss in this section the Business Process Modeling Notation (BPMN), which is a standard notation for business process modelling (has currently reached version 2.0) and is supported by a variety of tools.

The order in which activities can be executed is determined in BPMN notation by so called *gateways*. We distinguish the following basic gateways: AND, XOR and OR, having both splitting and joining behavior.

To mark the triggering of an activity, *events* are used. We distinguish two special events: a *start event*, marking the start of the execution of a process instance, and an *end event*, marking the end of the execution of a process instance.

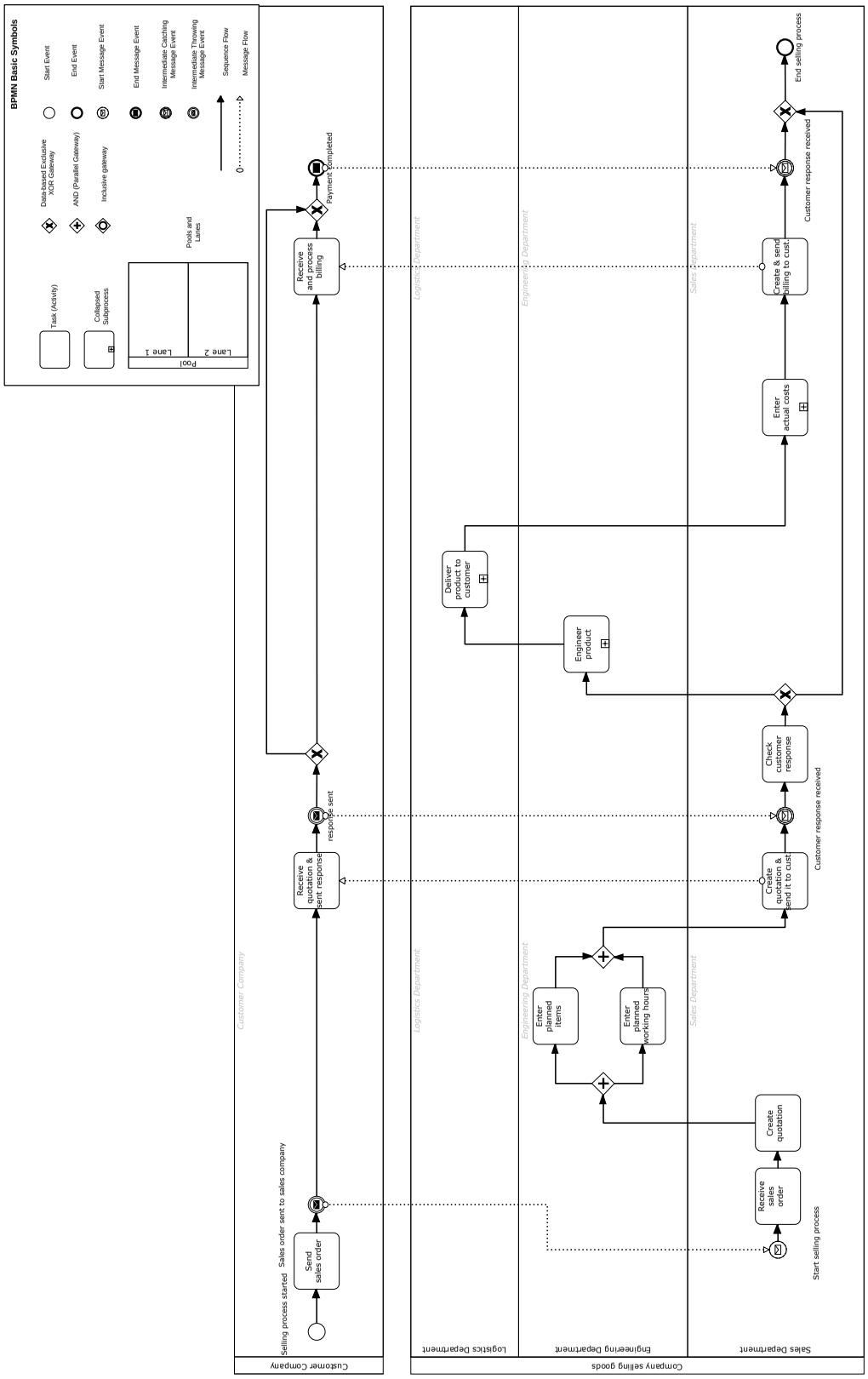

Figure 2.1: Sales process in BPMN

A BPMN model has at least one start event and at least one end event. Intermediate events are also part of the BPMN specification and are used to trigger activities or to signal the result of an executed activity; another line of classifying events is according to their trigger, BPMN 2.0 supporting a varied collection of triggers. We will describe in this section only to the message, timer, error and condition event, which are the most common types of events. The message event is triggered by the receipt or sending of a message. A condition event has a rule attached to it and is triggered when this rule becomes true. A timer event delays the process execution until a certain point in time is reached or a particular duration has elapsed. Finally, the error event is triggered when an error occurs.

In terms of activities, BPMN supports both atomic and composite activities - subprocesses.

By using lanes and pools, the process can be split among various parties which can be organizations, different roles or systems. Lanes subdivide pools or other lanes hierarchically. Each party involved in the process has its own process and the exchange of messages between the various participants synchronizes these processes.

Let us consider again one of the variations of the running sales process example defined in BPMN notation (Figure 2.1). We can see that the process is split between the two interested actors, the selling company and the buying company. Moreover, within the selling organization we can distinguish several organizational units (departments), namely the selling, engineering and logistics department. The exchange of messages between these two organizations synchronizes the two parallel processes. The buying company executes a minimal process of sending a sales order for a product, reviewing the quotation document and sending a response for it and finally making the payment arrangements in case it decides to buy the product. The business process within the selling organization follows the same flow as presented in Chapter 1.

For more information concerning the BPMN semantics, please refer to [23].

2.2 Configurable Process Models

As introduced in Section 1.1.1, industry reference models represent a proper solution for describing at a conceptual level the business processes supported by ERP software, helping in saving both time and money for Enterprise Systems (ES) (e.g. ERP Systems) vendors or consultancy companies implementing ERP projects to customers. Some of the benefits brought by capturing business processes as industry reference models include faster implementation of ERP projects to customers, standardization and reuse of the business processes supported by ES, less resources involved in the implementation process, personnel training or communication improvement between business process stakeholders [42]. Such industry reference models, that have a clear focus in capturing the functionality of ES solutions, are categorized in [41] as application reference models. Moreover, these models, which describe a general process, usually need to be adapted to meet the requirements of a specific organization. Consequently, it is important to derive from such industry reference models an individualized solution, in accordance with the requirements of a particular customer. This raises the need for a modelling language able to capture the different variations of a business process in a compact reference model and to further select an individual process variant from such a model. Furthermore, the application reference models have as a main purpose to capture the various business processes supported by enterprise systems; nevertheless, some of the reference models available today present errors which decouple them from the information system they intend to capture [14].

In line with the requirements mentioned above, Rosemann and van der Aalst introduced the notion of configurable reference/process models [46]. We will be using in the remainder of this thesis the notion of *configurable process models*.

A configurable process model captures in a compact representation both the similarities and differences of a business process, by merging several variants of the same process into one model. The configurable model captures all possible behavior, while at configuration time only certain paths in the process are selected and the other unnecessary paths are eliminated from the model. This approach is known in the specialized literature as the restriction of behavior principle. In this way a configurable process allows for reuse of best practices and answers in the same time to the specific requirements of organizations.

2.2.1 Labeled Transition Systems

In order to theoretically define the process configuration principles, by abstracting from any particular modelling language, van der Aalst et al. used the *hiding* and *blocking* operators on *Labeled Transition Systems* (LTSs) [2].

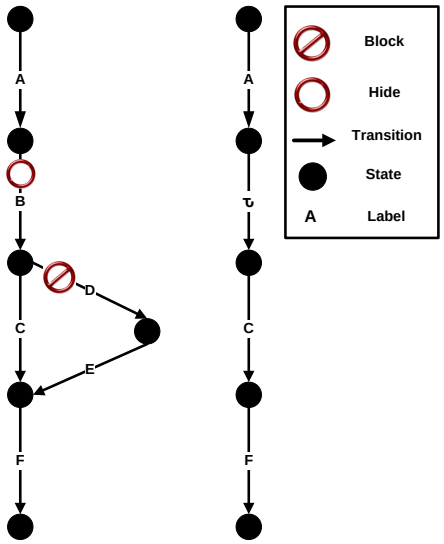

Figure 2.2: Configurable LTS

Any process model with a formal semantic can be mapped into a LTS. LTSs capture the behavior of a business process by mapping it to a directed graph where nodes represent states while arrows represent transitions from one state to another. A label is attached to each transition and represents an event, action or activity causing the state change. Another element appearing in a LTS is the silent transition, labeled with τ. A silent transition is not visible in the sense that moving from one state to the other is not triggered by the execution of an activity. A LTS can be configured by making use of the hiding and blocking mechanism.

Hiding and blocking operators are applied in a LTS to transitions. By blocking a transition in a LTS, the corresponding path cannot be followed anymore and needs to be removed. This also implies that the subsequent transitions are not executed if they cannot be reached by another path in the transition system. By hiding a transition in a LTS, the transition is

replaced with a silent one, meaning that the path in the LTS is still possible but the transition remains hidden to the exterior (it is skipped). If a transition is neither blocked, nor hidden, then it is allowed and the LTS remains unchanged. In the case of configurable LTSs, blocking and hiding a transition become configuration decisions.

When configuring a model (at configuration time), there might exist configurations that are not made due to restrictions imposed by the business domain in which the process is executed or because such a configuration would generate a deadlock situation. Therefore, only a subset of the total number of possible configurations is considered with a configurable process model.

Figure 2.2 illustrates a LTS on which two configuration decisions have been applied, namely to hide the transition labeled B, and respectively to block the transition labeled D. In the resulting LTS, transition B is replaced with a silent transition while transitions D and E are removed from the model as these state changes are no longer possible.

2.2.2 Configurable YAWL

The hiding and blocking principle described above can be applied to the academic workflow language YAWL (a shorthand for "Yet Another Workflow Language") in order to extend it with a configuration dimension [4, 25].

Before presenting the configuration aspects of C-YAWL, we will first give a brief introduction into YAWL, with a clear focus on the elements needed in order to support the configurable dimension. For a more in depth presentation of YAWL workflow language, we refer the reader to [25].

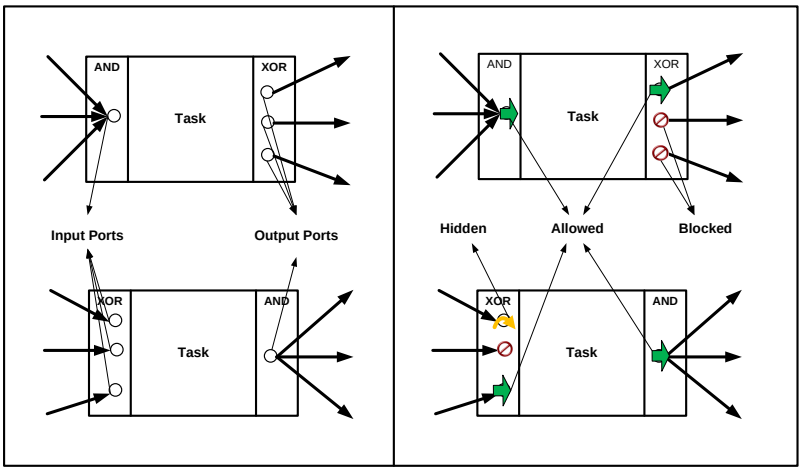

Figure 2.3: The number of ports associated with a task depend on the joining and splitting behavior of the task. A task can be triggered through its input ports, while the flow of the case through the process is determined by the output ports. Input ports can be configured as allowed, hidden or blocked, while output ports can be configured as allowed or blocked.

YAWL is a workflow modeling language based on the Petri net principles [25]. A YAWL specification consists of conditions (places in Petri net representation) and tasks. The control flow determines the flow of tokens through *tasks* and *conditions*. A YAWL model contains

exactly one starting condition and one ending condition, named *input* and *output conditions*. There exist also a variety of task types, each providing a different behavior to the control flow of the process. The most important task types, which are also extended with configuration behavior, are the *AND*, *OR*, and *XOR* tasks (considering both their *split* and *join* behaviors).

The blocking and hiding mechanisms introduced in LTSs are applied on input and output ports in C-YAWL specifications [19]. In LTSs the blocking and hiding operators are applied to transitions which represent state changes in the process. In (C-)YAWL state changes are represented by the execution of tasks which determines a particular flow of a case in the process. The conditions that determine when a task is enabled and can be executed are determined by the input paths entering the task and the different join patterns that define the task behavior, i.e. AND-join, XOR-join or OR-join. For example, while in case of an AND-join behavior, the task is enabled when all its incoming paths are active (all pre-conditions contain at least one token), in case of an XOR-join behavior, the task becomes enabled as soon as one of its incoming paths becomes active (one of its pre-conditions contains a token). In order to make a distinction between these various ways of enabling a task, each possible combination of incoming paths through which a task can be triggered is called an *input port*. Similarly, the possible combinations of output paths through which a case can leave the task are called *output ports*. Having all these notions, we see that a state change of a task is represented by a combination of one input port and one output port which together determine the flow of a case through the process once the task is executed [19] (see Figure 2.3). Consequently, tasks in a C-YAWL specification have associated input and output ports.

Ports represent the configurable elements in C-YAWL and therefore are associated with each configurable task in a C-YAWL specification. Input ports can be configured as either *allowed*, *blocked* or *hidden*, while output ports can be configured as either *allowed* or *blocked* (see Figure 2.3). In case of the input and output ports, allowing, blocking and hiding mechanisms have the same meaning as in the case of LTSs introduced earlier in this chapter.

Another important aspect concerning input and output ports in C-YAWL, that needs to be specified, refers to the number of such ports associated with each type of task. In case of a task with an AND-join behavior, one input port is associated with it, as such a task needs that all its pre-conditions to be enabled in order for the task to become enabled. The same applies when considering the number of output ports associated with an AND-split task. A task with an XOR-join behavior might be enabled by any of its input conditions which contains a token, therefore with such a task one input port for each of its input paths needs to be defined. The same reasoning applies for the XOR-split task and therefore such a task has associated with it a number of output ports equal with the number of output paths leaving this task. A task with an OR-join behavior synchronizes all its incoming branches similar with an AND-join and therefore has only one input port associated. The case of an OR-split task is different, as such a task can release tokens to any combination of outgoing branches. As any of these possible combinations represents a state change, an output port needs to be defined for each such combination. Finally, the simplest case is the one of a configurable atomic task, which has associated one input and one output port. For more in-depth knowledge into configurable workflow models and configurable YAWL we refer the reader to [19].

In order to better understand the C-YAWL principles les us consider the configurable process in Figure 2.4. This C-YAWL process captures the variations in the sales process family, which we have introduced in the previous chapter. We will explain now how the differences between the two processes are modelled with configurable elements in C-YAWL. We can see that the task "*enter planned costs*" is a configurable task, meaning that variations

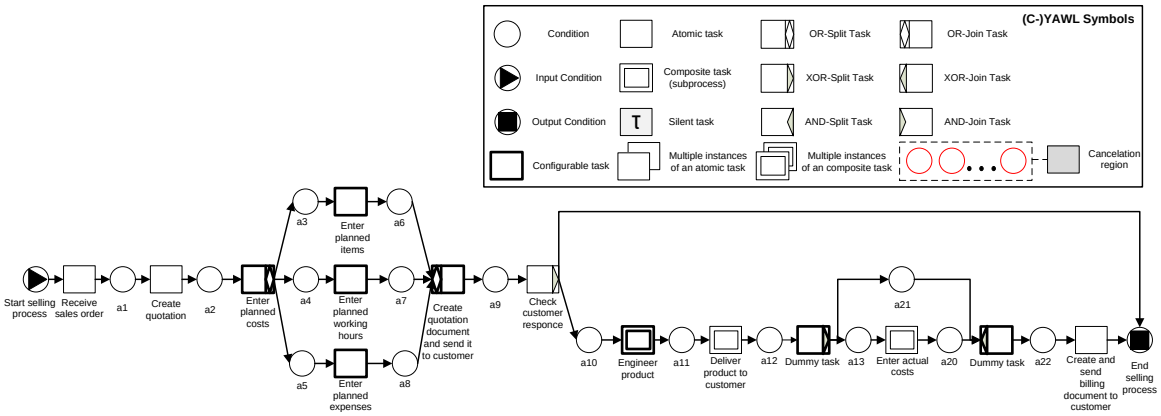

Figure 2.4: C-YAWL configurable sales process

can appear when configuring this sales process for different companies. Indeed, the two sales processes, presented in Section 1.1.2, showed variations when considering the type of costs needed for selling the product. Further, the task "*engineer product*" is a configurable one, as in one of the processes the product was engineered, whether in the other this was not happening. This is also the case of the task "*enter actual costs*".

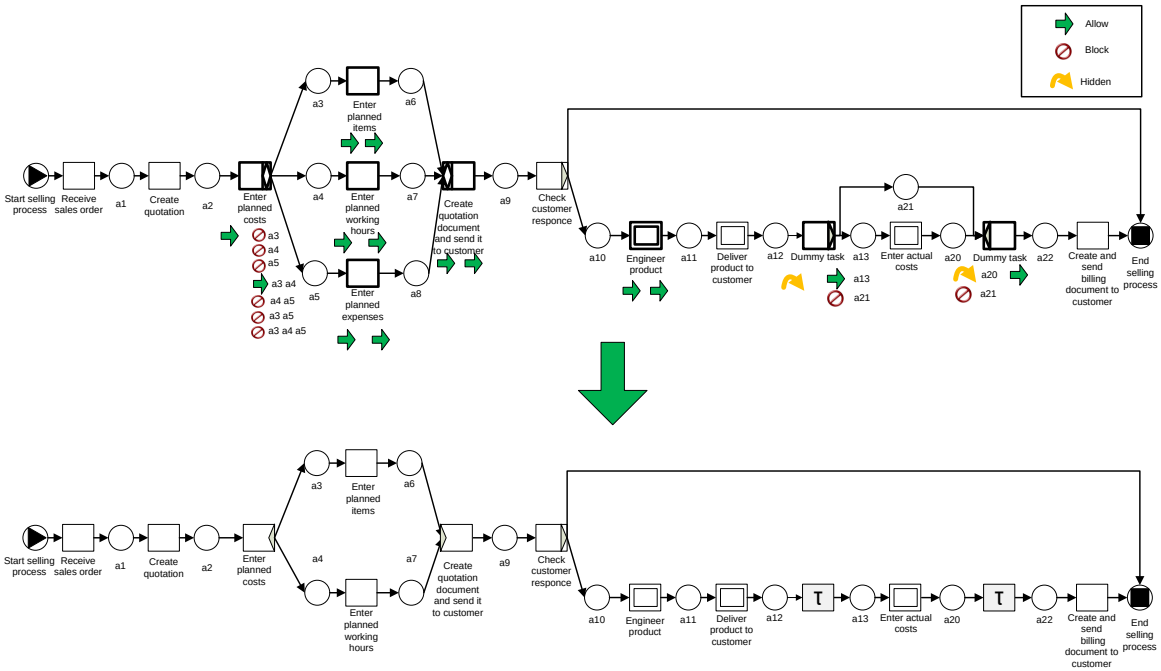

Figure 2.5: Configured sales process for Company A

Let us now consider two companies wishing to implementing this sales process in their ERP softwares. Company A is governed by the Engineer-to-Order (ETO) fulfillment strategy, in which the product is designed and built according to the specifications of the customer, each product possibly having therefore unique characteristics. Therefore, Company A engi-

neers a product, every time a sales order is received. Moreover, Company A includes in its quotation calculation only the items and working hours used as these two factors are relevant in the cost calculation. As products are built differently for every sales order received, the estimated costs which are used for the quotation phase may differ from the actual costs needed. Therefore the quotation document is informative, the actual expenses being finally billed to the customer. The choices made in order to configure the sales process according to the customer specifications and the resulting individualized process can be seen in Figure 2.5.

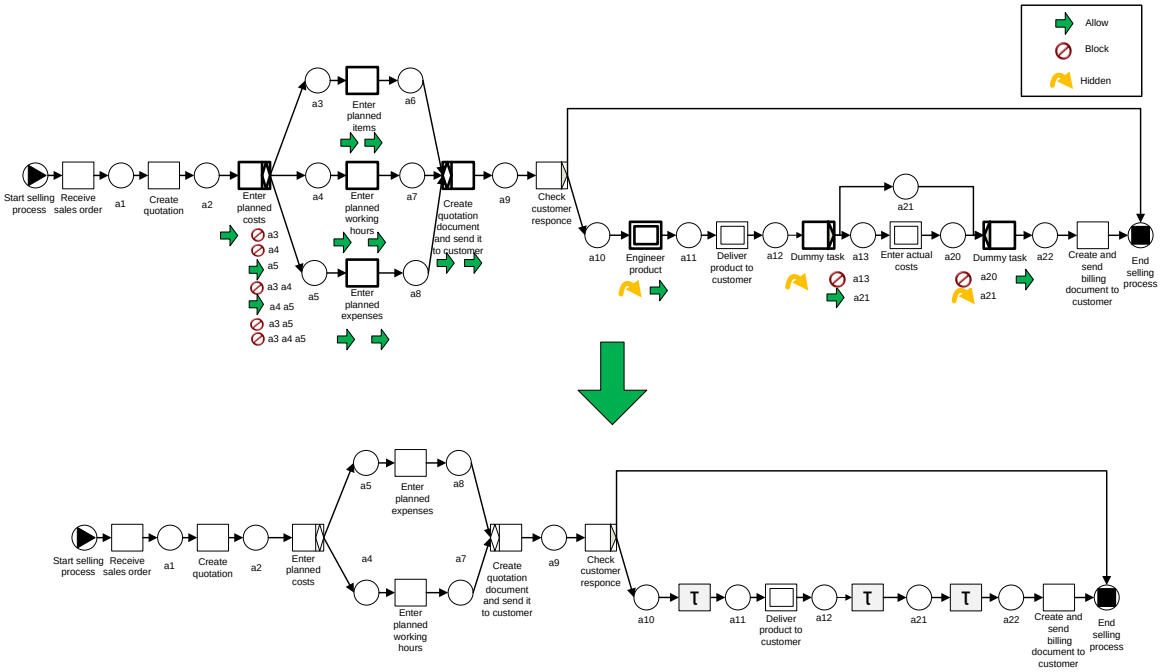

Figure 2.6: Configured sales process for Company B

In order to include only the items and working hours in the costs calculation, the output port corresponding to the outgoing paths *Enter planned items* and *Enter planned working hours* is allowed, and all the other output ports are blocked. Moreover, both the input and output ports corresponding to the task *Engineer product* are allowed, as this task needs to be always present in the sales process of Company A. As the actual costs corresponding to building the products need to be registered in the system, the XOR-split and XOR-join dummy tasks are configured to allow the execution of the *Enter actual costs* task in the system, blocking the skipping path. Further, the input ports of the two dummy tasks are configured as hidden, transforming these two tasks into silent tasks as they do not represent actual/meaningful activities in the selling process. The resulting individualized process corresponds to the specific sales process of Company A.

Company B runs the Make-to-Stock (MTS) fulfillment strategy, in which the product is built according to sales estimations and sold to the customer from the finished goods stock. Therefore, Company B is selling a specific range of products available in stock. The cost calculation in the case of Company B is influenced by the working hours or by expenses (which include in this case the fixed costs of the product) or by both of them. Nevertheless, the expenses are always used when making the cost calculation. As the product is delivered

from stock, no engineering phase is necessary. Further, as the range of products sold by Company B is limited and fixed, the estimated costs are also equal with the actual costs of the goods. Therefore, the quotation document is binding, meaning that once the product is delivered, the customer is billed the price agreed in the quotation document.

The input and output ports corresponding to the configurable tasks of the C-YAWL sales process are configured accordingly with the specifications mentioned above and the individualized process (Figure 2.6) represents the sales process run by Company B.

2.3 Mapping BPMN to YAWL

This section shows how the control-flow in BPMN is mapped to YAWL elements. BPMN and YAWL share common concepts, as the concept of *task* or the concept of *flow* which have a one to one match from BPMN to YAWL. Similarly, BPMN gateways are also mapped to tasks having XOR, OR or AND splitting and joining behaviors in YAWL [25].

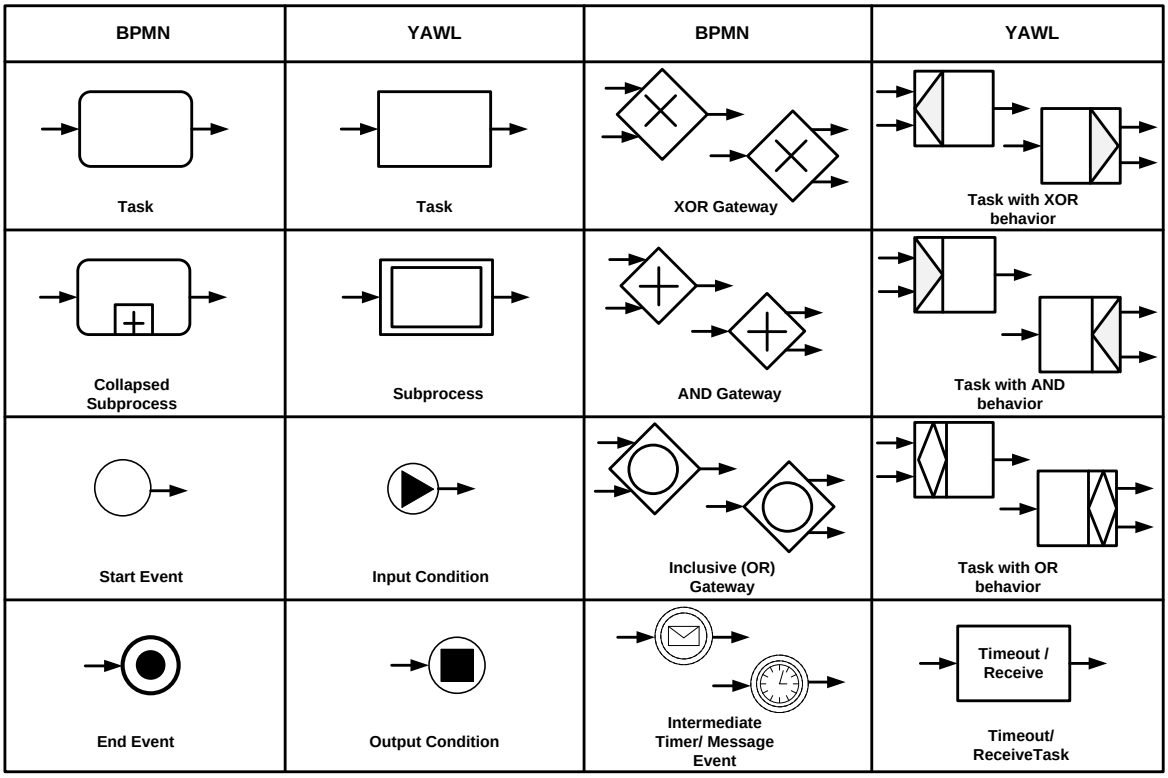

Figure 2.7: Control flow mapping from BPMN to YAWL

When translating a BPMN process model into a YAWL process model, usually a one-to-one correspondence exists. It might also be the case that empty tasks, having only routing purposes, are introduced in the YAWL model to map some of the gateways in the BPMN model. This is because there is not possible to attach this routing behavior to non-empty tasks in the YAWL model. This situation can be observed in the sales process, where in the BPMN model (Figure 2.1), before and after the task *Enter actual costs*, XOR split and join

gateways are used, whereas in the same process modelled in YAWL (Figure 2.4), the two gateways are mapped to two *dummy tasks* having XOR-split and XOR-join behaviors.

The mapping of control flow between BPMN and YAWL elements is summarized in Figure 2.7.

2.4 Modelling Knowledge with Ontologies

In this section we detail ontologies, as a knowledge representation formalism. First, we describe ontologies in terms of metamodel, model and instance level. Next, we describe how reasoning tasks can be performed on top of the knowledge captured in the ontology model.

2.4.1 Ontologies

Ontologies are knowledge models for a domain of interest and describe the elements within the domain and the relationships that hold between these elements. The knowledge captured with the help of ontologies can be easily extended and is represented in a manner close to the human understandable natural language.

For modeling knowledge with the help of ontologies, we have opted for the *OWL 2 Web Ontology Language* (OWL2) [8]. Let us briefly discuss the concepts used for modelling ontologies in OWL language. We will discuss ontologies considering the three levels that describe them, namely the meta-model, the model and the instance level.

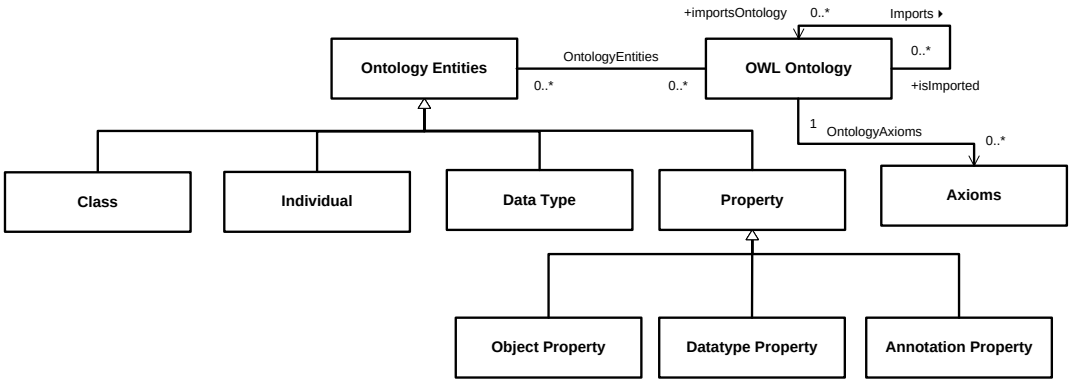

Figure 2.8: Ontology metamodel

For describing the terminology and the theoretical semantics of OWL language, we illustrate the *metamodel layer*, which we designed based on the "Ontology Definition Metamodel" technical report [22] (Figure 2.8). The metamodel depicts therefore the structure of OWL ontologies. An OWL ontology mainly consists of a set of *axioms*, which are statements describing what is true in a given domain (Figure 2.8). OWL 2 has an extensive set of axioms, which provide information about the *entities* in the ontology. For the complete list of axioms that can be defined in OWL ontologies, we refer the user to [9].

Entities are the fundamental building blocks of OWL 2 ontologies, and they define the vocabulary of an ontology. The ontology entities are *classes*, *individuals*, *properties* and *data types*. *Classes* can be defined as sets of elements in a domain and can be organized into

taxonomies, by using the *subclass* axiom. Classes can contain class members which are called *individuals*. The class membership of individuals in an ontology is defined by using the *class assertion* axiom. For using data in OWL ontologies, a predefined set of *datatype* classes exist (e.g., xsd:integer, xsd:literal). For describing relationships, *properties* are used. There are three types of properties in OWL, *object properties*, *annotation properties* and *datatype properties*. *Object properties* are used to describe binary relations between individuals. These properties can further be enriched using *property characteristics*; object properties can be defined as *functional*, *inverse functional*, *symmetric*, *transitive*, etc. For example, if an object property is functional, then a given individual can be linked through this property with at most one other individual. Consequently, setting a property as functional reduces the cardinality of the relationship to at most one. For more in-depth knowledge into property characteristics, we further refer the user to [8, 26]. *Datatype properties* describe relationships between an individual and data values. Finally, *annotation properties* can be used to add information to classes, individuals and object/datatype properties.

Class expression axioms, individual axioms, datatype property axioms, object property axioms and annotation axioms are used to define a particular domain in terms of the above entities [9].

Apart from axioms, an ontology can be also linked to zero or more other ontologies that it *imports* (see the *Imports* association - Figure 2.8). In this way, the extensibility of knowledge is achieved. As such, when creating a new ontology that imports already existent ontologies, new knowledge is added on top of the data already available in the imported ontologies.

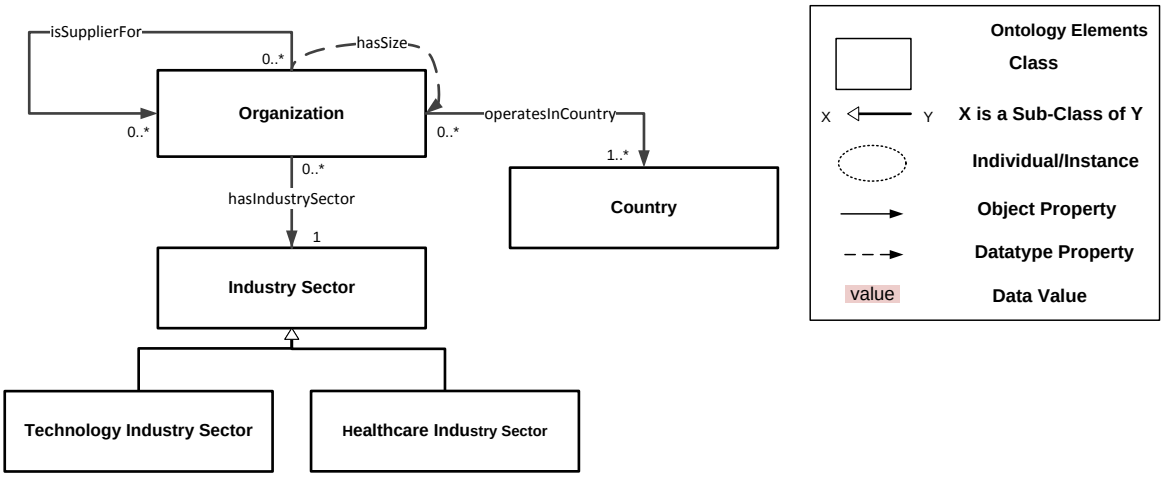

Figure 2.9: An ontology model

By using the semantics of OWL ontologies described above, any domain of interest can be formally defined in terms of classes and relations, resulting in an *ontology model* or *domain model* (Figure 2.9). To give a concrete example, we have defined a business ontology model containing three main classes, the class `Organization`, the class `Country` and the class `Industry Sector`. Further, the `Industry Sector` class is specialized into two classes, namely `Technology Industry Sector` and `Healthcare Industry Sector`, forming a taxonomic structure. For modelling the relations between these classes, several object properties are used. For example, the individuals in the class `Organization` can be related to the individ-

uals in the class `Country` through the object property `operatesInCountry`. Similarly, organizations can be associated with a particular industry sector through the `hasIndustrySector` property. Note that we have defined the object property `hasIndustrySector` as being *functional*, restricting the individuals of the class `Organization` to be related to at most one individual of the class `Industry Sector`. The other two object properties defined have not been marked as *functional*, therefore many to many cardinalities for these relationships are possible.

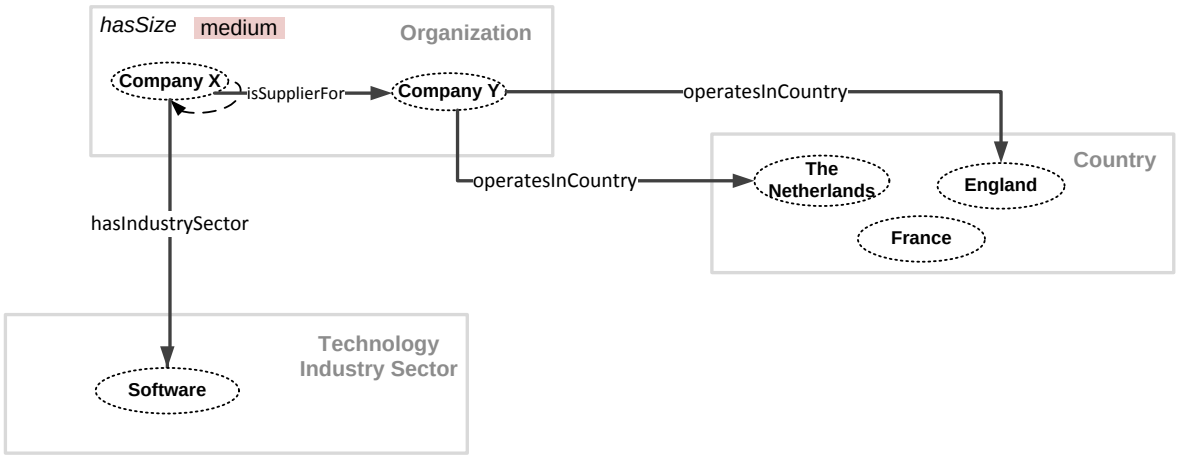

Figure 2.10: An Instantiation of the ontology model; In order to differentiate the instance model from the ontology model, we have depicted with gray the elements defined in the ontology model and with black the elements defined in the instance model

An ontology model describes therefore a domain by organizing it into classes. Next, for characterizing a domain in terms of specific individuals, an instantiation of the domain model needs to be done, resulting in an *instance of the ontology model* (Figure 2.10).

For example, we have created an instantiation of the business ontology model discussed above, by defining two instances for the class `Organization`, namely `Company X` and `Company Y`. Similarly, class `Country` contains three individuals: `The Netherlands`, `England`, and `France`. We completely characterize these individuals by linking them through object properties, or relating them to data values by using datatype properties. For example, `Company X` is linked to `Company Y` through the object property `isSupplierFor`. The size of the individual `Company X` is defined by the datatype property `hasSize`. The range of this property is restricted to the following collection of xsd:strings {“small”, “medium”, “big”}. As `Company X` is a *medium* sized organization, it has been linked to the data value *medium*.

As ontologies are open constructs, there is no clear separation between an ontology model and its instantiation. Therefore, maintaining a clear separation between these two layers is a design decisions of the ontology modeller.

2.4.2 Reasoning Support in Ontologies

One of the advantages of using ontologies for representing domains is the possibility to further process the encompassed knowledge by reasoning over it. Through reasoning, implicit con-

sequences can be derived from explicitly represented knowledge, which forms the knowledge base. Reasoning is performed mainly at the instance level of an ontology.

OWL ontologies allow for reasoning support by using *Semantic Web Rule Language* (SWRL). SWRL rules allow reasoning over the concepts of an ontology by using the Horn principle [27]. SWRL rules, as well as Horn rules, have the format $consequent \Longleftarrow antecedent$, both consequent and antecedent being conjunctions of atoms. Atoms are in this context concepts of the ontology, i.e. classes, individuals, properties, or data types. Moreover, SWRL also allows for build-in functions and predicates (e.g. Built-Ins for Comparisons, Math Built-Ins, Built-Ins for Strings) which enrich the expressiveness of the rules.

An example of a SWRL rule over the knowledge base present in the instantiation of the business ontology model (Figure 2.10) can state for example that, given an individual x of the class Organization which is supplier for another individual, y, of the same class, then the latter individual is a big sized organization. Formally, we can represent this rule, using a human readable syntax of SWRL, as follows:

$$\mathbf{hasSize(?y, big)} \Longleftarrow \mathit{Organization}(?x) \wedge \mathit{isSupplierFor}(?x, ?y) \wedge \mathit{Organization}(?y)$$

For the RDF/XML concrete syntax of SWRL rules, we refer the user to [10].

SWRL, as well as OWL, adopt the open world assumption paradigm, which limits the kinds of inference and deductions that can be made to those that follow from statements that are known to be true. Also, SWRL supports monotonic inference only, which implies that SWRL rules cannot be used to modify existing information in an ontology. As a consequence of SWRL monotonicity, negation as failure is also not supported in rules.

For further information over OWL and SWRL we refer the user to [10, 8, 9, 26].

2.5 Summary

After having provided an overview over process models, we offered a brief introduction in the BPMN modelling language. Next, we defined the blocking and hiding operators on LTS and we have illustrated the principles of configurable process models using the C-YAWL modelling language. After having introduced both BPMN and YAWL modelling languages, the mapping between the control flow of the two languages was discussed. In the last part of this chapter, we have presented OWL ontologies, illustrating the three layers which completely describe ontology models, namely the metamodel, the model and the instance levels.

Throughout this thesis, we will be using all the notions introduced in this chapter for describing the need of a new process configuration methodology, as well as for presenting the novel methodology we propose.

Chapter 3

Process Configuration Framework

In this chapter we start by describing the business process configuration framework. Next, the process configuration life cycle, mentioned in Chapter 1, is elaborated. In order to support the process configuration life cycle, more specifically the process configuration stage, we analyse the questionnaire-based methodology proposed in [33], in Section 3.3. The software architecture which supports the process configuration life-cycle is described in the end of this chapter.

3.1 Process Configuration Framework

The benefits of using configurable process models has drawn the attention of both academia and industry [46]. In order for consultancy companies to make use of the configurability techniques in a consistent way, a process configuration framework needs to be implemented. Such a framework could support the active use of configurable process models in industry.

Let us first consider the requirements that the process configuration framework should meet in order to allow for its consistent use in industry. First, *manageability* for a considerable collection of configurable business processes is a desirable property, as typically a company wishing to use configurable business processes in consultancy projects would be interested in maintaining a repository of such process models. Secondly, *maintainability* is an important aspect; therefore the loop in the framework allows for a constant update of the models in case changes occur. The use of tools to automate the various steps in the process is a must, as making changes to the repository of models manually is infeasible. Another requirement is for the framework to ensure the *correctness* of the generated process models. Such a requirement can be met by using a reliable collection of tools that support the framework and guarantee the correctitude of implemented algorithms. Next, *the ease of use and understanding* of configuration principles is important in order to allow for business consultants and domain experts to undertake the process configuration phase. Therefore, a process configuration methodology which guides consultants in making configuration decisions over the process needs to be provided.

In line with the above requirements, we consider the following dimensions in the process configuration framework:

Process Configuration Life Cycle The dynamics of configurable process models are captured in the process configuration life cycle. This cycle details the main steps through

which configurable business processes pass in their evolution to mature artifacts of an organization. A good understanding over the dynamics of configurable process models can help as a training tool for organizations willing to actively use configurable business processes, showing the main steps that need to be implemented. In the same time, the process configuration cycle ensures that the *maintainability* requirement is met, as the changes occurring to configurable business processes are addressed in the cycle.

Process Configuration Methodology The success of using configurable process models in practice is also influenced by the guidance offered to non-technical users (e.g. business consultants, domain experts) in actually configuring the processes for certain needs. Therefore, the development of a process configuration methodology that can allow business consultants to make configuration decisions is necessary. A proper methodology can therefore answer to the *ease of use and understanding* requirement.

Process Configuration Architecture Proper tool support along all the steps of the process configuration life cycle is very important for enabling the use of configurable process models. Requirements as *manageability* or *correctness* can only be achieved by having a proper tool support.

Having defined the process configuration framework, we will detail in the next sections of this chapter its main components.

3.2 Process Configuration Life Cycle

The life cycle in Figure 3.1 gives an overview of the main phases of the life cycle of configurable models, the main actors that take part in each particular step, the appropriate tools that can support these steps and the outputs in term of process models for each of the steps. We have identified four main steps that ensure a correct use of configurability techniques, which we will briefly describe.

1. **Data Collection Phase**

   A company willing to make use of configurable process models, should consider the creation or purchase of a *repository of business processes*, containing the different processes families. It is desirable that the business processes used in various projects are documented using a widely accepted modelling language (and an appropriate modelling tool). Normally, the main actor for this step is the business process modeller, which is a resource specifically trained in the business process management field. Another possibility, applicable in case event logs from systems that automate business process execution are available, is to directly retrieve the business process variants from event logs. This step can be achieved by using process mining tools, which are process centric applications aiming to discover real-life processes, by extracting knowledge from event logs [1]. ProM is, for example, an open-source process mining tool supporting a wide range of process mining techniques [20]. There are also an increasing number of commercial software products that offer process mining capabilities, e.g. ARIS Process

Performance Manager (ARIS PPM)[1], Disco [2], Reflect|one [3] or Reflect [4] [1].

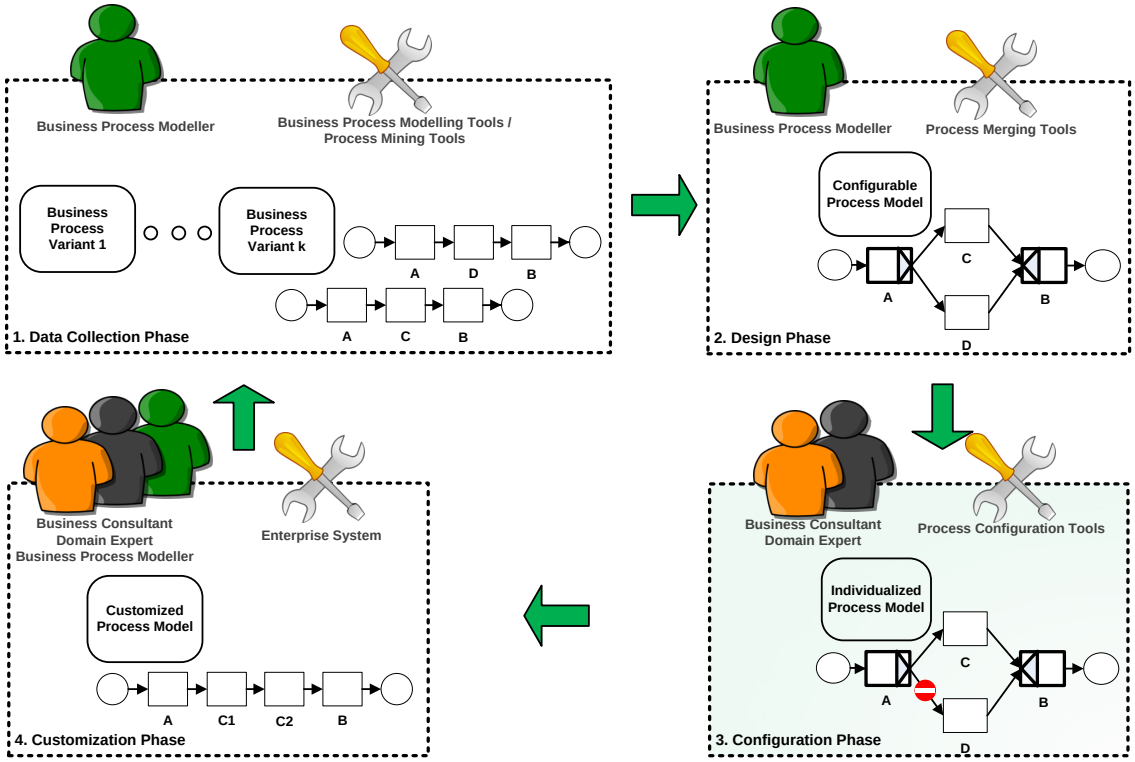

Figure 3.1: Process configuration life cycle

2. **Design Phase**

   The design step uses the various process families to generate the *configurable process models*. Again, the process modeller is the main actor, having the required knowledge to understand the configurability principles. It is important to automatically generate a configurable process model, given the process variations, as the manual execution of this phase is extensively time consuming and error-prone. To support this phase, already available process merging tools can be used [32]. However, automatic generation of merged processes remains very difficult, and research in this area is currently performed [36].

3. **Configuration Phase**

   While the two first steps can be considered part of the setup into using configurable process models, the configuration phase deals with the actual configuration of a model to suit the requirements of a particular customer, resulting in an *individualized model*. This

[1] www.softwareag.com
[2] www.fluxicon.com
[3] www.pallas-athena.com
[4] www.futuratech.nl

step is ideally done by the business consultants in collaboration with domain experts. The business consultants usually belong to the consultancy company implementing the configuration framework, while domain experts belong to the particular company for which the process is being individualized. These resources are not trained in business process modelling and do not have a broad understanding of the variation points present in the configurable process. Therefore, the configuration techniques used for this step need to decouple the low level variation points present in the process from the domain specific configuration options, and to relate more to the business environment in which both business consultants and domain experts operate. As a general requirement, a company will want to automate this step and to apply reusable information in order to decrease the implementation time in projects. This step is also the main focus of the current research project, and will be further detailed in this chapter.

The output of the configuration stage is the individualized process model. This process is the first attempt to comply with the requirements of the particular customer for which it has been configured. In case the fit between this process and the customer needs is sufficiently achieved, this process might be directly implemented in the organization of the customer and can be supported through the use of ES, such as ERP or Workflow Management Systems.

4. **Customization Phase**

   If the particular requirements of an organization cannot be met by the individualized version of the model, and the configurable model does not capture these requirements, then the model can be customized to a specific version. A *customized model* is derived from the individualized model, to which further adaptations are applied. The business consultants together with domain experts and process modellers are in charge of performing this step. The customized version of the process is implemented at the customer side, the effects of customizing the process model being also configured in the supporting ES. In the same time, the customized process is returned to the process family of the respective process model. In this way, the cycle can be replayed and the accuracy of the configurable models increases.

The process configuration life cycle covers the dynamics and evolution of configurable process models over time and can be used as a tool for training people into the use and implementation of configurable process models.

3.3 Questionnaire Process Configuration Methodology

We will now focus on the process configuration methodology, supporting the configuration step highlighted in Figure 3.1.

Making configuration decisions in order to individualize configurable models requires trained specialists able to understand the low level variation points captured in the configurable model. These specialists are typically process modellers. Moreover, as configuration decisions are influenced by information belonging to the specific domain of the model, domain experts and business consultants can offer input for configuring these processes.

Let us consider again the sales process introduced in Chapter 1. Choosing, for example, between engineering the product or not, requires knowledge about the type of order fulfillment strategy which is used, knowledge representing domain information. Further, a clear

understanding of the configuration principles and the configurable process modelling language used to capture the process, is needed. Nevertheless, it is unrealistic to assume that domain experts have the required process modeling skills to understand these formalisms. Therefore, a method able to guide business consultant and domain experts in taking configuration decisions over a business process, abstracting from the technical elements in the process, is needed.

La Rosa et al. [33, 34] proposed a method for process model configuration, based on a set of structured *questions*, that abstracts away from the configurable model. In this approach, the different requirements of the organization are called domain variability. This variability is captured independent of the process, through a set of domain facts. *Domain facts* are boolean variables that represent a feature of the domain. For example, we can define as domain facts the two fulfillment strategies supported in the model, namely engineer-to-order or make-to-stock. Further, domain facts having similar content (e.g. fulfillment strategies) are grouped into questions. By answering a question, all the facts related to the question are set. In our example, such a question could be *"What type of order fulfillment strategy is implemented?"*. Moreover, even if a fact can appear in multiple questions, it is set only once.

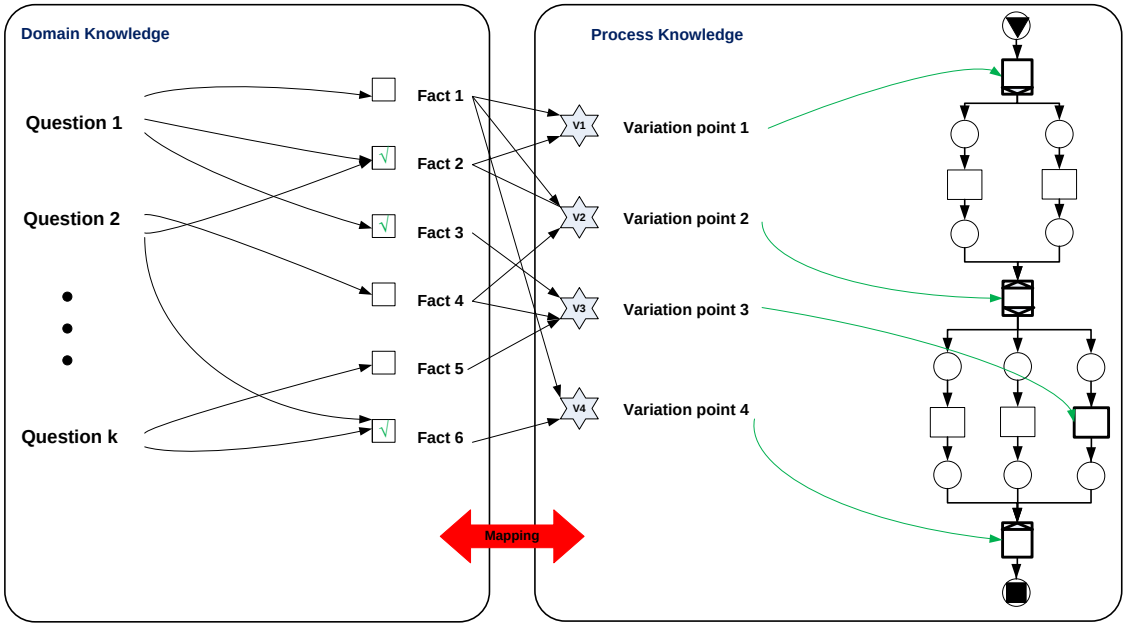

Figure 3.2: Questionnaire model

A domain fact has a default value (true or false) and can be marked as *mandatory*. A mandatory fact needs to be explicitly set by answering the corresponding question, whereas for a fact not explicitly marked as mandatory, the default value can be used without setting the fact through questions. Further, *dependencies* (partial or full) can be set between facts and/or questions. Dependencies determine the order of questions. These dependencies define the conditions that need to be satisfied in order to enable setting a fact. There are two types, namely *full* and *partial dependencies*. For example, if a fact *fully depends* on a set of other facts, it can only be set after all the facts in its preconditions (facts it depends on) have been previously set. In case of a *partial dependency*, only one of the facts in its preconditions

needs to be set before enabling setting the current fact. Further, dependencies over facts can be transferred to the question level in such a way that, assuming a fact x fully depends on another fact y, then a question that sets fact y needs to be put before a question that sets fact x. In this way, ordering between questions is determined.

Domain constraints represent the last concept in this approach. These constraints are logical expressions over domain facts which ensure that a correct configuration (according to the requirements of the domain) is always obtained.

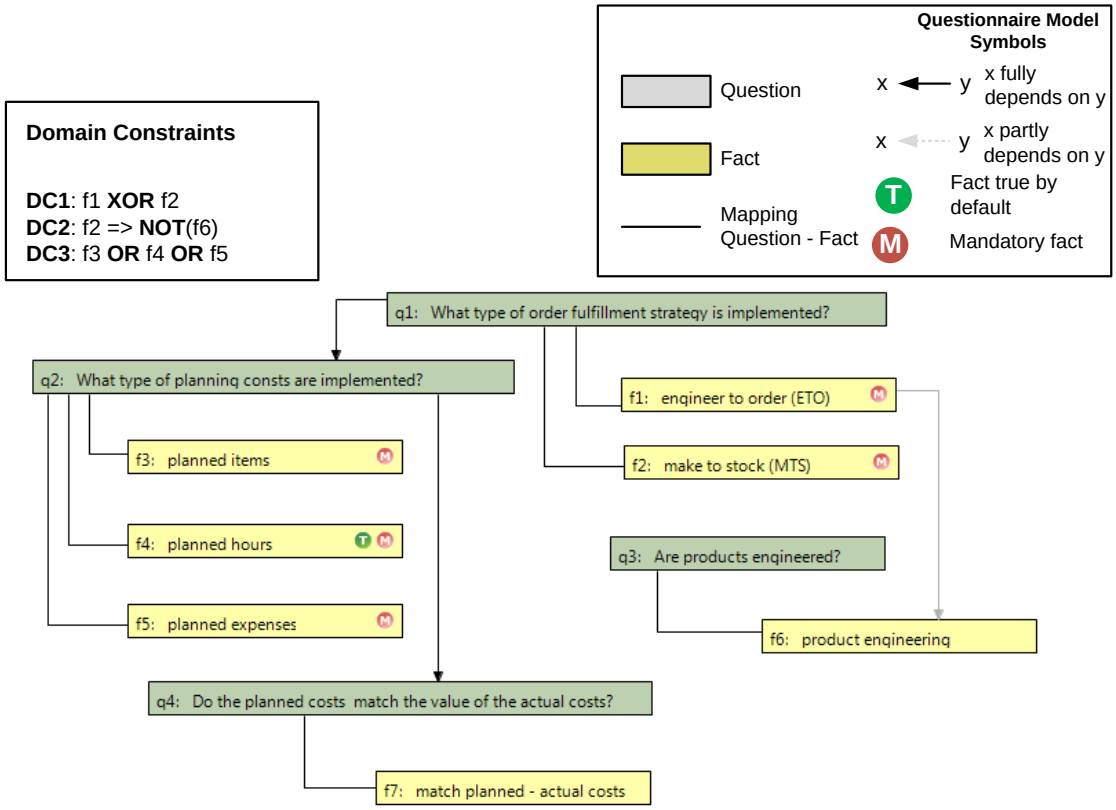

Figure 3.3: A Questionnaire model for the sales process

The last step consists in linking the domain knowledge with the configuration choices available within the configurable process model. The configuration choices captured by the configurable process model are represented as process facts, while the domain knowledge is represented through domain facts. Therefore, in order to link these two worlds, domain facts and process facts need to be mapped. One or several domain facts can influence a process fact, as it can be seen in Figure 3.2. A process fact can be seen as one configuration option of a variation point (configurable element) in the model. Therefore a process fact can be set to true, if the configuration option is chosen during process configuration or to false otherwise [18].

To illustrate these concepts, let us have a look at Figure 3.3, which presents a possible structure of questions and domain facts to capture the variability for the sales process. In this model the first question to be answered corresponds to the two possible fulfillment strategies

discussed, *engineer-to-order (ETO)* and *make-to-stock (MTS)*, represented by facts *f1* and *f2*. Both facts are mandatory, therefore a user answering the questionnaire needs to answer the first question. Planning the costs can be made as *planned items*, *planned hours* or *planned expenses*, indicated through facts *f3*, *f4* and *f5* respectively and grouped into question q2. Again, setting these facts is mandatory and planning hours is true by default. Fact *f6* concerns the *engineering of a product* and is the only fact grouped through question 3. This fact fully depends on f1, meaning that it can only be set after fact f1 has been set. Therefore, question 3 can only come after question 1. The last question investigates the *matching between the planned and actual costs*, setting fact *f7*. If f7 is set to true, than there is no need to enter actual costs in the process, as they correspond with the planned ones. Otherwise, actual costs need to be assessed.

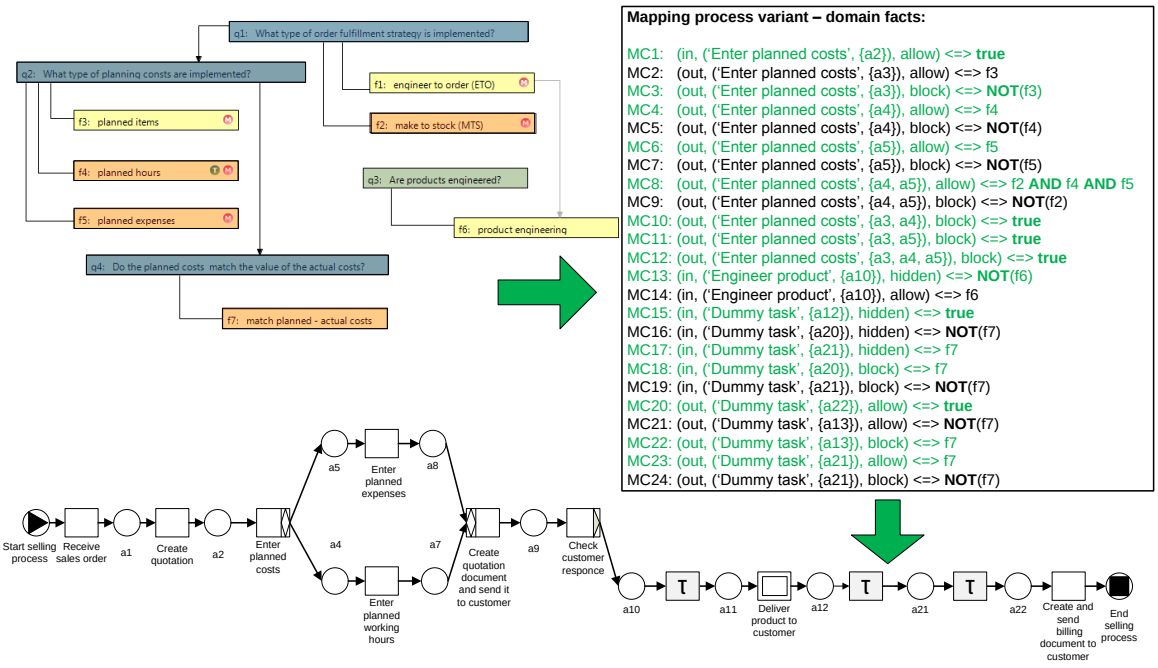

Figure 3.4: Sales process configuration for Company B

In order to ensure a correct valuation of facts, domain constraints can be also set. For our example, we have set three such constraints. The first constraint (DC1) ensures that only one of the two fulfillment strategies is chosen. Therefore, the process can only be configured for MTS or ETO separately. Further, we make sure that in the case of MTS strategy, fact f6 is automatically set to false, meaning that no product engineering is needed (DC2). This is indeed a true constraint as for a company implementing MTS, products are sold from the finished good stock. The last constraint (DC3) ensures that at least one of the three planned costs possibilities (f3, f4, f5) is chosen.

Further, domain facts are mapped to variation points in the process model, such that a domain fact can influence several variation points. This step is done by defining a set of mapping constraints (MC). Figure 3.4 illustrates the steps of configuring the sales process for Company B. First, the questionnaire is answered. The dark blue colored questions have been put to the domain expert. We can see that question 3 has not been asked. This is not

necessarily, since Company B implements the MTS strategy, and then the second domain constraint (DC2) is applied and f6 is set automatically to false. Further, the orange facts are the facts set to true through answering the questions. This valuation of facts is used against the mapping constraints and the variation points in the process are set, resulting in the individualized process for Company B.

Having explained the questionnaire methodology, we will present next the tool architecture supporting the entire process configuration life cycle introduced in Section 3.2.

3.4 Process Configuration Architecture

The principles of process model configurability are supported by an open source collection of tools gathered under the name *Synergia*. Synergia contains eight tools that provide support for the process configuration life-cycle, from the design of the configurable process models, until the individualization of the models by using the questionnaire approach [32, 33], and to the further enactment of these models through YAWL workflow engine. The toolset supports models defined in C-YAWL and C-EPC specifications. We will present in this section only the part of the architecture that supports the configuration of C-YAWL specification, which we are directly using during this research project. For the complete architecture, please refer to [32].

Each tool in Synergia is an independent application that handles a certain step in the process configuration and takes one or more files as input and generates a new file as output. The software architecture of Synergia toolset, specifying how the tools interact and which are the inputs and outputs corresponding to each of the tools, is depicted in Figure 3.5. We will further briefly describe each of the tools supported by the Synergia software architecture.

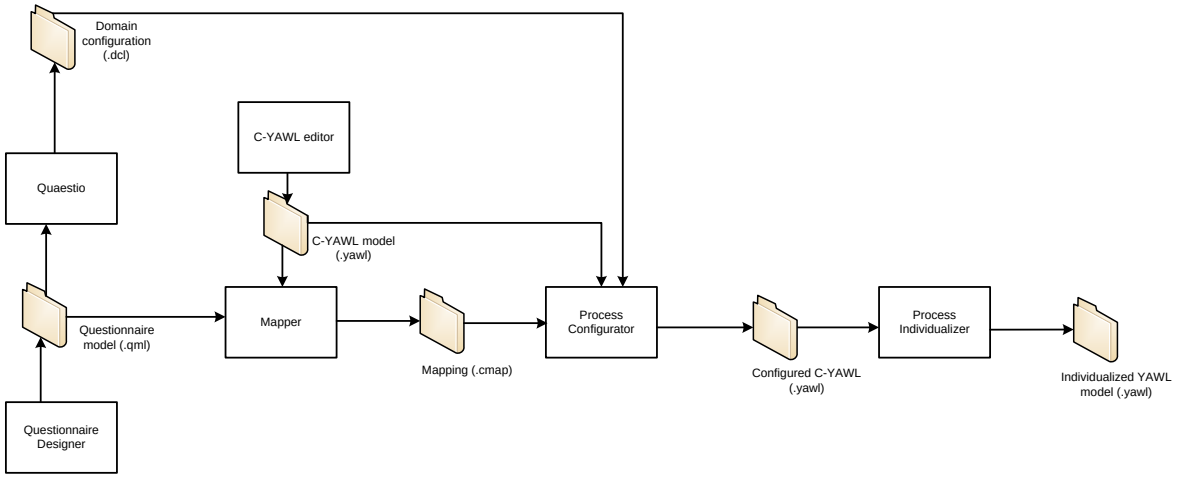

Figure 3.5: Synergia toolset software architecture

The *Questionnaire Designer* tool allows users to create questionnaire models graphically. The tool supports the creation of facts and questions, the grouping of facts into questions, setting dependencies between facts or between questions and defining domain constraints. Moreover, the resulting questionnaire model can be validated in order to avoid undesired

circular dependencies among facts or questions. The tool generates an XML file which encodes the information from the questionnaire model.

Quaestio is a tool which receives a questionnaire model as input and supports the interactive inquiring of questions to domain experts in order to set the collection of domain facts. The order in which questions are displayed to users is determined by the dependency constraints defined in the model; moreover, questions are posed only if they are relevant (there are unset facts belonging to the question that need to be answered). In case users want to change the answers given to a question, the tool offers the possibility to roll back answers. After answering all the questions (setting the entire collection of facts) the result is exported as a domain configuration (.dcl), which is also an XML serialization of the facts valuation.

The *C-YAWL Editor* allows the creation of configurable YAWL models. Tasks in a YAWL specification can be defined as configurable and for such tasks the input, output or cancelation ports can be configured.

The *Mapper* tool still needs to be developed and should support the mapping between the questionnaire model and the configurable process model. Such a tool receives as input a questionnaire model (.qml) and a configurable process model (either in the C-EPC or C-YAWL specification) and generates an XML file (.cmap), representing the mapping between the domain facts in the questionnaire model and the process facts in the configurable process model, by assigning each process fact a boolean formula over domain facts.

The actual configuration of the process model according to the answers given to questions by the domain expert is performed by the *Process Configurator* tool. It accepts a domain configuration (.dcl) - the valuation of domain facts resulting after answering the questions, a configurable process model (either in the C-EPC or C-YAWL specification) and the mapping between the domains facts and the process facts (.cmap) and configures the variation points in the process in accordance with the domain configuration. The resulting file is therefore a yawl model (.yawl) in which the configuration for each of the ports is encoded.

The last step in the configuration process is performed by the *Process Individualizer* tool, which implements an algorithm to transform each configured variation point in the model to its resulting variant; moreover in this last step, the unnecessary paths in the process, which may result from the configuration decisions, are eliminated from the model. The tool also guarantees, that providing it receives as input a well-formed model, it generates as output a well-formed model as well.

3.5 Conclusions

In this chapter, we have introduced the process configuration framework, detailing the three dimensions that constitute it. We have shown the main stages through which configurable process models pass during their evolution. Further, we have presented the questionnaire configuration methodology, which guides users in configuring processes through a set of questions. We completed this chapter by illustrating the Synergia architecture, the collection of tools that support the process configuration life cycle.

Our next goal is to critically discuss the shortcomings of the questionnaire approach and to propose a new process configuration methodology aiming to better support business consultants in using configurable process models. For covering the complete process configuration framework, current tooling needs also to be adapted to provide support for the application of our novel methodology.

Chapter 4

Knowledge-Driven Process Configuration Methodology

In this chapter, we introduce a novel methodology for configuring business processes, based on capturing the knowledge model that influences the processes. After critically assessing the questionnaire-based methodology and identifying its limitations, we introduce the notion of context models in Section 4.2. We first introduce context using a reference context model available in literature [45]. Further, we aim to validate the layers of the reference context model by evaluating several context models captured with the help of ontologies in literature. Based on our evaluation, we derive the requirements for structuring context and describe the modelling of context with ontologies. The context model-driven methodology is discussed in Section 4.3. In Section 4.4 we apply the methodology on the running example process. The chapter ends with a short summary and conclusions, in Section 5.5.

4.1 Analysis of Questionnaire Methodology

The questionnaire methodology makes a first attempt into guiding non-technical users in configuring business processes, bringing several improvements to the configuration phase. The configuration methodology based on questionnaire models abstracts from the concrete process modelling notations. The questions are expressed in natural language, and tools support the methodology.

One of the drawbacks of the questionnaire methodology is the fact that questions do not necessarily address the business context in which a process runs, but are generated rather ad-hoc. Moreover, questions are related to specific activities or group of activities in the business process model, and do not tackle the bigger picture of the business context of a particular organization. These questions are similar to examples like: "*Do you want to perform a warranty check?*" or "*Do you do quotes?*". Such questions rather focus on a specific check task in the process, or on including or not activities related to quotes in the process. As such, the questionnaire fails to create a clear image over the business model of organizations.

Decisions as to include a *warranty check* or not are dependent on the business environment in which the process is run. For example, a company would typically perform a warranty check if it provides warranty agreements to its customers. Therefore, considering the bigger picture and analyzing the business model that a particular company is implementing, can determine the appropriate business process for such a company. A method able to encompass business

knowledge about the company for which the process is individualized is needed. Information such as the industry sector in which the company operates, factors related to customers, suppliers or competitors represent elements normally considered by consultants in business process implementation projects.

Another problem with this approach is its lack of guidance into the steps that need to be followed to design a good questionnaire. Creating the questions to be included in the questionnaire is completely the responsibility of the persons developing the questionnaire. Most probably, if two different consultants create questionnaires for the same configurable business process, the results obtained will be different. In this sense, creating the questionnaire is a nondeterministic step in the configuration.

The environment in which a company operates has a direct impact on the business processes running within the company. Therefore, by placing the company in its specific business context, configuration decisions over processes can be automatically inferred. We propose a method able to derive values for domain facts automatically, by using a **context model** which incorporates knowledge about the context in which an organization runs its processes.

In the next section, we will detail the notion of context models.

4.2 Context Models

The idea of context circumstances having an impact on business processes has recently emerged in the Business Process Management field. Rosemann et al. introduce the concept of *context-aware business processes* [44]. The authors define *"the combination of all situational circumstances that impact process design and execution"* as the "*context*" or situation in which a business process is enclosed. Rosemann et al. argue that changes in the business context have an impact over the business processes running in that specific context.

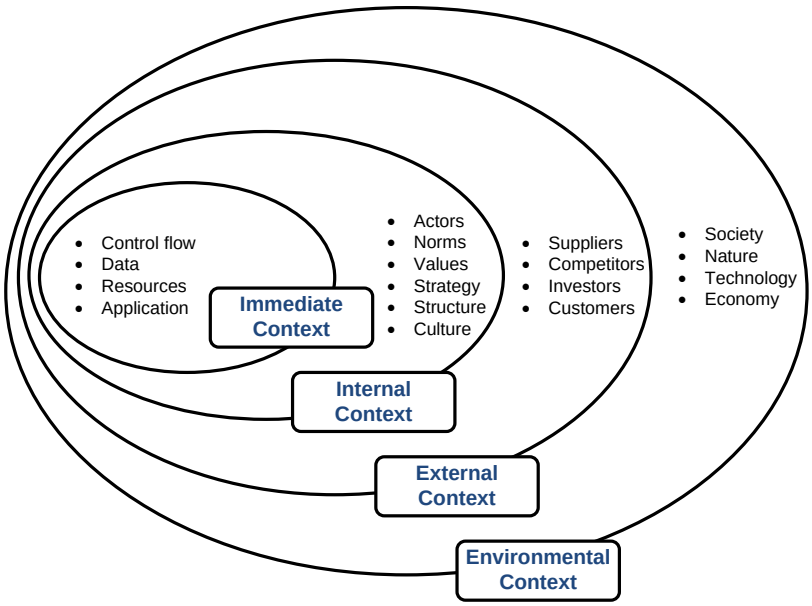

Figure 4.1: Context taxonomy

Context represents a collection of business variables that have impact over business processes. As a large variety of such variables exist, the same authors propose in [45] a layered model of context which structures the range of context elements into four categories. This proposed layered model follows a concentric distribution ("onion" model), as seen in Figure 4.1.

Immediate Context The first layer covers elements which are essential to the execution and understanding of a business process. This layer is supported by most existing modelling languages and includes elements as: control flow, data or resources.

Internal Context The second layer comprises the structure of the organization. It includes elements as actors, values, strategy or culture which have an impact over the business processes implemented within the organization.

External Context The next layer comprises the elements that are outside the control of the organization but are located within the business network where the organization operates. Such elements include: suppliers, competitors, investors or customers.

Environmental Context The last layer aims to capture the environment in which the organization is placed and relates to political, social or cultural factors.

Considering the above discussion, it is clear that the idea of context influencing the dynamics (design and execution) of business processes has already been investigated in the BPM field. We aim to bring this concept in the configurable business processes field, showing that the business context in which a business process is embedded determines the configuration decisions that need to be taken in the process. For example, if we consider our sales process running example, we can argue that by implementing this process for a manufacturing organization we configure the process differently than when implementing the process for an organization operating in the consumer goods industry. The industry sector in which an organization operates has direct impact on the configuration choices that need to be taken in the process. Similarly, aspects like the size of the organization, the product being delivered or the goals that aim to be attained by an organization can determine different configurations for the process.

Configuration decisions over business processes imply, as we have shown in Section 2.2.2, the elimination of unnecessary behavior from a configurable model, which initially comprises in its structure multiple variants of the same business process. Therefore, the configuration step has a direct impact over the structure of the individualized process. As such, in the setting of configurable process models, we consider *a context model to be an ontology model that captures the business context in which an organization operates and has impact over the structure (control flow perspective, data perspective, resource perspective) of the process model*. We will limit ourselves in this thesis to the control flow perspective, as the configurable modelling language considered, i.e. C-YAWL, only captures the variations at the control flow perspective.

As Rosemann et al. is one of the first authors introducing the concept of context in relation to BPM, we treat the onion model as a context reference model. The authors introduce the model only theoretically, therefore we are interested to investigate other approaches that use context models practically.

4.2.1 Context Models in Literature

In this section, we evaluate several projects which define and use context models, available in literature. We include in our evaluation the projects that model context either to use it in relation with BPM or for better aligning IT and business dimensions in an organization. The goal of this evaluation is to determine whether the onion model is a valid model which has applicability in practice. Further, we want to see if the structuring of context proposed in [45] has correspondence in the projects reviewed. All the concept models presented in this section have been captured with the use of ontologies and have focused mainly on the enterprise, as a central element.

The context ontologies proposed in research projects describe concepts related to the structure, activities, processes, information, resources or goals of an enterprise. These initiatives differ in the level of detail considered, scope or formalism used. We are going to briefly describe six context ontologies that we have considered in this research project.

One of the earliest initiatives is the *Enterprise Ontology* (EO) of the Edinburgh Group [49]. EO is intended to improve communication between the main actors of the organization, and to enhance operability between the different systems in the enterprise and between actors and systems. The first layer of EO is a meta-ontology which contains the foundational concepts (e.g. Entity, Relationship, Role), which are further used to define the elements describing the organization. The latter are split into four categories: *terms related to process and planning* (e.g. activity, planning, authority, resource allocation), *terms related to how organizations are structured* (e.g. person, legal entity, organizational unit), *terms related to high level planning for an enterprise* (e.g. purpose, mission, decision, critical success factor), and finally, *terms related to marketing and selling goods and services* (e.g. sale, customer, price, brand, promotion).

Similar to the EO project, the main goal of the *TOronto Virtual Enterprise* (TOVE) project is to create a data model that provides a shared terminology between the different actors of an enterprise, and that is able to automatically answer to a set of questions about the enterprise by using a set of axioms [16, 17]. TOVE is a complex project which comprises a set of integrated ontologies for modelling enterprises, including the *organization ontology*, *product and requirements ontology*, *activity ontology*, *resource ontology*, etc. The organization ontology consists of a set of basic elements that describe the organization. More specifically, an organization in TOVE consists of a set of *divisions*, a set of *organization-agents* (members of a division of the organization), a set of *roles* that the members play in the organization, and an *organization-goal tree* that specifies the goals (and their decomposition into subgoals) the members try to achieve.

The *Business Model Ontology* (BMO) proposes four pillars that describe the business activities within an enterprise [40]. These pillars are: *the product perspective*, which describes an organization in term of the products and services offered to its customers; *the customer interface*, which describes an organization by considering the targeted customers and the relations between the organization and its customers; *the infrastructure management interface*, which is concerned with the infrastructural or logistical issues of the company and the partnerships to which the company takes part in order to create and deliver value to its customers; and *the financial perspective* which describe the cost structure and the revenue model of the organization.

The *Core Enterprise Ontology* (CEO) project aims to gather a set of general enterprise concepts that can guide business experts in defining detailed enterprise ontologies [6]. These

concepts are structured into four main categories: the *active entities*, which describe as a main component the enterprise; the *passive entities*, which relate to the documentation or resources of an enterprise; the *transformations*, which contain the concepts of planning, performing and managing; and finally the *conditionals*, which describe concepts as state, goal, event and state of affairs.

Within the *Ontology Based Business Process Management* project, the main goal is to better align the IT and the business views of an organization through an ontology stack [28]. The business ontology included in this stack is further structured in three layers. The *Core Business Ontology Layer* defines basic concepts that are valid in an organization, regardless of its type or industry sector (e.g. resource, organization unit, role, business process). The *Industry Specific Business Ontology Layer* defines a set of industry-specific concepts. The last layer is the *Organization Specific Business Ontology Layer*, which includes the concepts that differentiate an organization from its competitors.

A recently concluded European project (2009) is the *Semantic Utilities for Process management within and between EnteRprises* (SUPER) which has as main goal to place BPM to the business level, by the deployment of Semantic Business Process Management (SBPM). SBPM aims to combine ontologies with reasoning mechanisms and available BPM techniques in a context-aware framework which allows companies to become more adaptive. The semantic framework supports the four phases of the BPM life-cycle (Modeling, Deployment, Execution, and Analysis) [7]. To offer a semantic representation of BPM, Filipowska et. al propose to divide the knowledge into three main groups: the *process*, the *organization-related* and the *domain-specific*, and to define the corresponding ontologies for each of these layers [15].

After reviewing several projects in which context models have been used, we grouped the main aspects considered in each of these context models with regard to the four layers of context proposed in the onion model (see Figure 4.1). As we could not find a clear separation between the external and environmental layers, we treated them as one. The result of our evaluation is described in more detail in Appendix A. The evaluation reveals that the layers in the onion model have support in all the other projects we inspected, showing that the model has applicability in practice. Moreover, its structure has been confirmed by the projects inspected, all the projects considering to split the various dimensions of context into a layered structure.

4.2.2 Context Model Structure

Having as a starting point the onion model presented in Section 4.2 and the context models we have considered from literature and introduced in Section 4.2.1, we propose three layers that describe the different facets of context.

The three context layers considered are:

1. The *process layer* includes the business concepts related to business processes, describing the activity flow, the data flow or the resource flow. This layer can include classes as business process, activity, event, resources, etc.

2. The *organizational layer* captures general concepts that describe the structure and internal activities of an organization. These include elements as organizational departments, organizational agents, goals, Key Performance Indicators, roles, etc.

3. The *external layer*, is oriented towards the external environment that encompass the organization and the relations that the organization has with other entities in the market, as suppliers, customers, competitors. This layer includes elements such as business network, industry sector, distribution channels etc.

We consider that these three layers create a comprehensive image of context and create a proper context model, which can be used in configuring business processes. The proposed layers can be mapped to the categories listed in [15, 28, 45]. For example, the *process layer* can be mapped to the immediate context layer proposed in [45], or to the process ontologies proposed in [15]. The *organization layer* can be mapped to the internal context layer proposed by Rosemann et al. [45], the organizational ontologies proposed in the SUPER project [15], or to the core business ontology layer from [28]. The last layer has no direct correspondence in the context taxonomy proposed in [45], but rather aims to integrate the last 2 layers from the onion model, namely the external context and the environmental context. However, it can be also mapped to the domain ontologies [15], or to the industry-specific ontology layer [28].

This section has provided an initial glimpse into the process of engineering a context model.

4.2.3 Modelling Context

A context model is therefore an ontology model which captures the business domain in which organizations operate.

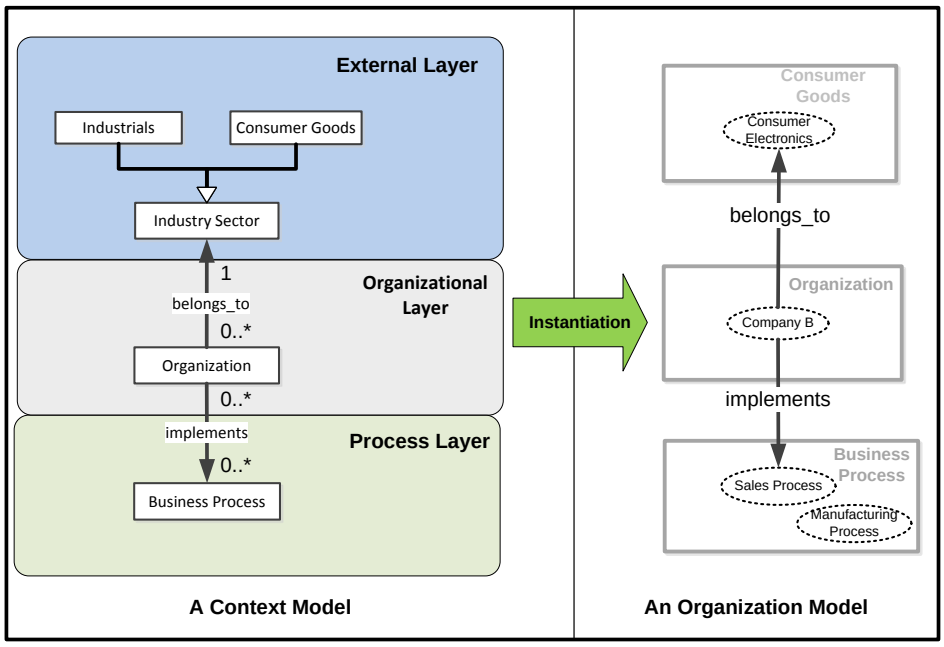

Figure 4.2: Example of a context model and an organizational model

We have opted for ontologies for several reasons. Ontologies allow the modelling of knowledge of a domain of interest, by capturing the concepts relevant to that domain and the

relationships between them. Moreover, ontologies are open structures, therefore the knowledge can be easily extended. Next, as we are not interested only to store knowledge, but also to actively use it in deriving configuration decisions over business processes, we are particularly interested in the support offered by ontologies for performing reasoning tasks.

In line with the description of ontologies we have provided in Chapter 2, a context model is an ontology model. To capture the characteristics of a particular organization, we instantiate the ontology model. We will call the instance of the context model in the remainder of this thesis as the *organization model*.

In Figure 4.2, we present a very simple context model and its instantiated organization model. The context model is structured on the three layers described in Section 4.2.2. We can see that three classes describe context, namely the `Organization`, the `Industry Sector` and the `Business Process` classes. When instantiating the context model, we define a particular organization `Company B` and position it in the business context, by defining the relations it has with other individuals. For example, we define that Company B belongs to the `Consumer Electronics` Industry Sector.

Further, we will explain how the configuration of business processes can be achieved by using context.

4.3 Context-Model-Driven Methodology

We propose a methodology of configuring business processes based on context models, structured on three main steps:

1. **Model context as ontology**

   The first step consists in engineering a *context model* with ontologies. A comprehensive context model, covering all the elements describing the business context of organizations is hard to achieve, due to its complexity. Industry sector specific characteristics or business process specific characteristics need to be captured in such a model, making the model more complex with each new industry sector or business process considered. Therefore, this model is usually tailored to the particular needs of the consultancy company engineering it, serving the organization to configure the business processes corresponding to the industry sectors in which it operates. As such, it is important for this model to allow for extensibility. In this way, in case the consultancy organization decides to provide consultancy services for a new industry sector, the knowledge corresponding to the new sector and the new business processes considered can be easily added to the context model. Once properly engineered, the same context model can be used by a particular consultancy company to perform business process configuration for many organizations.

2. **Define configuration rules for inferring domain facts**

   For maintaining an abstraction over process variations, we consider the use of domain facts in the same way as they have been used in the questionnaire methodology described in Section 3.3. A domain fact has associated a *default value* and a *description*. In order to configure a domain fact, a boolean value needs to be assigned to it, which might be different from its default value.

Instead of asking direct questions for configuring these facts (as the approach in the questionnaire methodology is), we aim to reason over the instantiation of the context model for a specific company. We propose to accomplish this by using a set of *configuration rules* over the knowledge in the organization model. A configuration rule is a logical expression over the elements in the organization model that has as a result the assignment of a boolean value to a domain fact. In case no rule can be applied for setting the value of a particular domain fact, the inferred value for the fact becomes equal to its default value. Both the domain facts and the configuration rules are captured using ontology concepts. The configuration rules are expressed using SWRL rules, whereas the domain facts are defined using datatype properties (see Section 2.4).

The configuration rules capture best practice scenarios. A best practice scenario decides upon the business process variant that the organizations sharing the same business context usually implement. By reasoning over the elements in the organization model, configuration rules set the value of domain facts, determining what business process variant should an organization implement.

For the creation of these rules, the expertise of business consultants and domain experts is necessary. These specialists have the proper practical experience in order to recognize the various best practice scenarios applicable for a particular business process.

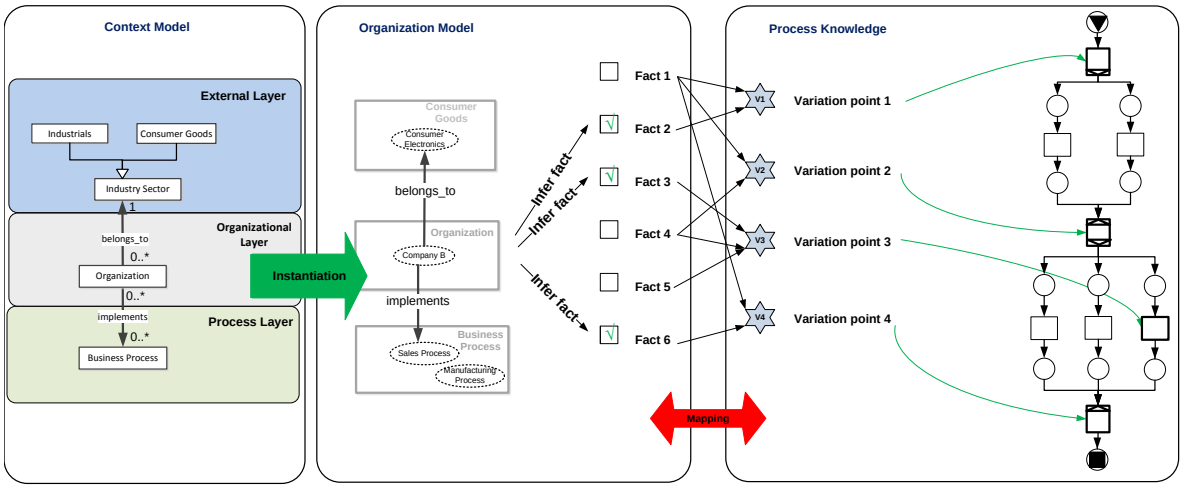

Figure 4.3: Context model

Unlike the creation of a questionnaire model, the generation of configuration rules is directly triggered from the business context. The best practice scenarios that determine what business process variants are implemented by organizations, given certain business contexts, are commonly accepted among domain experts and business consultants operating in a particular domain.

3. **Apply the configuration rules over an instantiation of the context model to obtain a valuation over domain facts**

   The next step consists in instantiating the context model for a particular company, resulting the *organization model*. This model can be potentially used in all process

configuration projects undertaken for that particular company. The last step is to apply the configuration rules over the knowledge captured in the organization model. The result of this step is a valuation over domain facts which is used to configure the process.

In this approach, configuration decisions are tackled by the business context in which the company operates, abstracting from the process model. The domain facts and configuration rules remain process specific, being directly related to a particular process. Nevertheless, the organization model is specific for a particular organization. Therefore, there is a clear separation between this model and the business processes.

Figure 6.6 summarizes the steps taken in configuring business processes by using the context approach.

4.4 Running Example

For a better understanding of the context-model-driven methodology, let us consider the sales process introduced in Section 1.1.2. The goal of this section is to configure the sales process by applying the three steps proposed in the methodology. First, a context model has been defined. This model captures the main elements that directly influence the configuration of the sales process. Further, a set of configuration rules have been considered. These rules infer the value of domain facts. Last, the organization model for Company B (see Section 2.2.2) has been generated, and the configuration rules have been applied over this knowledge. This results in a valuation of the domain facts. This valuation has been further used to generate the individualized sales process for Company B.

4.4.1 Context Model

The first step consists in defining a context model and modelling it as an ontology.

We will consider a simple context model, containing the three layers discussed in Section 4.3, which is an extension of the context model depicted in Figure 4.2. An overview of this context model, containing the main classes, structured on the three layers is given in Figure 4.4.

1. For the *Process Layer*, we have considered the class `Business Process`, which encompasses the business processes run by an organization.

2. The *Organizational Layer* contains three main classes: the class `Organization`, the class `Role` and the class `Department`. Further, the class Role is specialized into `Internal Role`, and `External Role`. A department is an administrative unit of the organization, which manages a set of activities in order to achieve a set of goals, and is composed by a group of people working together [3]. We consider a role to be a collection of responsibilities. An internal role is considered a collection of responsibilities at a functional level, and can be associated with an employee of the organization, while an external role can be associated with the organization itself (e.g. Manufacturer Role, Trader Role, Supplier Role, etc.), being a set of responsibilities at a structural level [6, 7, 28]. We have defined three properties as follows: the object property `has_role` connects

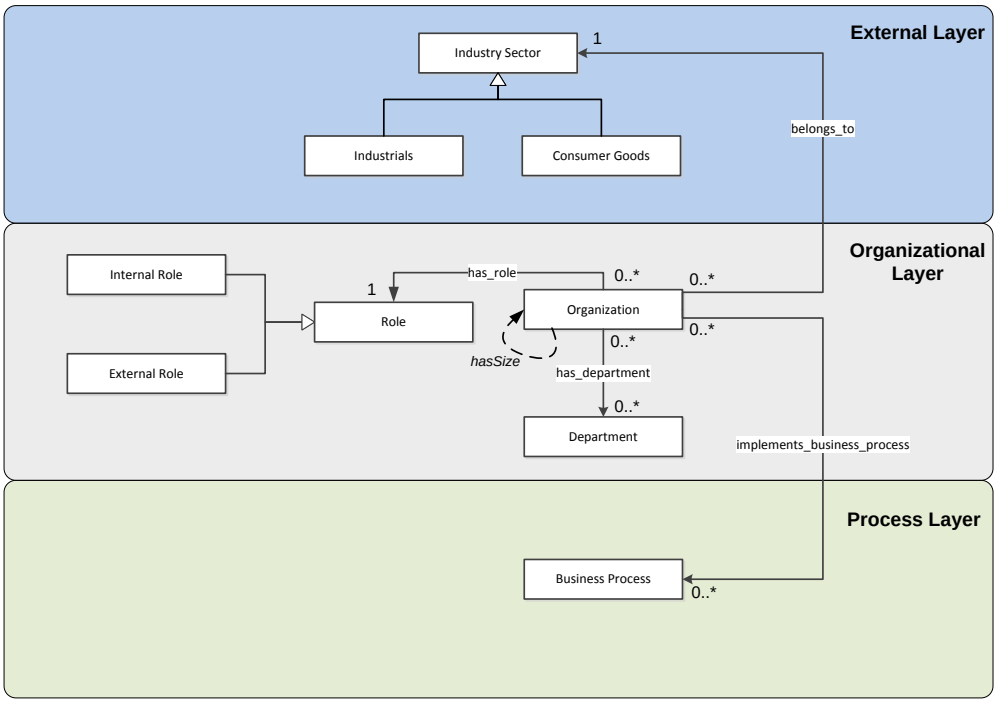

Figure 4.4: Example of a context model used for configuring the sales process

individuals of the class `Organization` with individuals of the class `Role`, the object property `has_department` links individuals of the class `Organization` with individuals of the class `Department`, while the datatype property `has_size` attributes a size value to individuals of the class `Organization`.

3. The *External Layer* contains the class `Industry Sector`, which has been further specialized into the sub-classes `Industrials` and `Consumer Goods`.

An organization (i.e. individual of the class `Organization`) is further connected with an industry sector through the object property `belongs_to` and with individuals of the class `Business Process` through the property `implements_business_process`. We have defined the object property `belongs_to` as *functional*, restricting an organization to belong to one industry sector. Similarly, the relation `has_role` is also *functional*, one organization having one major role on the market. The other object properties defined allow for many to many relations between individuals.

When a class is sufficiently specialized, individuals can be assigned to it. For example, we have assigned three individuals to the class `Department`, namely `Sales Department`, `Logistics Department` and `Engineering Department`. Similarly, we have added as a member of the class `Business Process` the `Sales Process`.

4.4.2 Configuration Rules

The second step consists in defining a set of configuration rules over the context knowledge, which infer values of the domain facts.

Concerning the domain facts, recall from Section 3.3 that we have defined a set of seven facts for the configuration of the sales process. These facts are: *f1: EngineerToOrder*, *f2: MakeToStock*, *f3: PlannedItems*, *f4: PlannedHours* or *f5: PlannedExpenses*, *f6: Engineering-Product* and finally *f7: MatchingPlannedActualCosts*.

Considering these facts, we have defined a set of seven configuration rules that infer values for them based on context knowledge. The configuration rules defined are output in Figure 4.5.

Configuration Rules

(R1) EngineerProduct(?x, true) ← Organization(?x) ∧ has_department(?x, EngineeringDepartment) ∧ EngineerToOrder(?x, true)

(R2) EngineerToOrder(?x, true) ← Organization(?x) ∧ belongs_to(?x, IndustrialMachinery) ∧ has_role(?x, Manufacturer)

(R3) MatchPlannedActualCosts(?x, true) ← Organization(?x) ∧ MakeToStock(?x, true) ∧ implements_business_process(?x, SalesProcess)

(R4) MakeToStock(?x, true) ← Organization(?x) ∧ belongs_to(?x, ConsumerElectronics) ∧ has_role(?x, Retailer)

(R5) PlannedItems(?x, true) ← Organization(?x) ∧ EngineerToOrder(?x, true) ∧ implements_business_process(?x, SalesProcess)

(R6) EngineerProduct(?x, false) ← Organization(?x) ∧ MakeToStock(?x, true)

(R7) MatchPlannedActualCosts(?x, false) ← Organization(?x) ∧ EngineerToOrder(?x, true) ∧ implements_business_process(?x, SalesProcess)

Figure 4.5: Configuration rules defined for the sales process

For example, the first rule assigns the value *true* to the domain fact *f1 : EngineerToOrder* in the case of a `manufacturing` organization which belongs to the `Industrial Machinery` Sector.

Similarly, rule R2 states that an organization which has as an industry sector the `Consumer Electronics` one and which is a `Retailer`, normally implements a MTS policy, and therefore this rule attributes the *true* value for the *f2: MakeToStock* fact, given that the described conditions are met.

In the next step, the configuration rules are applied on a concrete organization model, which describes the business context of Company B.

4.4.3 Applying Configuration Rules Using a Concrete Organization Model

The last step is to instantiate the context model into an organization model, tailored for a particular company, and to apply the configuration rules over this model in order to obtain a valuation over domain facts.

Considering the case of Company B, which we have described in Section 2.2.2, we are interested to define which are the contextual elements that apply for this specific company. In Figure 4.6 it can be seen that Company B is an individual of the class `Organization`. Company B `has_department` the `Sales Department`, which is an individual of the class `Department`, `has_role Retailer` in the market, which is modelled as an individual of the class `External Role`) and `belongs_to` the industrial sector `Consumer Electronics`, which

is an individual of the class `Consumer Goods`). Further, Company B is a small sized company (described by the data property `hasSize`) and `implements_business_process` the `Sales Process`. These relations describe the organization model of Company B, placing this organization in its context.

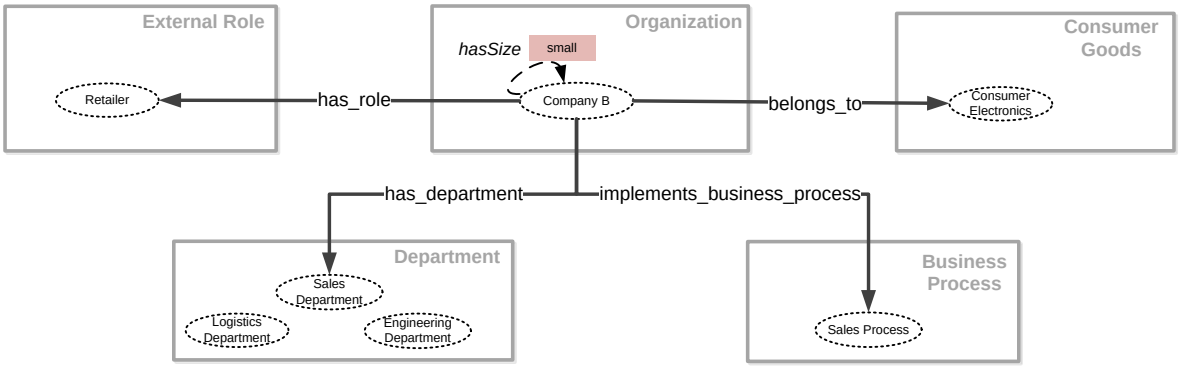

Figure 4.6: Organization model - instantiation of the context model for Company B

The next step is to apply the set of configuration rules over the Company B organization model in order to automatically infer the values for domain facts. From the seven configuration rules defined for this process, which we have provided in Section 4.4.2, three rules have been fired for the specific context of Company B, namely R3, R4 and R6 (see Figure 4.7). This resulted in setting fact *f2: MakeToStock* as *true*, fact *f6: EngineerProduct* as *false* and fact *f7: MatchPlannedActualCosts* as *true*. The remaining domain facts, which have not been directly set by firing the configuration rules, have been set to their default values. The resulted valuation over domain facts for Company B can be inspected in Table 4.1. The table outputs the default values initially set for the domain facts and the inferred values resulted after firing the configuration rules.

Table 4.1: Inferred values for domain facts for Company B

ID : Domain Fact	**Default Value**	**Inferred Value**
f1 : EngineerToOrder	false	*false*
f2 : MakeToStock	false	*true*
f3 : PlannedItems	false	*false*
f4 : PlannedHours	true	*true*
f5 : PlannedExpenses	true	*true*
f6 : EngineerProduct	false	*false*
f7 : MatchPlannedActualCosts	false	*true*

The configuration rules that have been applied for Company B, the valuation of domain facts and the resulting individualized sales process obtained for this company are depicted in Figure 4.7.

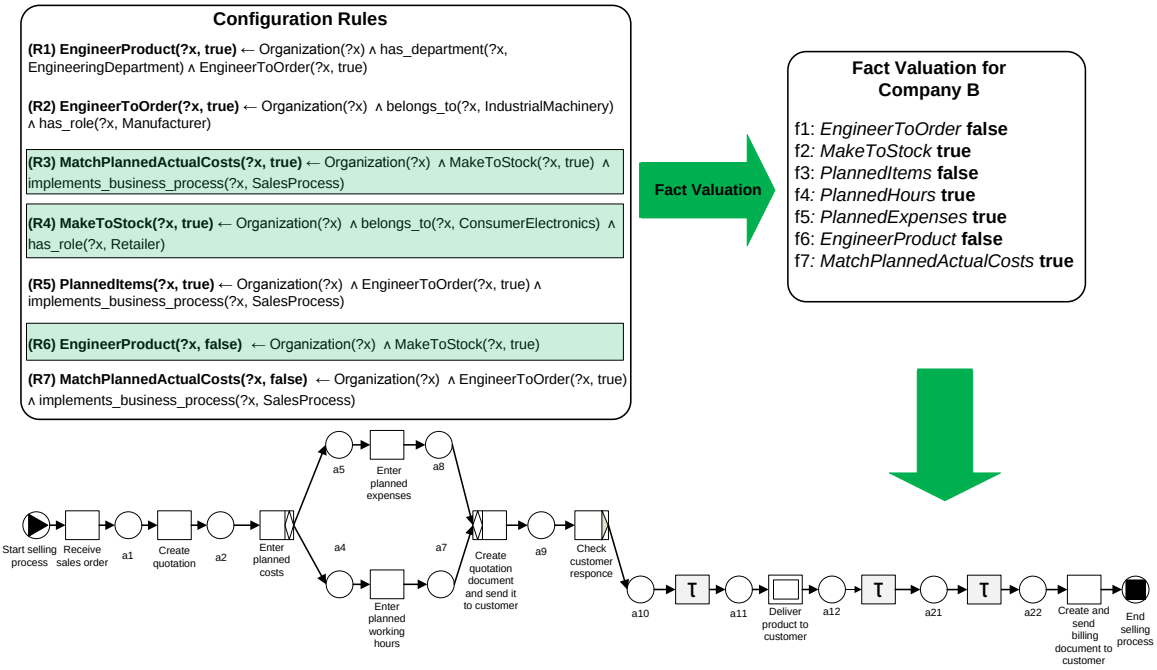

Figure 4.7: Resulting sales process for Company B

4.5 Conclusions

In this chapter, we introduce a business-like analysis for the business process configuration, based on context models, which abstracts from the process configurable elements.

In Section 4.2, we provided an answer to the first research question stated in the introduction of this thesis, namely *Which are the desirable characteristics of a relevant knowledge model for the configuration of business processes?*, by introducing context models. Next, in Section 4.2.2, we discussed to structure the context model into three layers, namely the *process layer*, the *organizational layer* and the *external layer*, answering to the second research question *What elements need to be considered in engineering the knowledge model?*.

In Section 4.3, we proposed a methodology for business process configuration based on context models. We used two separate views, the *context model* which models context at a general level with ontologies, and the *organization model*, which is an instantiation of the context model for a particular organization. Further, we derived configuration decisions about business processes by reasoning over the organization model, obtaining a valuation of domain facts. For reasoning, we used configuration rules, which realize therefore the mapping between the knowledge model and the domain facts. This provided an answer to research question 3, *What mapping exists between the knowledge model and the variations points of a configurable business process?*. The methodology proposed in this section provided therefore answer to research question 4 *How can the knowledge model be used during the process configuration step?* stated in the introduction of this thesis.

Finally, in Section 4.4, we showed that the methodology can be applied in theory, by using it to configure the sales process running example.

In the next chapter, we will discuss the process configuration architecture supporting the new methodology.

Chapter 5

Tool Integration

Having defined the new process configuration methodology, and having in mind the process configuration framework defined in Chapter 3, the next step is to adapt the process configuration architecture to offer support for the methodology. In this chapter we start by describing how the Synergia Tool Set can be adapted to provide support for the new methodology. Next, we describe the Domain Expert software tool, which we developed during this research project. In Section 5.4 we illustrate the configuration of the sales process, showing how the Synergia toolset can be used to provide tool support for the configuration of business processes based on context models. The chapter ends with conclusions in Section 5.5.

5.1 Process Configuration Architecture

In Chapter 4 we proposed a new process configuration methodology. The main innovation of this proposal is that it abstracts from the configurable process model and infers configuration decisions based on the business context of an organization. This approach masks the complexity of the process model and allows non-technical users to configure business processes.

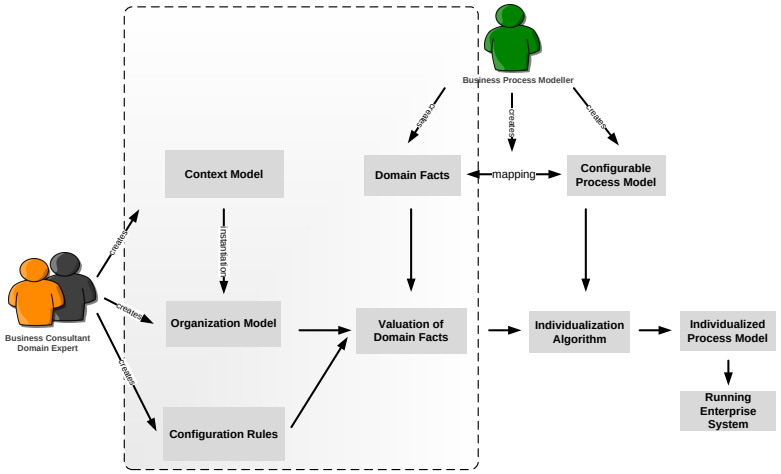

Figure 5.1: The configuration scenario based on context models. The dotted rectangle marks the elements that have been defined especially for the context-driven-methodology

We proposed to model the context as an ontology model and to instantiate this ontology model for a particular organization. We called the instance of the ontology model an *organization model*. Next, by applying a set of configuration rules over the organization model, we automatically infer values for domain facts. From the valuation of domain facts, the variation points in the configurable process model can be automatically configured, by using the generated mapping between the domain facts and the configuration points in the process. The last step remains to apply the individualization algorithm to eliminate the unnecessary paths from the configurable process model, obtaining an individualized process model. This model can be further used directly to implement a running enterprise system. An overview of the approach is shown in Figure 5.1.

The process configuration architecture introduced in Section 3.4 does not completely support the above configuration scenario. Therefore, we have extended the architecture of Synergia toolset to offer support for the new methodology. In order to accomplish this, we included tools which allow for modelling context with the help of ontologies, and for further automatically deriving configuration decisions by applying a set of configuration rules over a particular instantiation of the context model.

The adapted Synergia architecture can be seen in Figure 5.2. The tools that were previously supporting the questionnaire methodology, namely *Questionnaire Designer* and *Quaestio* (see Section 3.4), have been replaced with two additional applications, *Protégé Ontology Editor* and *Domain Expert* software tool.

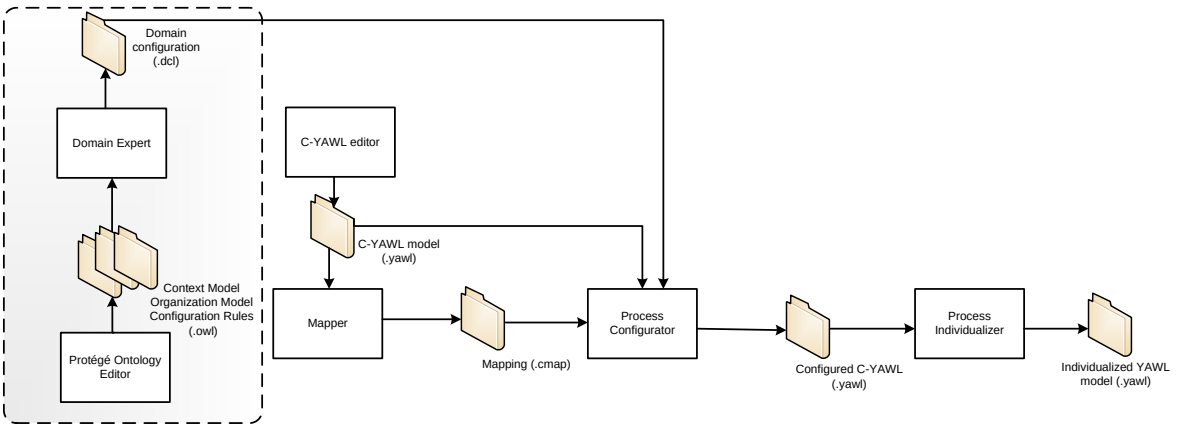

Figure 5.2: Synergia toolset software architecture supporting the context-driven process configuration methodology. The dotted rectangle delimits the new tools introduced in the context of Synergia

For modelling context with the help of OWL ontologies we have opted for the open-source ontology editor *Protégé*, developed by Stanford University in collaboration with University of Manchester [48]. The Protégé-OWL editor enables loading and saving OWL ontologies, editing and visualizing classes, properties and SWRL rules, defining logical class characteristics as OWL expressions, and editing of OWL individuals [48]. Considering its full support for working with ontologies, its mature framework (Protégé has reached over 175,000 of registered users), and its comprehensive documentation, we have selected it for modelling the context model, the organization model and for specifying the configuration rules.

We propose to split the context model, the organization model and the configuration

rules in three separate ontology files (.owl), which we call the *ontology stack*. Keeping the knowledge modular (in separate files) allows for *extensibility* and *reusability* of information. By maintaining the context model in a separate file, we keep this model general, allowing it to be used for the configuration of different business processes for different organizations (i.e. reusability). Moreover, new information can be added to the context model. In this way extensibility of knowledge is achieved. Each instantiation of the context model for a particular organization is also maintained in a separate file. As such, if several business processes need to be configured for the same organization, the organization model generated for that particular company is used for the configuration of these business processes (i.e. reusability). The configuration rules, which are process dependent, are also maintained in separate files. The context model remains therefore general and easily extensible, while the process specific information (configuration rules) or organization specific knowledge (organization model) are maintained separately.

For each business process configuration a collection of three files is always needed: the context model, the organization model and the configuration rules. This ontology stack generated with Protégé (.owl) is used in deriving a correct valuation over domain facts. For this purpose, we have developed the *Domain Expert* tool. This is an expert system that loads the ontology stack output by Protégé and exports a (.dcl) file, which is an XML serialization of the facts valuation.

Next, we detail the functional requirements that triggered the development of Domain Expert.

5.2 Domain Expert Functional Requirements

Domain Expert is an expert system able to extract knowledge from a set of ontology files, and to apply a set of inference rules over this knowledge base in order to draw conclusions. This is in line with our needs: the knowledge base is represented by the context and organization model, the inference rules are in fact the configuration rules, and the result is the setting of domain facts.

In order to build a system in line with the above needs, we defined a set of functional requirements that shall be supported and which are presented in this section. These requirements further guided us in the process of implemented the required tool.

1. *Domain Expert shall support the loading of the three OWL files which compose the ontology stack: the context model, the organization model and the configuration rules.* The rationale for this requirement lies in the integration link between Protégé, in which the ontology stack is modelled, and Domain Expert.

2. *Domain Expert shall check upon the consistency of the context and organization models.* In order to derive correct configuration decisions, it is important to reason over a consistent knowledge base.

3. *Domain Expert shall support the visualization of the organization model, including classes, individuals, data properties and object properties, through the Graphical User Interface (GUI).* This information constitutes the knowledge base of the system and provides to the end user an image over the organization model.

4. *Domain Expert shall support the visualization of domain facts through the GUI.*

5. *Domain Expert shall support the visualization of configuration rules through the GUI.*

6. *Domain Expert shall determine the value of domain facts by applying the set of configuration rules over the knowledge base.*

7. *Domain Expert shall support the visualization of the inferred valuation for the domain facts through the GUI.*

8. *Domain Expert shall allow users to change the inferred value of domain facts.* In case exceptions to the configuration rules take place in practice, business consultants need to have the possibility to further adapt the inferred values for domain facts.

9. *Domain Expert shall allow the export of an XML based file (.dcl) which contains the valuation of the domain facts.* The format of the file is especially chosen to conform to the structure of the domain configuration file (.dcl) in order to integrate the tool within the Synergia architecture.

By meeting the above functional requirements, Domain Expert is able to support the new context-driven-methodology and to help providing end-to-end support for the process configuration life-cycle. The input files (the three OWL files containing the context model, the organization model and the configuration rules) and output file (the domain configuration file (.dcl)) for Domain Expert are depicted in Figure 5.2. Next we detail the software architecture and user interface of Domain Expert.

5.3 Implementation of Domain Expert

For the purpose of providing tool support for the process configuration methodology based on context, we developed the Domain Expert tool. In this section, we detail the software architecture and user interface of this tool.

5.3.1 Domain Expert Software Architecture

Domain Expert adheres to the *rule-based system* architectural pattern, a typical architectural pattern for building expert systems. This consists of three aspects, the knowledge base which represents the data, the rules which are represented in the form of conditions, and an engine which applies the rules over the known data. By the action of rules, new facts are asserted, which can further trigger other rules [5].

Figure 5.3 presents the system architecture for the proposed tool. Four communicating modules ensure the functionality of Domain Expert, namely the graphical interface module, the ontology manager module, the inference engine and the fact manager module. The tool accepts a collection of ontology files (ontology stack) as input, and provides as output a domain configuration file.

Through the *graphical interface*, the user can visualize the relevant information (e.g. organization model, configuration rules, domain facts) and can interact with the application. The *ontology manager* loads the knowledge from the ontology stack into memory. Further, this module is in charge with updating the ontology in case changes occur as a result of the reasoning process. The *inference engine* module manages the application of the configuration rules over the knowledge captured in the organization model. For consistency reasons,

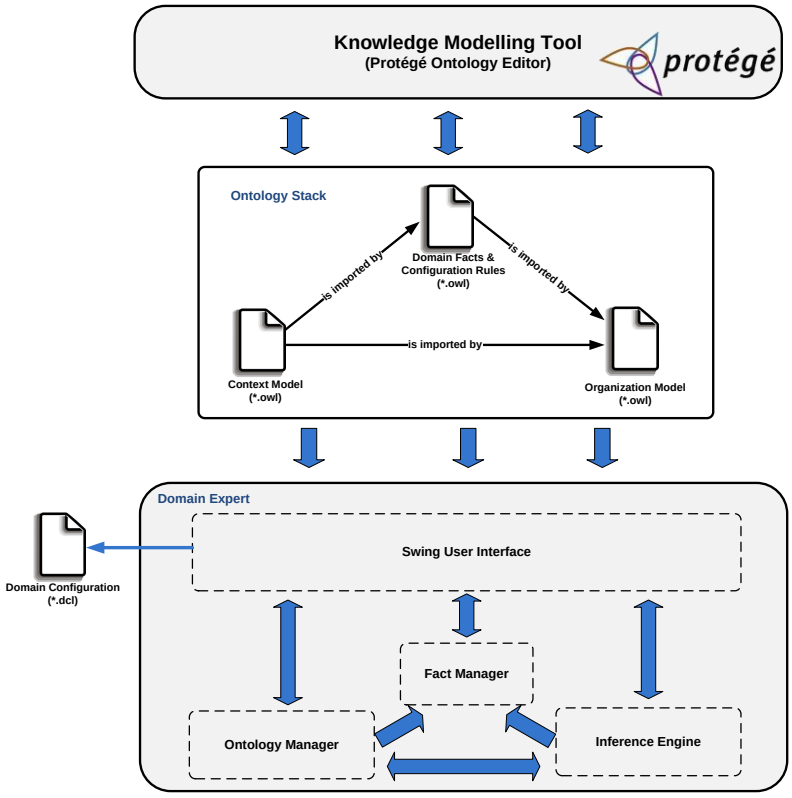

Figure 5.3: Domain Expert software architecture

this module also checks the correctness of the knowledge base and provides feedback to the user. Lastly, the *fact manager* module handles the domain facts, being in charge with their representation into data structures, the update of their values, and the export of the final valuation into an xml serialization, which is the domain configuration file.

The proposed architecture satisfies the functional requirements identified in Section 5.2 as follows:

1. Domain Expert loads the context model, the organization model and the configuration rules from a collection of 3 OWL files, satisfying requirement (1).

2. The tool checks the consistency of the loaded ontology model, by using the inference engine module, in line with requirement (2).

3. Domain Expert allows for the visualization of the organization model, domain facts and configuration rules, as well as the inferred value of domain facts, through the GUI, in line with requirements (3), (4), (5), and (7).

4. The inference engine module applies the configuration rules over the knowledge base and derives a valuation over domain facts. This fulfils requirement (6).

5. Through the graphical interface, users can change the value of domain facts before exporting the domain configuration, satisfying requirement (8).

6. Finally, the proposed tooling exports the valuation of domain facts as a .dcl file (requirement (9)).

Next, we present the user interface of Domain Expert.

5.3.2 Domain Expert User Interface

During the software design process, a series of decisions have been made, which we will briefly describe in this section. These design decisions affect the degree to which the software can support the novel methodology, but do not lower the validity of the theoretical method.

1. We opted for the use of OWL as an ontology language, as OWL does not only support the implementation of ontology models, but comes also integrated with a collection of tools which can create a complete environment for building a software application. First, OWL is a mature language endorsed by the World Wide Web Consortium (W3C) and is supported by several semantic editors (e.g. Protégé) and semantic reasoners (e.g. HermIT, Pellet, FaCT++). Second, building applications based on OWL ontologies is supported by the OWL Application Programming Interface (API), which is a comprehensive development tool in this sense [9]. Third, OWL enables automatic reasoning over the inconsistencies in the ontology [50], through the use of reasoners. Reasoners are OWL tools that can automatically compute the consistency/inconsistency of an ontology.

2. The expert system built to support the context-model-driven methodology is based on the open world reasoning paradigm. This affects the way in which the reasoner can answer to queries. In the open world assumption paradigm, the reasoner cannot infer, for example, that two individuals are distinct if they have different names, unless it is specified in the ontology [38].

3. Another design decision was to use Semantic Web Rule Language (SWRL) syntax for the expression of rules [10]. SWRL are supported in Protégé and allow the expression of rules in terms of OWL concepts. SWRL also introduces a series of limitations, as its share with OWL of the open world assumption paradigm, monotonic inference (SWRL rules cannot be used to modify already existing information in the ontology) and its difficulties in supporting negation [10].

From a technical perspective, we chose Java as the implementation language. For building the user interface, we used Swing Application Programming Interface (API). For manipulating ontologies, we used the Java OWL API (version 3.2.3), maintained by the University of Manchester [39]. For performing reasoning tasks, the Pellet API (version 3) was chosen, an open-source OWL 2 reasoner for Java [35]. The complete tooling totals approximately 3500 source lines of code, distributed across 16 files.

The user interface of Domain Expert guides the users through the logic of deriving the values of domain facts for a particular business process, and it can be seen as a wizard process. The tool is organized into four tab views.

1. As mentioned earlier, the tool takes as input a collection of OWL files (ontology stack). In the current version of the system, three separate OWL files need to be provided as input, namely the file which contains the context model, the file which contains the

organization model and the file in which the domain facts and configuration rules are specified. The files need to be loaded in the sequence corresponding to their dependencies (Figure 5.3), namely the context model ontology first, the file specifying the domain facts and configuration rules second, and the organization model ontology third. The first tab handles the loading of these input files, providing feedback to the user in case the loading sequence is incorrect (Figure 5.4).

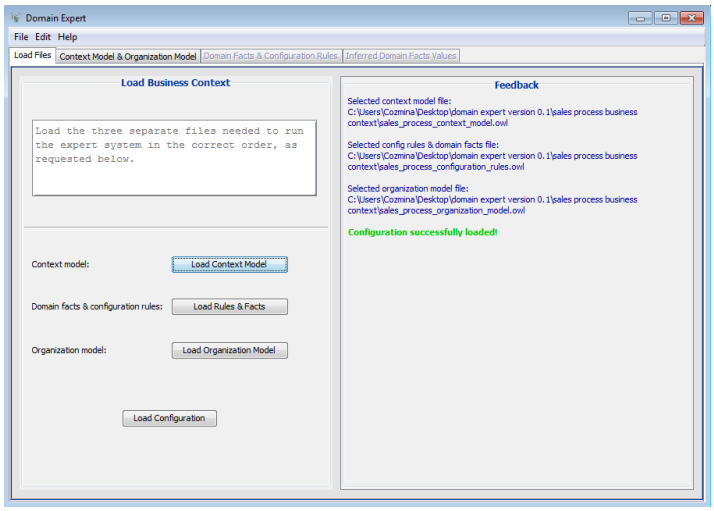

Figure 5.4: Load files tab in Domain Expert

2. The *Context Model & Organization Model* tab presents, in the case of a consistent ontology model, the taxonomy of classes in the context model and the individualizations available for the current company. If the ontology is inconsistent, the user is notified and the process needs to be restarted, new files being provided to the software tool.

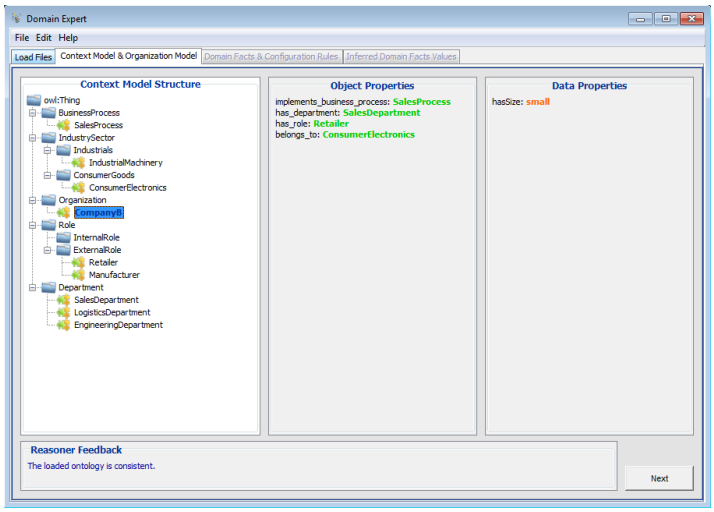

Figure 5.5: Context Model & Organization Model tab in Domain Expert

This tab is organized into three main panels which provide a comprehensive view over

the context and the organization model (Figure 5.5).

In the *Context Model Structure* panel, different shapes and colors are used to differentiate between the classes in the ontology and the individuals instantiating these classes. Individuals are displayed at the lowest level in the taxonomic tree, as it can be seen in Figure 5.5. When the `Company B` individual is selected, the next two panels (the *Object Properties* panel and the *Data Properties* panel) are populated with information which describes the business context of `Company B`. For example, we can see that `Company B` is related through the object property `belongs_to` to the `Consumer Electronics` Industry Sector, which is an instantiation of the class `Consumer Goods`.

3. Further, the defined domain facts for the configurable process are outputted in the *Domain Facts & Configuration Rules* tab (Figure 5.6). This tab has a similar interface with the precedent one, being also organized in three main panels. The set of domain facts is output in the *Domain Facts* panel. When a domain fact is selected by the user, the information relevant for that fact is presented. This includes the set of configuration rules that either set or test the value of the selected fact (*Configuration Rules* panel) and the specific properties of the fact, i.e. its default value and its description (*Domain Fact Properties* panel). This information has been previously defined in the Protégé ontology editor, therefore Domain Expert does not allow users to make changes here over this data. If changes need to be made in the case of configuration rules, domain facts or context knowledge, these can be handled using Protégé.

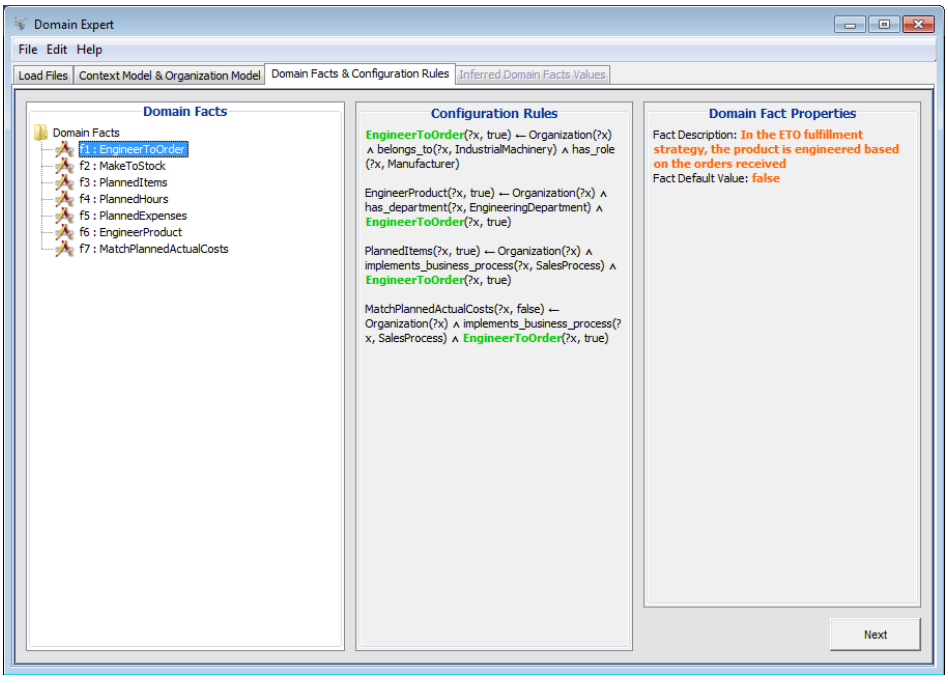

Figure 5.6: Domain Facts & Configuration Rules tab in Domain Expert

4. The last step consists of automatically inferring the values for domain facts by applying configuration rules over the organization model (*Inferred Domain Facts Values* tab). In this view (Figure 5.7), when a user selects a domain fact from the list of facts displayed

in the interface, the default value and the inferred value for that particular fact are shown. Moreover, the rules applied by the reasoner for reaching its conclusion with respect to this fact can be inspected (*Fact Value Explanation* panel). Users are allowed to change the inferred value of facts according to their needs, in case they want to further adapt the configuration decisions. The configuration process completes once the user agrees with the domain facts' valuation and decides to make no further changes. The domain configuration can be exported to an XML file (.dcl) which keeps track of the facts set and their related information.

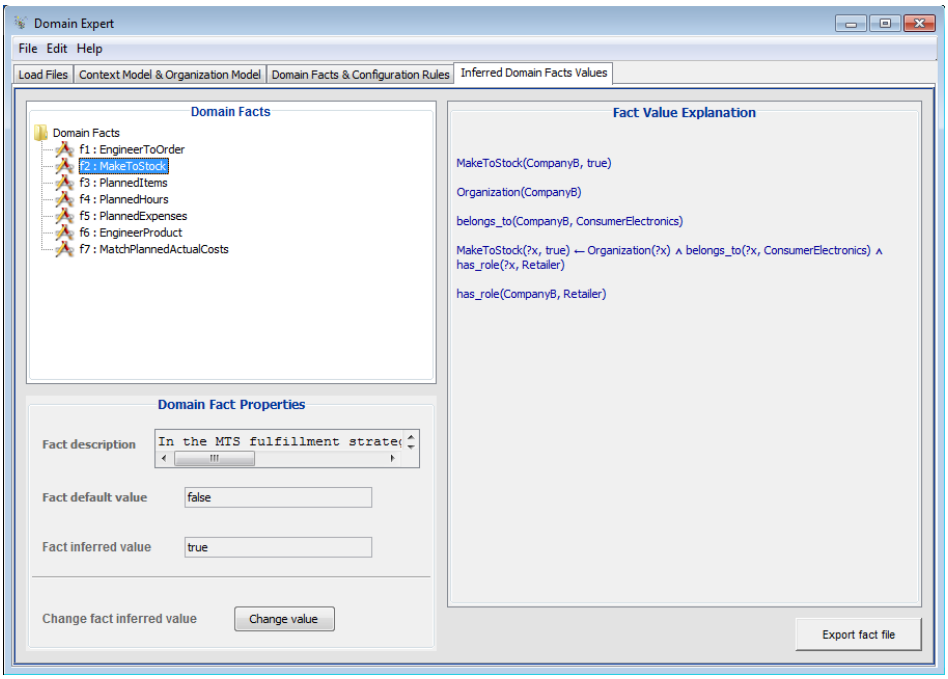

Figure 5.7: Inferred Domain Facts Values tab in Domain Expert

The Domain Expert software, the ontology stack for the sales process and a presentation movie of the knowledge-driven-configuration methodology can be found at `http://www.win.tue.nl/~jmw/process-configuration`.

5.4 Running Example

For a better understanding of the contribution brought by Domain Expert to the process configuration phase, let us now demonstrate how the configuration of the sales process example can be handled with the Synergia Tool Set presented in Section 5.1.

Recall from Section 3.3 that we have defined a set of seven domain facts that handle the configuration decisions which can be taken in the case of the presented sales process. These facts capture the fulfillment strategies used (*f1: EngineerToOrder*, *f2: MakeToStock*), the cost planning policies (*f3: PlannedItems*, *f4: PlannedHours*, and *f5: PlannedExpenses*), whether the product needs to be engineered or not (*f6: EngineerProduct*), and whether there is a match between the planned and actual costs (*f7: MatchPlannedActualCosts*). We also

considered in Section 4.4 a context model, a set of configuration rules that infer the described domain facts, and an instantiation of the context model for Company B (organization model). In a first step, this knowledge needs to be represented in an ontology structure, using *Protégé*. This results in a collection of OWL files (ontology stack) which are further used as input for *Domain Expert*.

Domain Expert allows for the visualization of the company specific organization model in the Ontology Structure panel (Figure 5.5). We can see that `Company B` is an instance of the class `Organization`. We can also see, for example, that the `Industry Sector` class has as subclasses the `Industrials` and `Consumer Goods` industry sectors, which further have as individuals the `Industrial Machinery` and `Consumer Electronics` industries.

Further, the next two panels show the Object and Data Properties which define the organization model of Company B. As such, `Company B` is a `small Retailer organization`, operating in the `Consumer Electronics` industry, and has a specialized `Sales Department`.

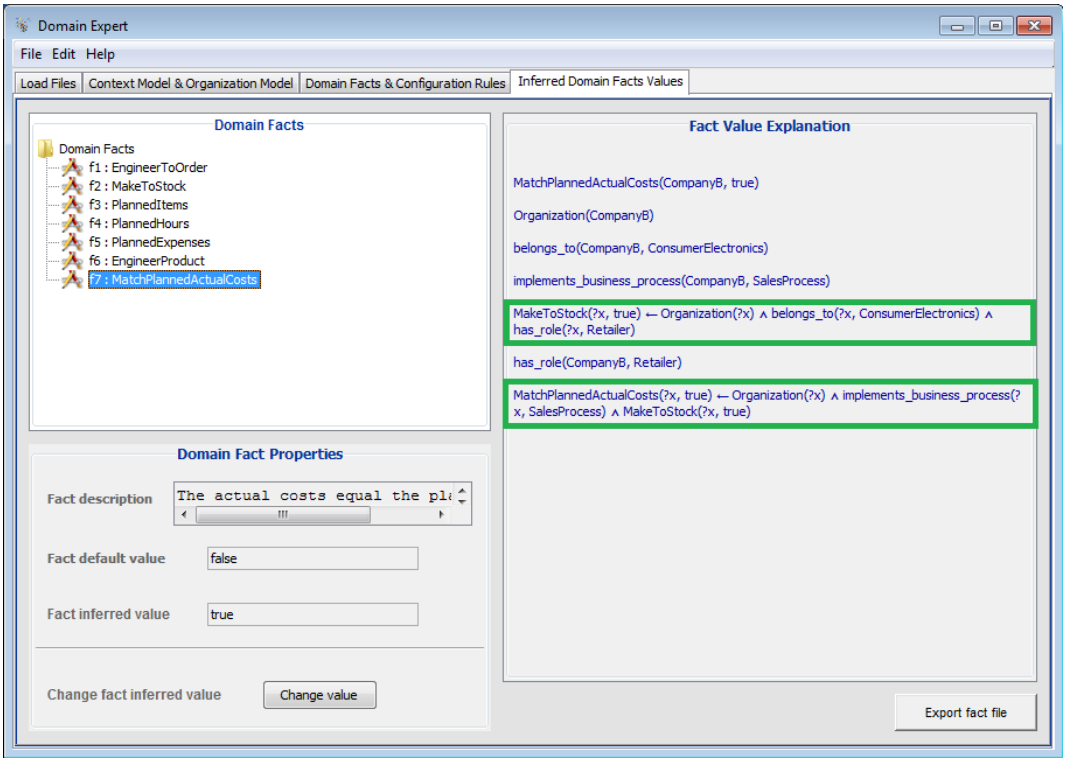

Figure 5.8: Company B valuation of domain facts output in Domain Expert

When deriving the valuation of the specified domain facts, the reasoner inspects the organization model of Company B and fires the rules that are applicable. For example, considering domain fact f7, the reasoner applies two configuration rules for setting this fact as *true* (depicted in green squares in Figure 5.8). The first rule is the one that sets as *true* the domain fact *f2:MakeToStock*, while the second rule sets domain fact *f7*. The rule setting fact *f7* states that an organization implementing the make-to-stock fulfillment strategy usually has a match between the planned and actual costs when selling goods.

f7: MatchPlannedActualCosts(?x, true) $\Longleftarrow$

(*Organization*(?*x*)

$\wedge$ *MakeToStock*(?*x*, *true*)

)

As the conditions for firing both rules are met, fact f7 is inferred by Domain Expert as being *true*, while its default value was set to $false$.

There are also cases, as for example the case of fact *f3 - PlannedItems*, in which no configuration rule can be fired to set this fact. In this situation, Domain Expert automatically sets for this fact its default value, in this case, $false$. The final valuation, as inferred by Domain Expert, is: f1 - $false$, f2- *true*, f3 - $false$, f4 - *true*, f5 - *true*, f6 - $false$, f7 - *true*.

Whether a business consultant decides that the values of one or several domain facts need to be changed to reflect the situation of a particular organization, he/she can choose to change the inferred values for those facts. Let us assume that Company B works with three different shipment companies, each requesting slightly different prices for their deliveries. Company B always selects the shipment company which can process the shipment order the fastest, in order to deliver the goods as soon as possible to its clients. Such a shipment policy would cause a small difference between the initial estimation of shipment costs (which can be done, for example, based on the average sum of the three possible costs) and the final shipment costs (which would be the costs requested by the selected shipment company). This could be an exceptional situation of Company B and in this case, the business consultant can decide to change the value of domain fact *f7 - MatchPlannedActualCosts*. However, if this situation represents a common practice among similar organizations, then the consultant can choose to further extend the context model with this knowledge and to add an additional configuration rule which considers these situations.

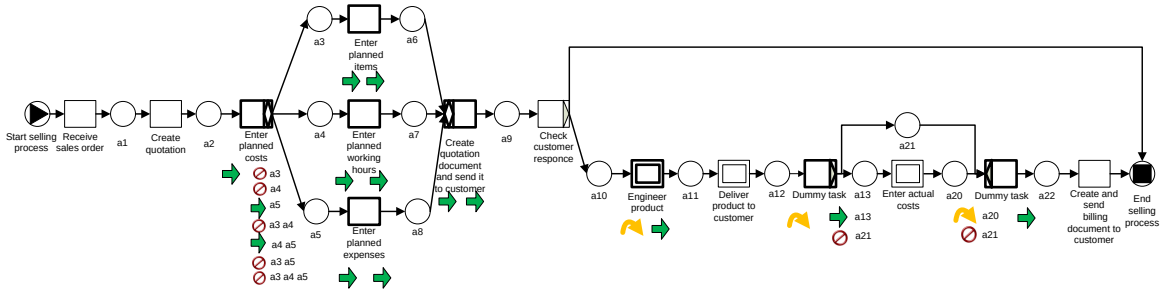

Figure 5.9: The configuration decisions of Company B according to the domain facts valuation inferred by Domain Expert

Considering the above situation, the new valuation for domain facts becomes f1 - $false$, f2- *true*, f3 - $false$, f4 - *true*, f5 - *true*, f6 - $false$, f7 - $false$. This valuation is exported as a domain configuration file (.dcl) from Domain Expert. The next step is to load the .dcl file into the *Process Configurator* together with the corresponding configurable process model and the mapping between domain facts and low-level variation points in the process. This tool uses the mapping to generate the configuration decisions over the configurable process

model (in terms of setting to hide, block or allow the various input/output ports for the configurable elements in the model) and adds these configuration decisions to the process model, as annotations. The decisions resulting from our example can be seen in Figure 5.9.

In the last step, the *Process Individualizer* considers the annotations of the configuration decisions in the process model and runs the individualization algorithm to eliminate the unnecessary behavior in the process, resulting in a YAWL specification of the sales process for Company B. For our example, we show the resulting YAWL model for Company B in Figure 5.10.

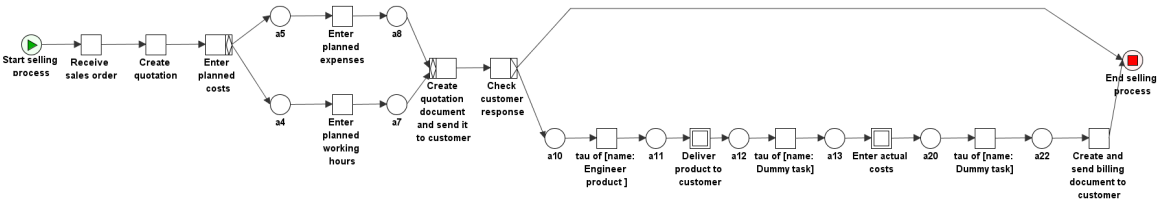

Figure 5.10: The individualized sales process for Company B from YAWL Editor

The individualized YAWL model, enriched with the data and resource perspectives, is directly executable in the *YAWL workflow engine*. Therefore, the result of the configuration phase is a running system.

For illustrating the run-time perspective of the resulting system, we show the execution of a case of the individualized sales process in the workflow system. Let us consider that the execution of the sales process is handled within the Sales Department, by specialized sales operators. The tasks that need to be executed are offered and allocated to one available sales operator by the system, based on the shortest queue allocation strategy. The human actor decides when to start each of the work items allocated to him/her and completes the work items one by one. We consider the scenario in which a sales order is received, a quotation document based only on the estimated working hours is created and send to the customer and the customer decides to reject the quotation, the case ending immediately. The path that this case takes through the process is depicted in Figure 5.11.

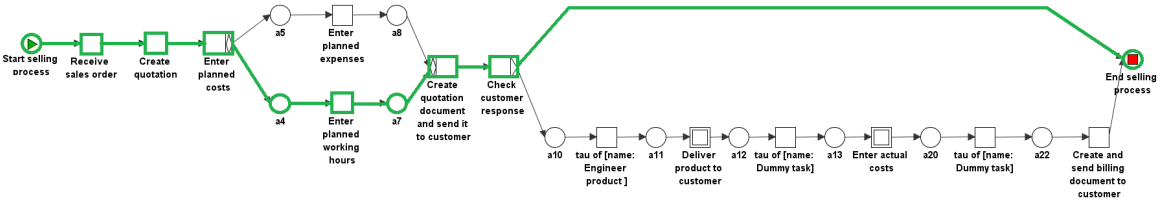

Figure 5.11: Run-time scenario of the sales process

The step by step execution of the described scenario in the YAWL system is shown in Figure 5.12.

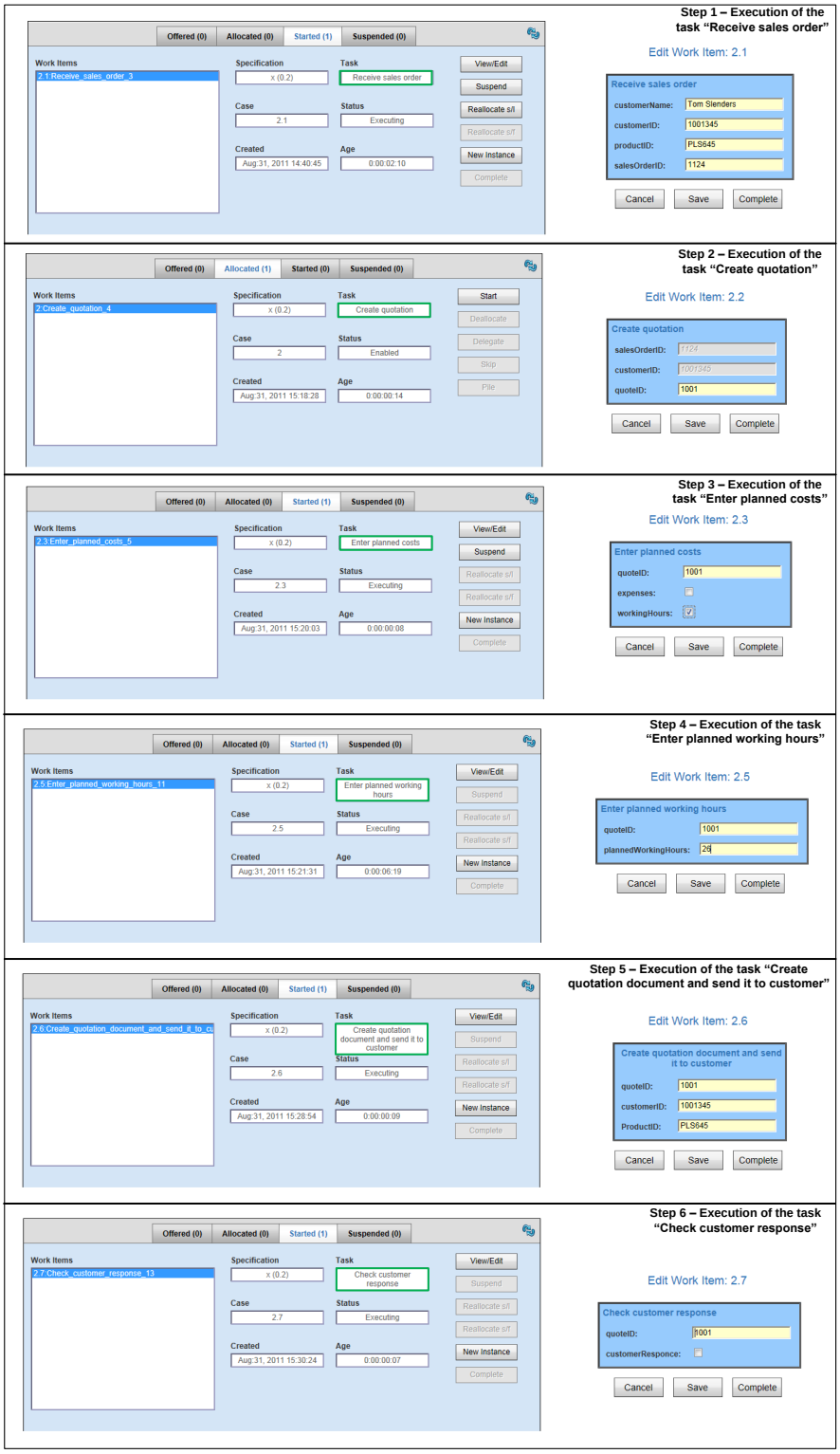

Figure 5.12: Step-by-step execution of the considered sales process scenario

5.5 Conclusions

In this chapter we have presented the process configuration architecture complementing the process configuration methodology based on context models. For providing end-to-end support for the new configuration methodology, we adapted the Synergia toolset by incorporating two additional tools, namely the open-source ontology editor Protégé and Domain Expert, a new tool developed during this research project. We have shown how the collection of tools can be used to obtain an individualized model from the configurable sales process.

By providing tool support for the proposed methodology, we could test its applicability on a small configurable model, by using the sales process example. The initial analysis shows that the method is functional, and correct individualized processes, according to our expectations when considering the configuration rules provided, are obtained. The next step is to test the applicability of the method on a bigger model, where the number of configuration rules and domain facts increases. We want therefore to evaluate the method in the context of a real-life process in order to test if the process configuration methodology can be also applicable in practice.

To test the applicability of the method on real life business processes, we developed a case study. In this case study we use the context model-driven methodology introduced in Chapter 4 and the collection of tools discussed in this chapter. In the next chapter, we will present this case study, which is based on the technical service process.

Chapter 6

Case study - Configuring the Technical Service Process

The goal of this chapter is to present a case study on the technical service process supported by the Industrial Equipment Manufacturing (IEM) solution, a To-Increase proprietary information system. For this case study a configurable process model for the technical service process has been developed. Further we applied the configuration methodology based on context to configure the process.

First, we briefly describe the service process and the supporting system. Next, we describe how the configurable process model was created and individualized, by following the phases in the *Process Configuration Life Cycle*, introduced in Chapter 3. The chapter ends with a discussion over the practical experiences and observations gained while performing the case study and conclusions.

6.1 Validation Approach

We conduct the case study detailed in this chapter having three main purposes in mind.

1. First, we are interested to see if the configuration principles can be applied to core business processes supported by ERP systems.

2. Second, we want to see if the configuration of real-life business processes is possible by applying the new context-driven process configuration methodology.

3. Last, we want to get initial feedback over the method from business consultants, in order to evaluate its applicability in practice.

In order to attain our purposes, several sessions with business consultants within To-Increase have been carried out.

Initially, we carried out two sessions with two business consultants within the company, with the purpose to document the different variations of the technical service process. Once we created the configurable process model for the technical service process, we had another session with one business consultant to validate the variants and the resulting configurable model.

For creating the context model and the configuration rules which enable the context-driven-methodology, we followed a two-step approach. We had initially a discussion with

one business consultant specialized on the service module. As a result of this discussion, we designed the configuration rules and the context model. Next, we showed the resulting rules to the same consultant and to one external reviewer (from a partner company To-Increase). The output of the second step was an adaptation of the configuration rules. In the end, the rules were compliant with the feedback from both consultants.

For getting insights into the practical applicability of the method, we sent a feedback document explaining the method and illustrating it on the technical service process to the same business consultants which provided input for designing the context model and the configuration rules. The questions asked in the feedback document can be inspected in Appendix B.

6.2 Technical Service Process

ERP solutions provide specialized software modules or components that support core business processes implemented within organizations, such as the logistics process, the marketing process, the manufacturing process. Such a generic process can be further adapted to comply with the particular needs of an organization, as the software solution is flexible and configurable. These adaptations might represent small variations from the core process or more important variations, caused by the business context in which the process is implemented. For example, a small variation could be the decision of an organization to perform a check task earlier or later in the process. On the other hand, depending whether a manufacturing organization implements the Configure-To-Order or Engineer-to-Order strategy, more important differences in the manufacturing business process emerge.

The Technical Service Process (TSP) is implemented by organizations building different sorts of equipments which provide after sales technical support to their customers in case of problems encountered with the delivered equipment. We studied this process in the context of industrial equipment manufacturing industry.

Let us first provide a short introduction to the Industrial Equipment Manufacturing (IEM) solution. This solution supports the main processes implemented whiting a machine manufacturing company, including the TSP discussed in this chapter.

6.2.1 IEM Solution

The Microsoft Dynamics Industrial Equipment Manufacturing (IEM) is an industry solution developed by To-Increase B.V. on top of the Microsoft Dynamics AX ERP solution, the ERP solution for midsize and large organizations, part of the Microsoft Dynamics family. The IEM solution is designed to support four sub-verticals within the equipment manufacturing industry, namely a) *Special and large machinery*, b) *Oil, gas and mining machinery*, c) *High-tech machinery* and d) *Commercial and Original Equipment Manufacturer (OEM) machinery and parts*.

The IEM solution supports three order fulfillment strategies, namely *Engineer-to-Order (ETO)* - strategy in which the product is designed and built according to the specifications of the customer, each product having therefore unique characteristics, *Configure-to-Order (CTO)/Build-to-Order (BTO)* - in which the product is created according to a standard design; nevertheless the specifications of the customer can influence the component production or the manufacture of the final product and *Make-to-Stock (MTS)/Build-to-Forecast (BTF)* - where the product is built according to sales estimations and sold to the customer from the

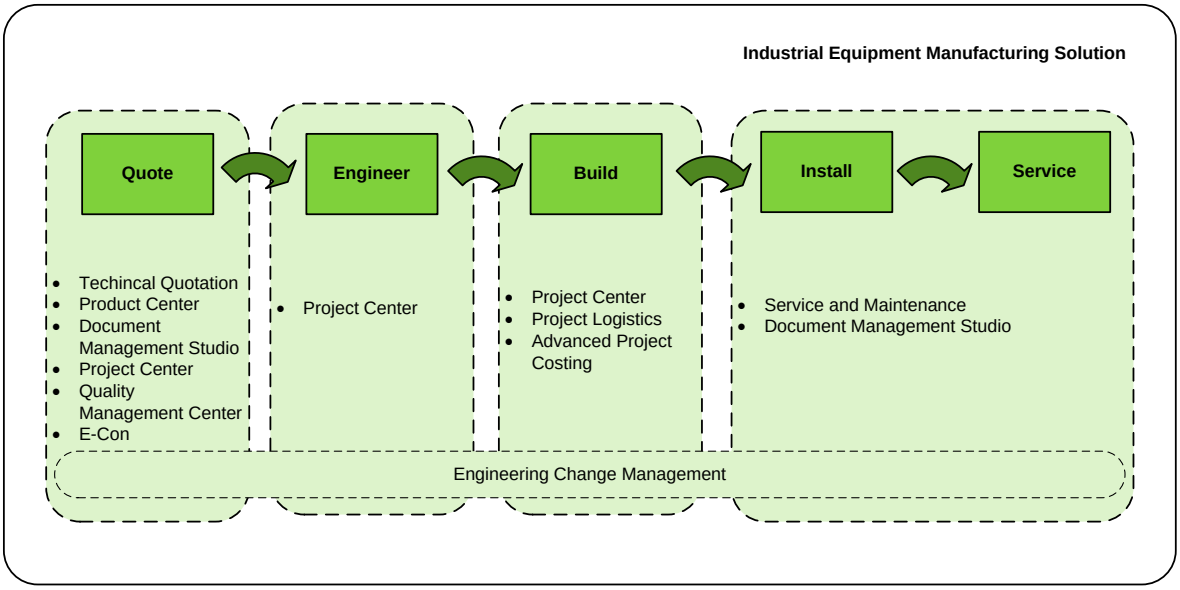

Figure 6.1: The core business processes supported by the IEM solution are *quote*, *engineer*, *build*, *install* and *technical service* (top part of the figure). Specific software components support each of these core business processes (bottom part of the figure). The description of the software components is provided in Appendix C

finished goods stock. Nevertheless, the main focus of the IEM solution is on the ETO and CTO strategies.

The IEM solution supports the main business processes that are run within a machinery manufacturing company, from the initial contact with a potential customer and until the installation of the machinery to the client, including further service jobs for maintaining the equipment in a functional state. Therefore, the main products that are commercialized are *industrial machineries*. An overview of the business processes supported by the IEM solution is depicted in Figure 6.1.

The first process is the *Quote* process in which a customer willing to buy a specific machinery contacts the vendor (manufacturing company) and asks for a detailed quote - a cost estimation of the future product. After the quotation is sent to the customer and the customer agrees with the costs of the machinery, the *Engineering* process follows where the exact design of the new product is created. Further, the machinery is built in the *Build* stage. After building the new machinery, it is delivered and installed at the client side (*Install* process). Further, based on warranty policies or on service contracts made between the manufacturer and the customer, the manufacturer answers to the customer service requests and repairs the machinery in case of problems (*Technical Service* process).

The IEM solution integrates a collection of software components which support the described business processes running within a machinery manufacturing company, as illustrated in Figure 6.1. A concise description of each of these main software components is provided in Appendix C.

The case study presented in this chapter has been build based on the TSP supported by the IEM solution.

6.3 Data Collection Phase

The first goal of the case study was to test the feasibility of using configurable process models in the context of ERP systems. A core business process supported by an ERP solution can suffer major changes when it is implemented in an organization, depending on the business context in which it is executed. Therefore, the goal was to see if the different variants that an ERP system can support for a particular process can be unified in an integrated process model.

In order to do this, we selected the TSP because it allows for a number of different variations in its implementation, providing reach material for a configurable process model, and documented its main variations.

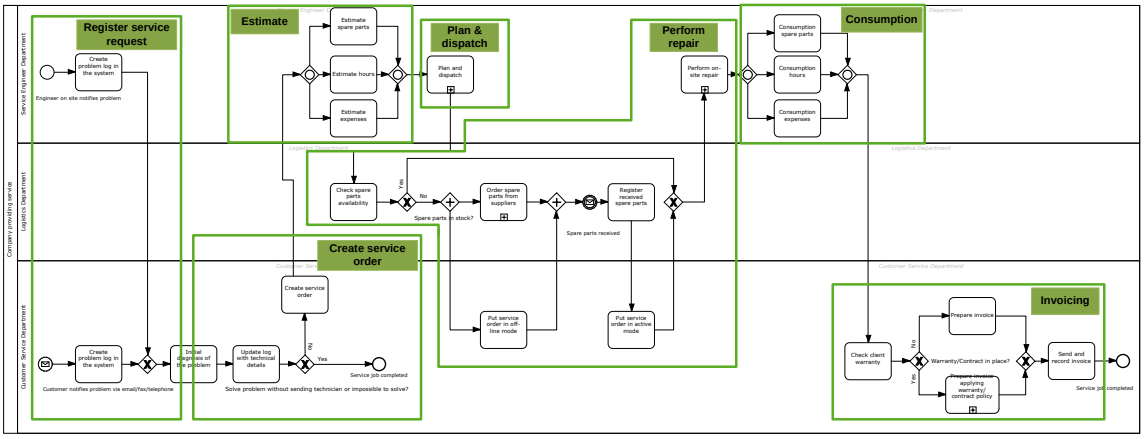

Figure 6.2: The TSP Variant 1 is the variant implemented by the majority of the organizations providing technical service (the enlarged figure can be found in Appendix D - Figure D.1)

The TSP variations were not documented within To-Increase before conducting this case study. Therefore, the first challenge was to retrieve and document as business process models the different variants for the service process. For this purpose, we have used various sources of information. We conducted two sessions with business consultants within the company, in which they explained how the service process is supported by the system. Additionally, we developed some practical experience with the IEM service module, to understand how the system can be configured to support various scenarios. Finally, we reviewed some projects that have been conducted by the company for several customers for which the service process was implemented.

This resulted in a total of seven variations, which capture the main scenarios for the implementation of the TSP. The seven process variations have been modeled using BPMN. We opted for BPMN as the business consultants in the organizations were also familiar with it. A final validation of the seven variants was realized with a business consultant specialized on the service module within IEM.

The main phases undertaken in the TSP are depicted in Figure 6.2. The steps include *registering the service request* notified by a customer, *creating a service order* in the system corresponding to the service request, *performing an estimation* of the labor hours, spare parts and expenses needed to perform the repair, *planing and dispatching* an available engineer to

go to the customer location to perform the repair, *performing the repair* at the customer location, *registering in the system the consumption* of spare parts, labor hours and expenses and finally *invoicing the customer*.

The variations identified for this process present differences mainly in terms of omitting some of the above phases or including additional phases in the process. For example, if a company performs service jobs for customers which are not covered by warranty or contract agreements, then an additional *quotation* phase needs to be included in the process in order to inform the customer about the costs of the repair. The software needs to be configured therefore to integrate quotations with the service process. Similarly, in case a company implements the TSP internally, for maintenance of its own machineries, no *registration of service requests* or *invoicing* are performed.

The seven BPMN models corresponding to the process variations, together with their description, can be inspected in Appendix D.

6.4 Design Phase

Having the process variations in place, these variants needed to be combined in an integrated configurable process model. All seven variations exhibit sufficient similarities to allow the merging. The differences between the seven processes were included into the integrated model by introducing choices between the various behaviors.

Needing to capture the integrated model into a modeling language supporting configurations, we opted for the C-YAWL environment. The translation of the control flow from BPMN to YAWL was performed manually, following the one-to-one mapping introduced in Chapter 2. An example of this translation is shown in Figure 6.3, where the mapping from the BPMN model of TSP Variant 1 to the corresponding YAWL model of the same process is depicted. The merging of the seven process variations into a configurable process model was also performed manually. The resulted configurable process model represents the integrated version of the seven TSP process variants. Figure 6.4 illustrates the mapping between the C-YAWL model and the YAWL model of the TSP Variant 1.

Figure 6.5 shows the enlarged C-YAWL model for the TSP. As it can be observed, the resulted process is rather complex, the number of tasks and arcs used increasing significantly. The model has approximately 70 tasks, out of which one third (24) are configurable.

Additional tasks, with no real meaning, have been introduced when creating the configurable model. These tasks have the purpose to capture the choices between various behaviors present in the input process variations and are transformed into silent tasks when retrieving individualized models. For example, the start of the TSP can be triggered by a number of different events (e.g. a customer notifies the service provider directly, via phone, e-mail or fax). If the service process is implemented for internal maintenance purposes, the system can be configured to automatically trigger service requests based on predefined metrics, for example the number of working hours that a certain machinery undertook. Similarly, in this case, a manual creation of a service work order can be followed when a machinery breaks. As configurable events are not supported in C-YAWL, we introduced the XOR-split and XOR-join configurable dummy tasks to model this behavior.

When inspecting the resulting model, it is clear that it is too complex to be configured by business consultants. Instead, we configure the model by using the methodology explained in Chapter 4.

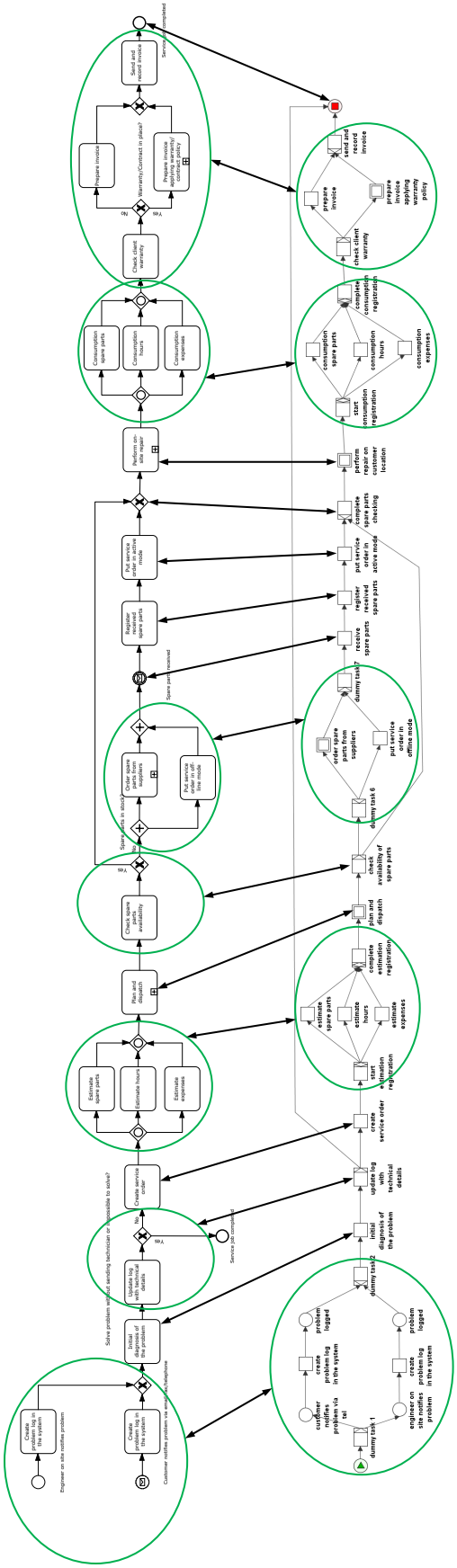

Figure 6.3: Example of mapping from the BPMN model to the YAWL model of TSP Variant 1 (the enlarged figure can be found in Appendix D - Figure D.1)

In order to apply the context driven methodology, we need first to set the scene for configuration. This requires for several steps to be carried out. First, a set of domain facts that abstract from the low level variation points present in the configurable process model need to be defined. Next, the mapping between the domain facts and the various ports in the YAWL model is created. Further, a context model able to capture the business context of manufacturing organizations must be engineered. Finally, a set of configuration rules that capture the various business scenarios which have impact over the business process and automatically set a valuation over domain facts are defined.

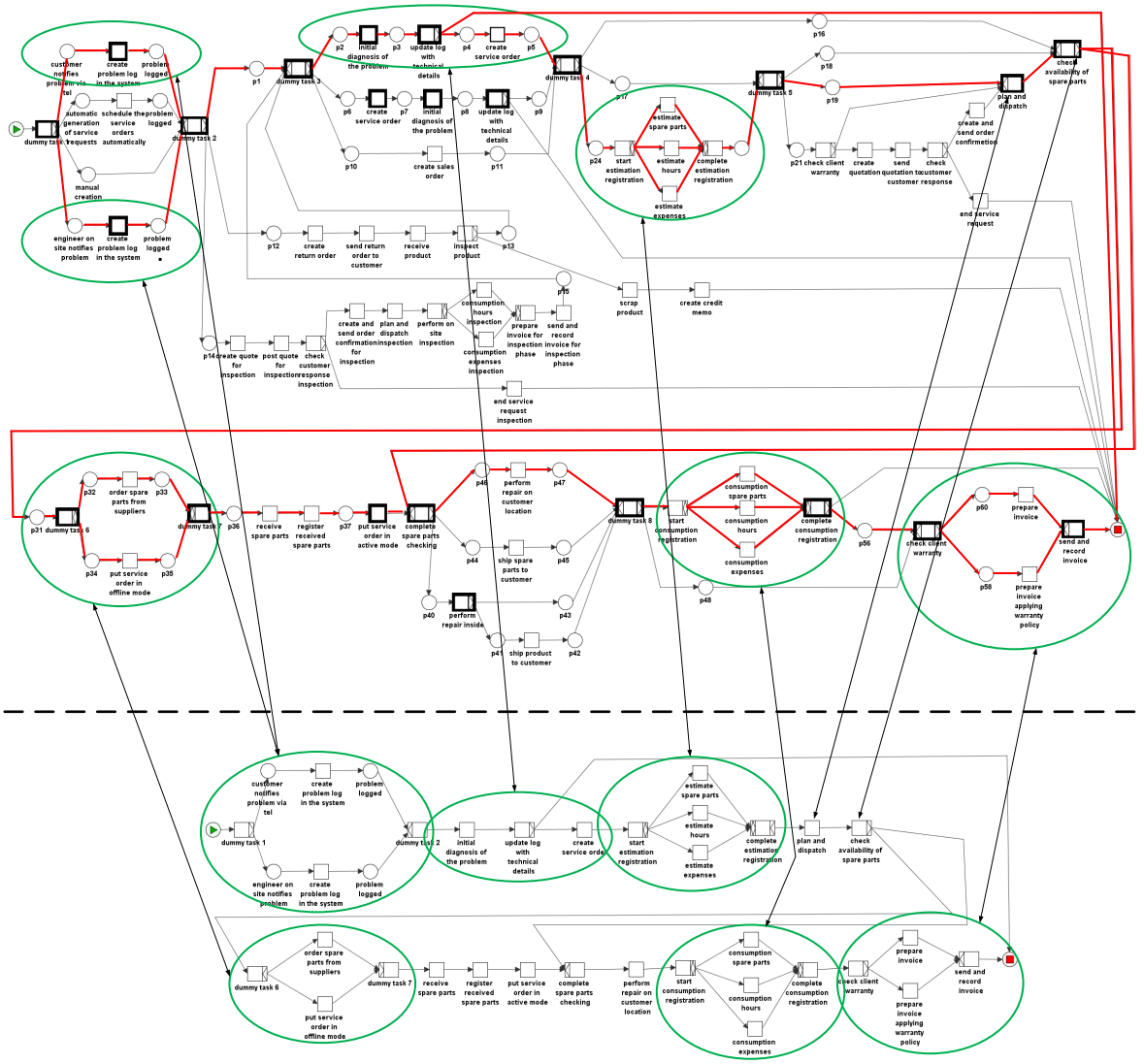

Figure 6.4: Top image - configurable TSP in C-YAWL; bottom image - YAWL model of TSP Variant 1; the purpose of this figure is to show that the configurable process model is indeed obtained as a result of the mapping between the seven TSP variations; it can be clearly seen in the figure that TSP Variant 1 can be identified in the control flow of the configurable process model by following the red marks in the C-YAWL model

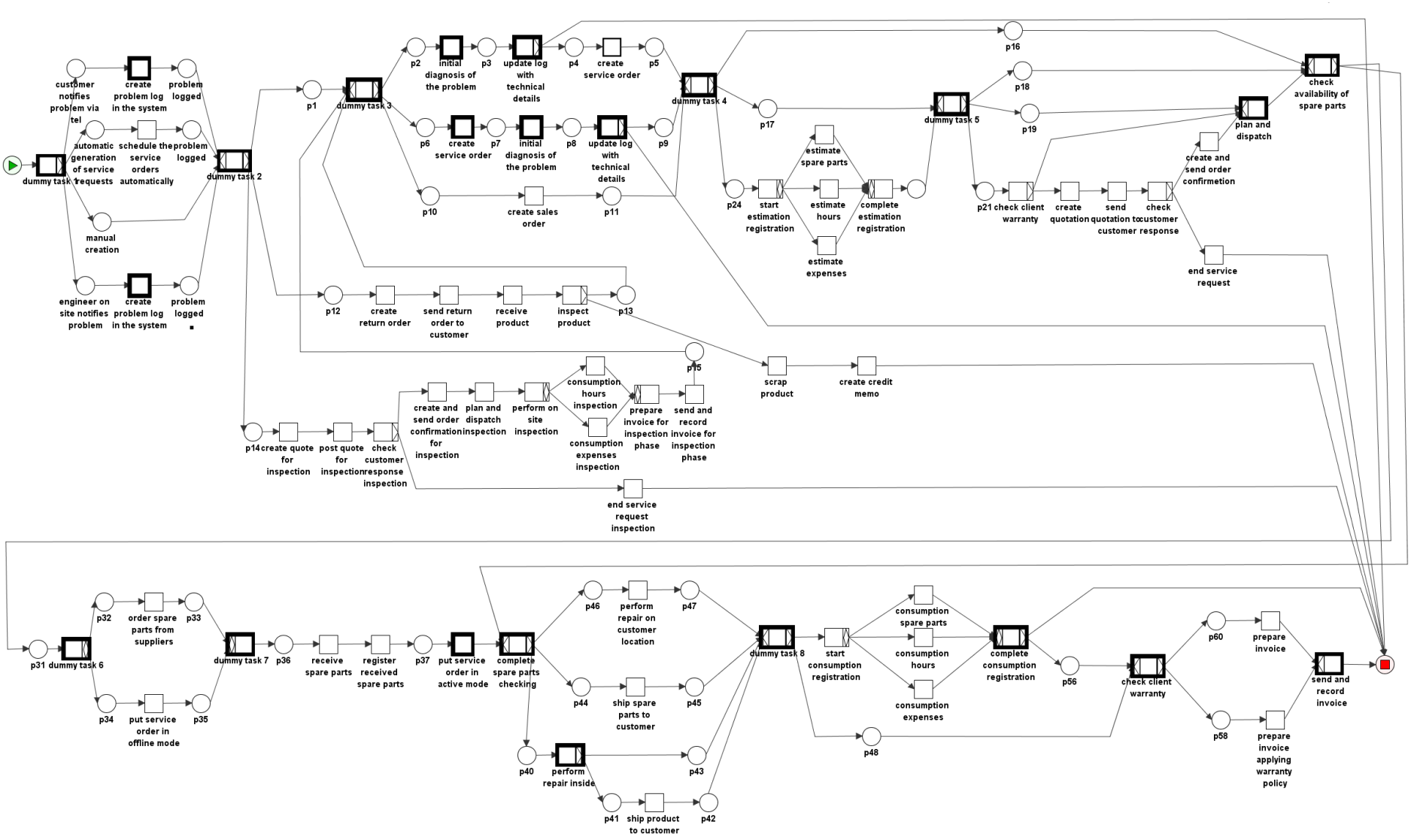

Figure 6.5: The configurable TSP in C-YAWL

6.5 Configuration Phase

The second goal of the current case study was to test whether the context driven methodology can be used to adequately configure a real life business process. Therefore, we carried out the configuration phase by applying this methodology.

The first step in the configuration phase was to define a number of domain facts that represent major configuration decisions over the technical service process. For example, such configurations include selecting between implementing the process for internal maintenance (domain fact *f3 : Internal maintenance*) or for providing after sales customer support (domain fact *f1 : Customer Service*). Further, selecting between the different variants of customer service are also captured with the help of domain facts. For example a selection can be made between implementing returns (domain fact *f2 : returns*), case in which a customer having problems with a particular equipment sends it back to the manufacturing company where the product is repaired or a compensation is payed back to the customer, or sending an engineer on the customer location for performing repairs (combination between domain facts *f1 : Customer Service* and *f6 : Engineer on site*). A total of 13 domain facts, representing high level configuration decisions over the process, have been defined. These domain facts are described in Appendix E.

The next step was to provide a correct mapping between the values of domain facts and the hiding, blocking or allowing behavior of the C-YAWL ports. For example, if a company implements returns for the service process (domain fact *f2* is set as *true*), the output port of *dummy task 2* to *p12* must be configured as allowed, as returns are processed in the system against a return order and not against a service order. The other output port of the same task needs to be configured as blocked. Some port configurations are dependent on various domain facts. For example the output port of the task *complete spare parts checking* to *p46* is configured as allowed if both domain facts *f1 : Customer Service* and *f4 : Engineer on site* are set as *true*. This is normal, as performing repairs by sending an engineer on field is a variation of performing customer service. The complete mapping between domain facts and configuration ports in C-YAWL are presented in the two tables in Appendix F.

For applying the context driven methodology, a context model able to capture the business context of manufacturing organizations was built. We had as a start the minimal context model presented in Chapter 4, which we extended to achieve the current purpose, based on the context models available in literature and presented in Appendix A. For validating the classes included in this extended model, we also had a session with a business consultant and discussed the various business scenarios in which an organization will implement a particular TSP variation. We used this information with two purposes, first to update the context model with the relevant information, and second to create the configuration rules based on this knowledge.

The classes included in the context model were structured in the three layers we proposed in Chapter 4, and are presented in Figure 6.6. The *Process Layer* contains some main classes which have the purpose to document the TSP in terms of main `activities` included in the process or the `organizational agents` that execute these activities.

For the *Organizational Layer* some elements are considered as very important for deriving configuration choices. For example, the type of `fulfillment strategy` that a manufacturing organization implements has impact over the service process. As such, an organization having a CTO strategy will most probably implement the service process for internal maintenance of the production lines it uses. We considered that an organization can implement one fulfillment

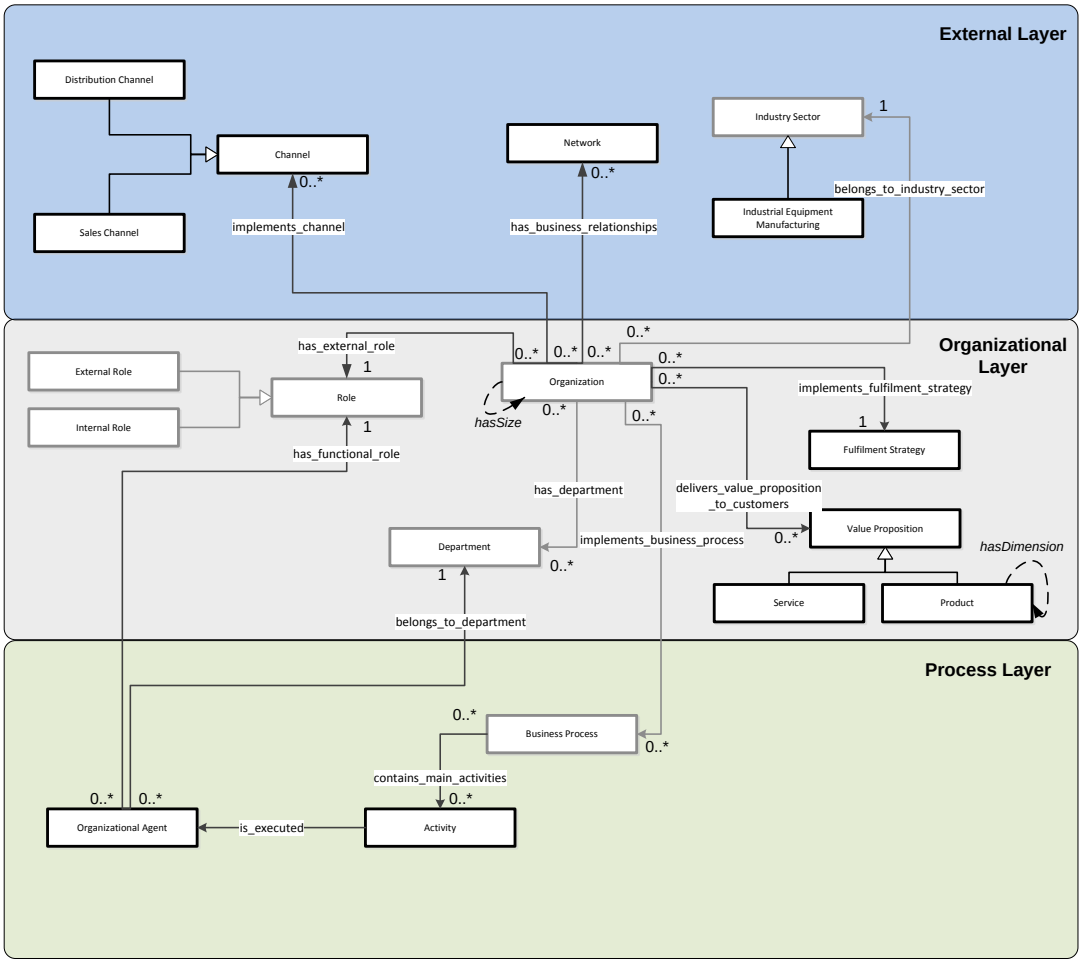

Figure 6.6: The context model used for configuring the TSP. The classes defined in the minimal context model presented in Chapter 4 are represented with grey. The classes specifically defined for the TSP process are represented with black

strategy (see the cardinality of the relation `implements_fulfillment_strategy`, Figure 6.6), therefore we have set the object property `implements_fulfillment_strategy` as functional. The type of `Product` or `Service` that an organization sells to its customers also influences the configuration of the service process. As such, a manufacturing organization building `parts` of complex equipments will most probably implement a returns strategy for its service process, whereas an organization selling `complex and large machineries` will implement a service strategy in which engineers are sent to the customer location for repairs. We grouped the `Product` and `Service` categories in the `Value Proposition` class.

The *External Layer* includes classes as the `Industry Sector`, in which we consider the four sub-verticals of the equipment manufacturing industry supported by IEM, the `Network` which encompasses the economic parties that interact with an organization (e.g. suppliers, subcontractors, representatives, etc.) and the various `Channels` through which an organization delivers its products to its customers. For example, differences in the variation of

TSP implemented exist between organizations that `directly sell` their products to the end customers and the manufacturing organizations that `sell their products indirectly`, through representatives. Therefore, inspecting the `Sales Channels` of a manufacturing organization is a factor to be considered when deciding what service strategy to implement.

Considering the knowledge captured in the context model and the main scenarios that point to a particular TSP variation, the next step was to define a set of configuration rules that generate a valuation over domain facts.

For example, a `manufacturing organization` within the `Oil, Gas and Mining Machinery` sub-vertical which implements `after-sales service` to its customers, provides service for equipments of `big` dimensions, and runs the `ETO` strategy, will most probably implement the TSP Variant 1, and not variations such as returns (TSP Variant 4) or internal maintenance (TSP Variant 5). This scenario is captured within the rule below, which sets as *true* the domain fact *f1: Customer Service*, in case the stated requirements are met.

f1: Customer Service(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $belongs\_to\_industry\_sector(?x, Oil\_Gas\_and\_Mining\_Machinery)$

$\wedge$ $has\_external\_role(?x, Manufacturer\_Role)$

$\wedge$ $implements\_business\_process(?x, after\_sales\_service\_process)$

$\wedge$ $implements\_fulfillment\_strategy(?x, Engineer\_To\_Order)$

$\wedge$ $delivers\_value\_proposition\_to\_customers(?x, ?value)$

$\wedge$ $hasDimension(?value, ?dimension)$

$\wedge$ $equal$[1]$(?dimension, "big")$

)

Dependencies between domain facts are also captured with the help of configuration rules. For example, if we consider the implementation of the service process for internal maintenance for a particular organization (domain fact *f3: Internal Maintenance* is set as *true*), then most probably the use of preventive suggestions generated automatically by the system (domain fact *f8: Preventive Suggestions*) is also used.

f8: Preventive suggestions(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $Internal\_maintenance(?x, true)$

)

A set of 21 configuration rules have been defined. The complete set of rules defined for the configuration of the TSP can be found in Appendix G.

Having the domain facts, the mapping between domain facts and process facts, the context model and the configuration rules in place, the configuration of the process can be managed by

[1] "*Equal*" is a SWRL build-in for comparison. The expressiveness of SWRL is extended by the use of several build-ins. For more information over the build-ins available in SWRL, we refer the user to [10]

business consultants by simply instantiating the context model for particular organizations. There is no need for the consultants to understand the implications of blocking or hiding certain ports in the C-YAWL configurable model.

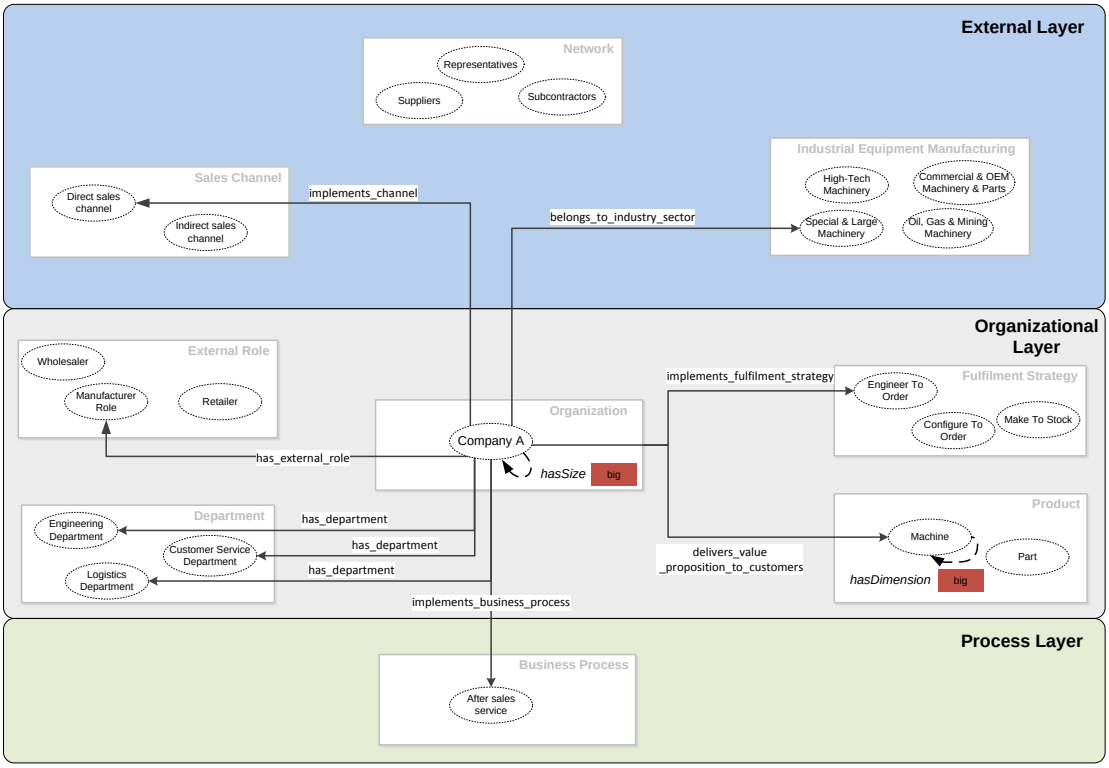

Figure 6.7: The organization model of Company A - an individualization of the context model

Having the scene for configuration set, we performed the actual configuration step for different scenarios, by instantiating the context model for several test organizations.

Let us consider a `big-sized` manufacturing company operating in the `Special and Large Machinery` sub-vertical and implementing `after sales process`. Further, this company implements as a main strategy `ETO`, and has a specialized `Customer Service department`, among other departments. The same company sells its `machineries` through a `direct sales channel`, directly to its customers. The organization model of the described company can be seen in Figure 6.7.

Typically, an organization positioned in the described business context will provide service to its customers by sending engineers on field. Therefore, a company with the characteristics described above will implement the TSP Variant 1, in which service notifications are processed in the system using service orders and engineers are planned and dispatched to the location of the customer (see Process Variant 1 - Appendix D).

The next step is to load the organization model together with the context model and the configuration rules in Domain Expert (see Figure 6.10), deriving automatically a valuation over the defined domain facts.

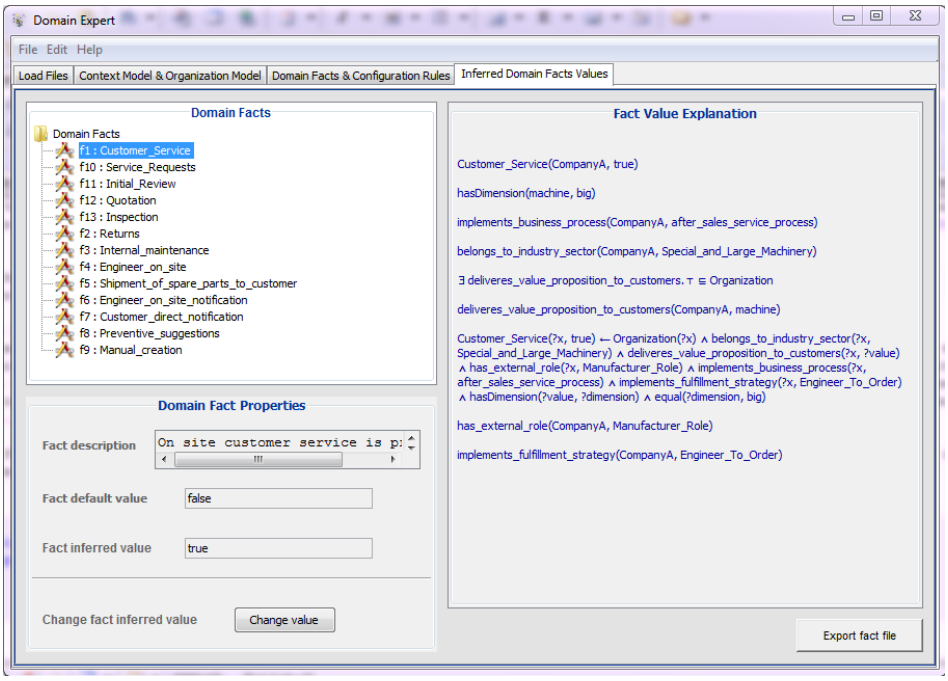

Figure 6.8: Domain Expert allows for obtaining a valuation over domain facts for a particular organization model

Further, using the Process Configurator and the Process Individualizer, the process model in Figure 6.9 is obtained from the integrated process model (Figure 6.5), by using the valuation over domain facts generated by Domain Expert. The individualized process (TSP Variant 1) is in line with our expectations when considering the organization model of Company A.

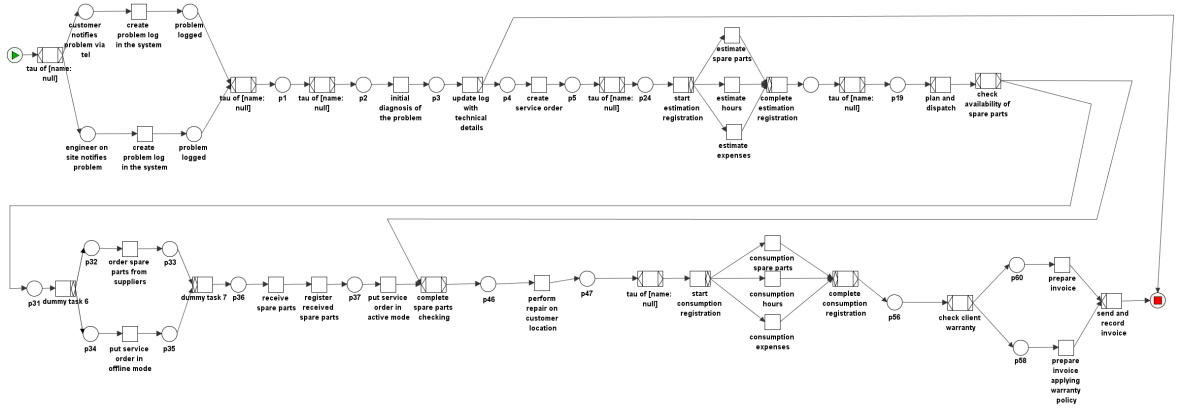

Figure 6.9: The individualized process resulting from configuring the TSP for Company A

Let us now consider a different situation. Due to some changes in the long-term business strategy, Company A decides to modify its selling scheme. As such, Company A wants to focus on manufacturing a larger variety of machineries, and decides to outsource the selling of its products. This implies that the organization will not sell its products directly to the end customers anymore. Instead, Company A will use a network of representatives to sell its machineries.

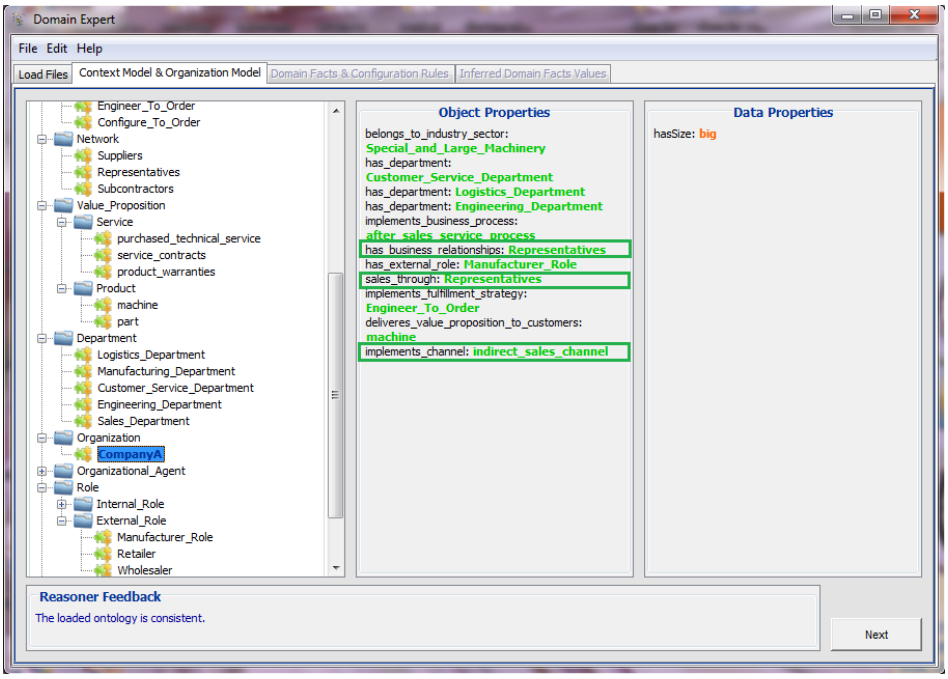

Figure 6.10: The modified organization model of Company A visualized in Domain Expert. The changes considered in the new scenario have been marked with green rectangles

Given this new scenario, Company A will need to implement a simpler variant of the service process. In this case, no engineers will be sent on field for performing repairs, but rather the representatives through which Company A sells its machineries, will provide the end customers with such service facilities. Therefore, Company A will implement a service variant in which the needed replacements (spare parts) for a repair will be shipped to the representatives, remaining for the representatives to provide on field repairs to the end customers (see Process Variant 3 - Appendix D).

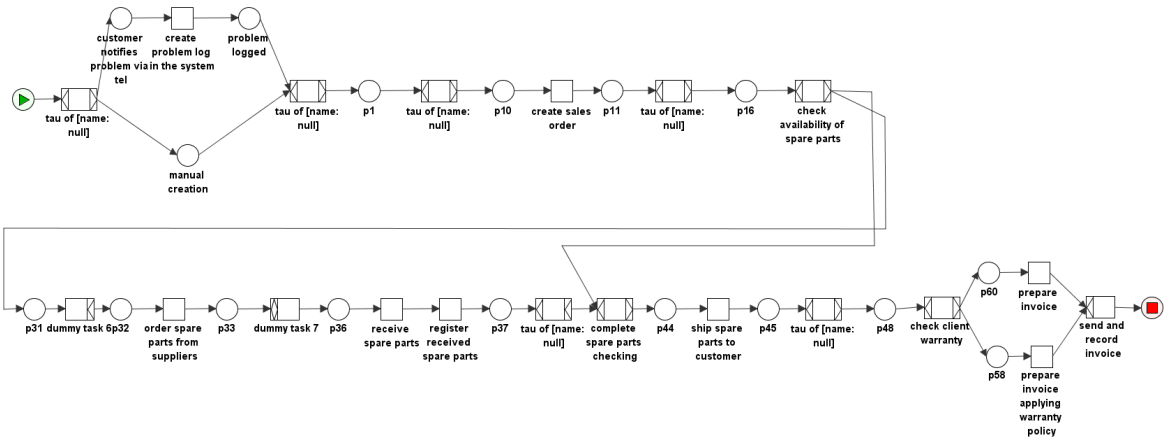

Figure 6.11: The individualized process resulting from configuring the TSP for the new business context of Company A

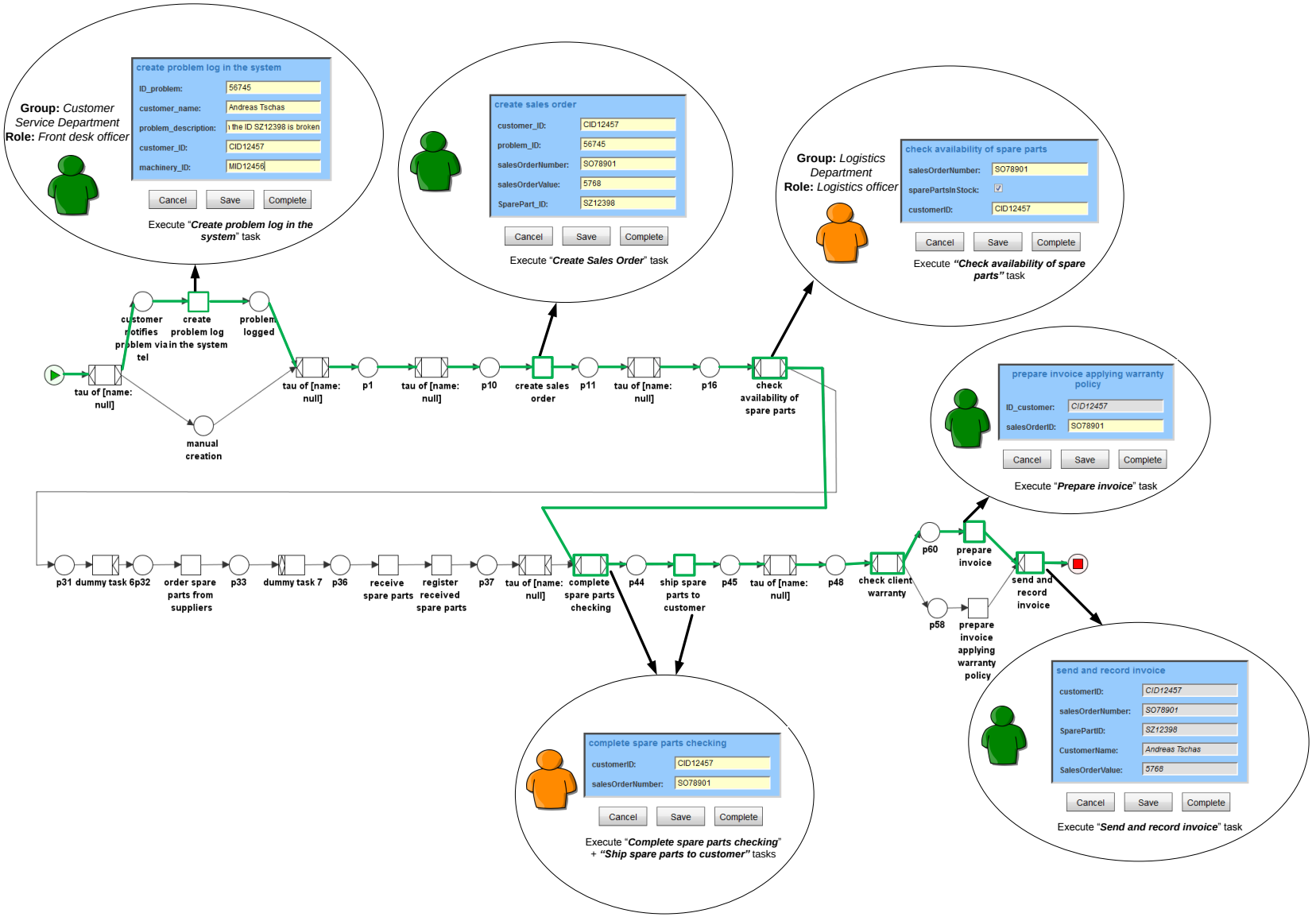

Figure 6.12: The step by step execution of the resulting YAWL specification of the service process

We load the new organization model of Company A in Domain Expert (see Figure 6.10) together with the context model and the configuration rules, and we derive a new valuation over the defined domain facts.

Using the Process Configurator and the Process Individualizer, the process model in Figure 6.11 is obtained, by using the valuation over domain facts generated by Domain Expert. The individualized process (TSP Variant 3) is again in line with our expectations when considering the new organization model of Company A.

The result of the configuration is a running system, the individualized YAWL model being directly executable in the YAWL workflow system. We have configured the system to offer and allocate the available work items to the human actors, and we have defined resources from the *Customer Service Department* and *Logistics Department* to handle the cases of the obtained service process.

Let us consider a simple scenario in which a request for a particular spare part is issued to the manufacturing company. A sales order is created in the system and the availability of the required spare part is checked against the stock inventory. As the specific spare part is in stock, it is shipped directly to the representative's location and an invoice is issued.

The step by step execution of the above case in the workflow management system is depicted in Figure 6.12.

6.6 Discussion

While performing the case study, several practical implications and observations were made. Let us summarize the most important of them in this section.

6.6.1 Observations

Most important, we conducted the case study with three purposes in mind, namely to see if the configuration principles are suitable for core business processes supported by ERP solutions, to see if a real-life business process can be configured by using the context driven methodology and to get initial feedback over the method from business consultants. By following the steps of the Process Configuration Life Cycle, *it was possible to create a configurable process model for the Technical Service Process and to configure this process by applying the context driven methodology.* This illustrates that it is possible to create configurable process models for core business processes supported by ERP systems and that the knowledge driven methodology can be used to retrieve individual variants from such integrated process models.

Creating the configurable process model was not a trivial task. Several iterations were necessary. Each change made to an individual variant of the TSP had a direct impact over the configurable process model. In the same time, making changes to the individual variants implied making changes to the mapping between domain facts and process facts as well. Such a cycle is extremely time consuming and error prone, therefore an automation of these steps is highly desirable. Tools for automatic process merging are very important in this context [32]. However, automatic generation of merged processes is an open issue, research in this area being currently performed [36].

The resulting context model turned out not to be very complex. Moreover, the relevant information that influenced the configuration decisions among the different service variants could be captured in its structure. The context model has two purposes: it is actively used

in the configuration phase and can be also used for documenting the configurable business processes within a consulting organization.

The poor expressiveness of rules creates some problems. The rules do not support negation and allow for monotonic inference. This has a direct implication over inferring the valuation of domain facts. Let us consider two configuration rules for the same domain fact, one which sets the value of the fact as being *true*, and another one which sets the value of the fact as being *false*. If both rules are fired, due to the monotonic inference property, both boolean values will be assigned to the domain fact.

Therefore, several constraints between domain facts that ensure a correct configuration of the process model can be hardly expressed using the rules.

f11: Initial Review(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge\ has\_department(?x, Customer\_Service\_Department)$

$\wedge\ hasSize(?x, ?size)$

$\wedge\ equal(?size, "big")$

$\wedge\ Customer\_Service(?x, true)$

)

Let us give a concrete example for a better illustration of such a problem. Usually a company implementing customer service and having a specialized Customer Service Department handles service requests in two steps. First it performs an initial review over the service requests with a help-desk operator in order to inspect the problem. Only if the problem cannot be solved immediately, a service order is created in the system and an engineer is planned and dispatched to perform the repair. Therefore, we created the configuration rule presented above to capture this situation.

In the same time, for capturing the business model of Company A presented earlier, we used the following rule.

f5: Shipment of spare parts to customer(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge\ has\_business\_relationships(?x, Representatives)$

$\wedge\ implements\_channel(?x, indirect\_sales\_channel)$

$\wedge\ sales\_through(?x, Representatives)$

$\wedge\ Customer\_Service(?x, true)$

)

The problem appears when a constraint between the domain fact *f11: Initial Review* and domain fact *f5: Shipment of spare parts to customer* needs to be set. These 2 facts cannot be true simultaneously as an initial review when only shipment of spare parts is provided to customer would not make sense. Adding a rule of the type $f5 \Rightarrow \overline{f11}$ is problematic. If we consider again the case of Company A previously discussed, we can see that this company meets the conditions for firing both rules presented above. This implies an initial valuation

over facts f11 and f5 as *true*. Firing also the rule $f5 \Rightarrow \overline{f11}$, will result in adding the value *false* to f11. As such, f11 will result as having both *true* and *false* values in the same time. This is due to the fact that rules have monotonic inference and cannot be used to modify existing information in the ontology.

As a generalization, adding rules that define the conditions for setting domain facts both as *true* and as *false* is hardly possible. A first attempt in solving this issue is to set each domain fact (represented as a data property in the ontology structure) as being *functional*. In this way, if two values are added to the same domain fact, the reasoner will evaluate the context model as being inconsistent. Nevertheless, this is a partial solution of the problem, as choosing between the two values of the respective domain fact needs to be done manually. A better approach is to extend the expressiveness of the rules with negation and support for both open and closed world assumption, while maintaining the modelling of knowledge with the help of ontologies. This would provide more flexibility to the reasoning process, and could better handle the situation mentioned above. Research is currently being conducted in this area, aiming to integrate these two formalisms [38].

We managed a work-around for this situation, by encoding the various constraints among domain facts when creating the mapping file, ensuring in this way a correct configuration for the process. Nevertheless, such a strategy requires a profound understanding over the configurable process model and the impact that domain facts have over the various C-YAWL ports. Therefore, only the configuration rules alone cannot ensure a correct configuration over the process, representing a major problem of the proposed methodology.

Another solution is to manually revise the inferred valuation of domain facts before using it for generating the individualized model. Nevertheless, this step requires a good understanding of the impact that domain facts have on the integrated process model. In this sense, a clear decoupling between the configurable process model and the process configuration phase is not obtained.

The last observation is that we could not define configuration rules for all the domain facts considered. The domain fact assessing whether an *inspection step* needs to be performed before sending a quotation to the customer could not be inferred based on the knowledge considered in the context model. For deciding the valuation of this domain fact, a direct question needs to be put to the customer. Therefore, combining the context driven methodology and the questionnaire approach introduced in [33] could constitute a step forward in the process configuration phase.

We defined 21 configuration rules for the case study. This list is not complete. Both the context model and the configuration rules are evolving entities, which need to be updated and extended over time.

6.6.2 Evaluation of the Approach

The third purpose we had when developing the case study was to get an initial feedback over the applicability of the new configuration method in practice.

We carried out an evaluation step in which the two business consultants which agreed to help along the project have been involved. This feedback session was performed by sending an evaluation document to the two consultants. First, the new methodology was explained in the document. Next, we illustrated how the method was applied on the TSP. Having this setup, we asked a set of questions.

Both consultants evaluated the method as having applicability in practice. When asked

which is the percentage of an individualized business process that can be covered directly by applying the configuration rules, a percentage of 75-80% was estimated as being attained directly by applying the method. Both pointed out that the remaining of 20% requires customizations that are particular for each customer situation and cannot be captured by the configuration rules. Also, the consultants judged that similarities exist between the method and their own method of providing consultancy, in which workshop sessions and questionnaires are conducted in collaboration with the customer. The main concerns included the need to test the method on other business processes and the need of providing the right tool support for the approach.

The results obtained are summarized in the table below.

Table 6.1: The main feedback comments from the business consultants answering to the feedback document

Interviewee	**Potential applications of the method (+), concerns (–)**
Internal Business Consultant (within To-Increase)	(+) The method certainly provides value. By applying the configuration rules, it can be determined what process applies to the customer. (+) What I usually do in a workshop I see back here. (+) With proper tool support, the method can be applied in practice. This will save a lot of time to get a basic setup. Then only small customizations are needed for a 100% fit for a customer. (+) The method can be applied for the configuration of other core processes supported by ERP software, besides the technical service process. (+) The method helps customers to get a clear overview of their process, which ultimately is implemented in the ERP system.
External Business Consultant (To-Increase partner company)	(+) The method is very effective in the scenario of the technical service process. (+) The method has certainly potential. The challenge is in identifying the right question set to ask to customers and to ensure that all possible configurations are captured in the configurable process model. These are both covered by the method. (–) In many cases you will be able to predict with a high percentage of accuracy the likely configuration for a particular scenario, but there will always be companies that do not fit the mould. (–) The method needs to be tested on other core business processes. The service process has a large number of variations. I cannot think of many other processes which allow for such a degree of configurability.

6.7 Conclusions

The goal of the case study presented in this chapter was to create a configurable process model for a core business process supported by an ERP solution, to configure the process using the context driven methodology and to get feedback over the applicability of the method in

practice.

We were able to attain our purposes and to create a configurable process model for the TSP and to further obtain individualized versions of the service process by creating different instantiations of the context model. We also shared our observations and practical implication resulted from performing the case study. We pointed out that the lack of expressiveness of rules is problematic and does not always ensure a correct configuration of the process. To solve this issue the valuation of domain facts needs to be manually reviewed by domain experts, in this sense preventing a complete decoupling between the configuration phase and the process model.

The method was evaluated as having applicability in practice by the business consultants we questioned. Both pointed out that manual adaptations to the individualized process need also to be performed in order to obtain a 100% fit of the process for a particular customer. The main points of concern regard the need to further test the method on other core business processes and to be able to provide proper tool support for the method.

The case study presented in this chapter answered to the last research question stated in the introduction of this thesis, namely *Is the novel configuration methodology feasible in practice?*

Next we will present the final conclusions of this thesis together with main directions for future work.

Chapter 7

Conclusions

This chapter summarizes the work undertaken during this master thesis. We also discuss the contributions that the method brings to the business process configuration domain. The chapter ends with some suggestions for future work.

7.1 Summary

In this thesis we propose a new process configuration methodology, which allows non-technical users (business consultants, domain experts) to perform the configuration of reference process models. We started by analyzing the questionnaire based methodology already available in literature. As a result, we concluded that, although the process configuration phase based on questionnaire models abstracts from the low level variation points of the process, and guides the user in making correct configurations, the questions are still driven from the process side, and do not encompass real business knowledge over the company for which the process is being individualized.

For filling this gap, we proposed to configure processes based on the business context in which an organization operates. We modeled the business context using ontologies.

In order to use context models in process configuration, we propose a three steps approach. First, the context model is captured in an ontology model. Next, an organization model is obtained by instantiating the context model for a particular company. Last, the domain facts specific to a business process are configured by using a set of configuration rules which are applied over the knowledge comprised in the organization model. By doing so, the configuration of the business process is directly derived from the specific business context of an organization.

We developed tooling to support the proposed methodology and provided an initial validation of the method on a real-life business process.

The main contributions of this thesis can be summarized as follows:

- The use of context models for business process configuration, and modelling context with ontologies.
- The development of a methodology for process configuration based on context models.
- The realization of the method by extending the Synergia toolset.

Considering these, we can conclude that the initially defined goal of the project (*Develop a new methodology and toolset for the configuration of process models, taking into account the business context that determines the configuration decisions over processes.*) was fulfilled.

7.2 Discussion

The method introduced in this thesis enables the configuration of business processes by reasoning over the business context of an organization. This makes the method accessible to non-technical people.

Our initial validation results have showed that the method is functional and has practical applicability. The business consultants questioned assessed the method as an useful tool in ERP project implementations. Moreover, the method seemed familiar to the business consultants, proving that it succeeds in abstracting from the technical variation points present in the configurable process model.

The result of the configuration phase presented in this thesis is a running system, as the individualized YAWL model can be directly deployed in the YAWL workflow management system. In the context of ERP systems, obtaining directly from the configuration phase a running system is not a trivial task. Nevertheless, the resulting process model can be used to guide the configuration of the ERP system, as the ERP system needs to be configured to support the behavior captured in the process. This is possible especially if the ERP system has internally the notion of a workflow used for guiding the sequence of tasks to be executed.

In this sense the method is able to advocate the use of configurable process models into practice. Moreover, the method maps nicely with the ERP business processes in the context of which was tested, as the core business processes supported by ERP systems allow for variations directly driven by the business context in which the business process is run.

Nevertheless, a better support for configuration rules and proper tooling along the process configuration life-cycle are main concerns in relation to this approach.

7.3 Future Work

We consider a number of directions for future work, which consider both the configuration phase as well as the entire process configuration life-cycle discussed in this thesis.

Context and Organization Model

In this thesis we propose to automatically infer the value of the domain facts specific for a configurable process model by reasoning over the organization model, which is an instantiation of the context model.

Therefore, the first step consists in engineering a context model, that can be used for the configuration of many organizations. The process of modelling a good and comprehensive context model is therefore of interest. A proper method to assess the validity of an engineered context model is through empirical studies.

Secondly, the generation of the organization model can be further improved. The organization model encompasses the business knowledge specific to a particular company (e.g. the industry sector of the company, the various organization units within the company, the business processes implemented by the company, the suppliers, etc.). In order to generate the

organization model, we proposed to instantiate the context model (which aims to comprise business knowledge at a general level) for a particular company. Currently, this instantiation step is performed using an unstructured set of questions that aim to particularize the context model for a given company. Nevertheless, as the context model becomes more comprehensive and gathers a large amount of information, we expect the instantiation phase to become more difficult, time consuming and error prone.

More research into the actual steps that need to be taken in order to obtain the organization model, given the context model, is further required. A possible direction could be to provide a questionnaire model on top of the context model, which aims to put specific questions, following a particular order to a customer, in order to derive the organization model. Such an approach would provide structure to the individualization phase, which is currently rather ad-hoc. Moreover, the valuation of domain facts which cannot be covered by defining appropriate configuration rules can be also performed by asking direct questions to customers. The work on questionnaire models performed in [33] can serve as a basis for this approach.

Rule Management and Maintainability

We proposed in Chapter 4 the use of configuration rules over an organization model in order to automatically derive a valuation over domain facts.

As the configurable models become more complex and increase in size, the number of configuration decisions that need to be taken increases also. As a direct consequence, the number of configuration rules that set the domain facts over such processes grow in number and complexity. Moreover, over time the set of configuration rules requires adaptations and extensions. A specialized software system, in charge with the definition and maintainability of configuration rules can provide help to process modelers in defining and further deploying and managing the set of configuration rules for a business process. Systems with similar applicability, such as business rules management systems, are already available in practice [31].

Configurable Modelling Languages

As a configurable modelling language for capturing variations in business processes, we used C-YAWL. By using C-YAWL as a modelling environment, the direct result of the configuration phase is a running system, which can be directly deployed in organizations.

In the same time, a process modeling language common in industry is the Business Process Modeling Notation (BPMN) [23]. BPMN provides a series of advantages for business analysts and consultants, including a process flow modelling technique that is more in line with the requirements of the business, allowing for the modelling of inter and intra - organizational processes and providing a high degree of flexibility to the modeller (e.g. use of multiple start/end events, various types of events with direct application in the business) [21].

BPMN does not support configurability. In [47], the idea of capturing variations in process models by using annotations has also been applied to BPMN tasks. Nevertheless, this approach has not been further extended and presents several flaws [33]. No other noticeable attempt to extend BPMN with a configurability dimension is present in literature.

There are two main problems that are discouraging the extension of BPMN with a configuration dimension. First, the complexity of the BPMN language (the high number of concepts that are defined in the standard) is a problem, adding a configuration dimension to all the concepts that BPMN supports being unfeasible. Second, the tools supporting BPMN are only focusing in supporting the modelling phase, and do not provide an executable environment

for the resulting BPMN models. Nevertheless, the main purpose of using configurable process models is to obtain suitable information systems that can further support the business processes of organizations.

In the context of ERP systems, where the systems are complex and do not necessarily have the notion of a workflow for supporting business processes, obtaining directly a running system from the configuration phase is difficult. Nevertheless, an individualized process model that captures the business process that the system should support can help consultants in configuring the ERP system. First, consultants can generate individualized models from the configurable process model. Further, these models can guide the configuration of the ERP system, customizing the system to allow for the functionality captured in the process. The result is again an information system that can be deployed to the customer. Therefore, we believe that a configurability layer added to the main elements of BPMN can help further in advocating the used of configurable process models in practice and ultimately in generating a suitable information system.

Validation

Further validation of the introduced methodology in industry is required in order to asses its practical applicability and to provide improvements. As both configurable process models and context models are changing entities, which need to be adapted over time, we consider as a proper validation not only to test the method on complex real-life configurable processes, but also to observe the applicability of the approach over time. This will allow us to reason over facts as the effect that changes of configurable process models or context model have over the configuration rules or the level of complexity that a context model should have to be used in practice.

Bibliography

[1] W.M.P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[2] W.M.P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, and M. Jansen-Vullers. Configurable Process Models as a Basis for Reference Modeling. In Christoph Bussler and Armin Haller, editors, *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 512–518. Springer Berlin / Heidelberg, 2006.

[3] W.M.P. van der Aalst and K. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, USA, 2004.

[4] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245 – 275, 2005.

[5] P. Avgeriou and U. Zdun. Architectural Patterns Revisited - a Pattern Language. In *Proceedings of the 10th European Conference on Pattern Languages of Programs (EuroPLoP 2005)*, pages 431–470, 2005.

[6] P. Bertolazzi, C. Krusich, and M. Missikoff. *An Approach to the Definition of a Core Enterprise Ontology: CEO*, pages 14–25. 2001.

[7] SUPER Project Consortium. SUPER - Semantics Utilised for Process Management within and between Enterprises. `http://www.ip-super.org/`, 2006. [accessed 15.06.2011].

[8] World Wide Web Consortium. OWL 2 Web Ontology Language Document Overview. `http://www.w3.org/TR/owl2-overview/`, 2009. [accessed 15.06.2011].

[9] World Wide Web Consortium. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. `http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/`, 2009. [accessed 15.06.2011].

[10] World Wide Web Consortium. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. `http://www.w3.org/Submission/SWRL/`, 2011. [accessed 10.05.2011].

[11] Supply Chain Council. Supply Chain Operations Reference-Model. `http://supply-chain.org/scor`, 2011. [accessed 1.05.2011].

[12] T. Curran, G. Keller, and A. Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Prentice-Hall, Inc., 1998.

[13] T.H. Davenport. Putting the Enterprise into the Enterprise System. *Harvard Business Review*, 76:121–131, July 1998.

[14] B. F. van Dongen, M. H. Jansen-Vullers, H. M. W. Verbeek, and W. M. P. van der Aalst. Verification of the SAP Reference Models Using EPC Reduction, State-Space Analysis, and Invariants. *Comput. Ind.*, 58:578–601, 2007.

[15] A. Filipowska, M. Kaczmarek, M. Kowalkiewicz, I. Markovic, and X. Zhou. Organizational ontologies to support semantic business process management. In *Proceedings of the 4th International Workshop on Semantic Business Process Management*, SBPM '09, pages 35–42, New York, NY, USA, 2009. ACM.

[16] M.S. Fox, M. Barbuceanu, and M. Gruninger. An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour. *Computers in Industry*, 29(1-2):123 – 134, 1996. WET ICE '95.

[17] M.S. Fox, J.F. Chionglo, and F.G. Fadel. A Common-Sense Model of the Enterprise. In *Proceedings of the Industrial Engineering Research Conference*, pages 425–429, 1993.

[18] F. Gottschalk. *Configurable Process Models*. PhD thesis, Eindhoven University of Technology, 2009.

[19] F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *International Journal of Cooperative Information Systems (IJCIS)*, 17:177–221, 2008.

[20] AIS Group. Process Mining. `http://www.processmining.org/`, 2011. [accessed 07.06.2011].

[21] Object Management Group. BPMN and Business Process Management. `http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf`, 2009. [accessed 10.06.2011].

[22] Object Management Group. Ontology Definition Metamodel. Technical report, Object Management Group, 2009.

[23] Object Management Group. Business Process Model and Notation (BPMN), Version 2.0. `http://www.omg.org/spec/BPMN/2.0`, 2011. [accessed 18.05.2011].

[24] C. Hestermann, R.P. Anderson, and C. Pang. Magic Quadrant for Midmarket and Tier 2-Oriented ERP for Product-Centric Companies. Gartner, Inc., 2009.

[25] A.H.M. Hofstede, W.M.P. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation: YAWL and its Support Environment*. Springer, 1st edition, 2009.

[26] M. Horridge. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools. Technical report, The University Of Manchester, 2011.

[27] I. Horrocks and P.F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 723–731. ACM, 2004.

[28] D.E. Jenz. Ontology-Based Business Process Management. Technical report, Jenz & Partner, 2003.

[29] H. Klaus, M. Rosemann, and G.G. Gable. What is ERP? *Information Systems Frontiers*, 2:141–162, 2000.

[30] R.K.L. Ko. A computer scientist's introductory guide to business process management (BPM). *Crossroads*, 15:4:11–4:18, June 2009.

[31] M. Koshkina, K. Huynh, and Y. Zhao. Achieving Business Agility with WebSphere ILOG JRules and WebSphere BPM. In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '10, pages 360–362, New York, NY, USA, 2010. ACM.

[32] M. La Rosa. Process Configuration.com. `http://www.processconfiguration.com`, 2008. [accessed 10.02.2011].

[33] M. La Rosa. *Managing Variability in Process-Aware Information Systems.* PhD thesis, Queensland University of Technology, 2009.

[34] M. La Rosa, F. Gottschalk, M. Dumas, and W.M.P. van der Aalst. Linking domain models and process models for reference model configuration. In *Proceedings of the 2007 international conference on Business process management*, BPM'07, pages 417–430, Berlin, Heidelberg, 2008. Springer-Verlag.

[35] Clark & Parsia LLC. Pellet Reasoner. `http://clarkparsia.com/pellet/`, 2011. [accessed 10.05.2011].

[36] B. Luka. Model Merging in the Context of Configurable Process Models, 2011.

[37] J. Mendling, M. Moser, G. Neumann, H. Verbeek, B. van Dongen, and W. van der Aalst. Faulty EPCs in the SAP Reference Model. pages 451–457. 2006.

[38] B. Motik and R. Rosati. Reconciling Description Logics and Rules. *Journal ACM*, 57:30:1–30:62, June 2008.

[39] University of Manchester. OWL API. `http://owlapi.sourceforge.net/index.html`, 2011. [accessed 20.04.2011].

[40] A. Osterwalder. *The Business Model Ontology - A Proposition in a Design Science Approach.* PhD thesis, University of Lausanne, Switzerland, 2004.

[41] J. Recker, M. Rosemann, W.M.P. van der Aalst, and J. Mendling. On the syntax of reference model configuration: Transforming the C-EPC into lawful EPC models. In Ekkart Kindler and Markus Nuttgens, editors, *First International Workshop on Business Process Reference Models (BPRM'05)*, pages 60–75, Nancy, France, 2005. Springer.

[42] I. Reinhartz-Berger, P. Soffer, and A. Sturm. Extending the adaptability of reference models. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions*, 40:1045–1056, September 2010.

[43] C. Rolland and N. Prakash. Bridging the Gap Between Organisational Needs and ERP Functionality. *Requirements Engineering*, 5:180–193, 2000. 10.1007/PL00010350.

[44] M. Rosemann and J. Recker. Context-aware Process Design: Exploring the Extrinsic Drivers for Process Flexibility. In T. Latour and M. Petit, editors, *18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium.*, pages 149–158, Luxembourg, 2006. Namur University Press.

[45] M. Rosemann, J.C. Recker, and C. Flender. Contextualisation of Business Processes. *International Journal of Business Process Integration and Management*, 3:47–60, 2008.

[46] M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32:1–23, March 2007.

[47] A. Schnieders. Variability mechanism centric process family architectures. In *Engineering of Computer Based Systems, 2006. ECBS 2006. 13th Annual IEEE International Symposium and Workshop on*, pages 10 pp. –298, March 2006.

[48] Stanford University. Protégé. `http://protege.stanford.edu/`, 2011. [accessed 15.06.2011].

[49] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The Enterprise Ontology. *The Knowledge Engineering Review*, 13:31–89, 1998.

[50] D. Yang, R. Miao, H. Wu, and Y. Zhou. Product Configuration Knowledge Modeling Using Ontology Web Language. *Expert Syst. Appl.*, 36:4399–4411, 2009.

List of Figures

List of Tables

Appendix A

Context Ontology - Glossary of Terms

The appendix contains the main categories describing business context that we identified in the context models reviewed from literature. We included in our selection those categories (classes) that have a correspondent into one or several of the major business ontologies existent in research projects. We further mapped each of the categories to the layers proposed in the onion model introduced in [45].

The literature sources reviewed are:

1. The TOronto Virtual Enterprise (TOVE) enterprise modelling project [16, 17]
2. The Edinburgh Enterprise Ontology (EO) project [49]
3. The Core Enterprise Ontology (CEO) project [6]
4. The Business Management Ontology project [28]
5. The Semantic Utilities for Process management within and between EnteRprises (SUPER) project [15]
6. The Business Model Ontology (BMO) [40]

External + Environmental	TOVE Project	EO Project	CEO Project	Business Management Ontology	SUPER Project	BMO
Industry Sector		Market	Branch			
Channels						Distribution Channels
Network		Stakeholder Customer/ Vendor				Target Customer
Relationships						Relationship

Figure A.1: External Layer Compared to Literature Review

Internal Context	TOVE Project	EO Project	CEO Project	Business Management Ontology	SUPER Project	BMO
Organization	Organization	Organization	Enterprise	Organization	Organization	
Organizational Unit	Division / Sub-division	Organizational Unit	Business Unit	Organization Unit	Organizational Unit	
Role			Trader / Financial/ Manufacturer/ etc.	Role	Role	
Goal	Organization goal/ Sub-goal	Objective/ Purpose	Goal	Business Goal/ Business Objective	Goal	
Key Performance Indicator (KPI)					KPI/metric	
Product	Product	Product	Product			Value Proposition

Figure A.2: Enterprise Layer Compared to Literature Review

Internal Context	TOVE Project	EO Project	CEO Project	Business Management Ontology	SUPER Project	BMO
Business Area				Business Area	Business Function	
Business Process	Process	Process		Business Process	Business Process	
Activity	Activity	Activity		Task	Activity/Task	Activity
Organizational Agent	Organizational Agent	Person/ Activity Owner		Resource	Resource/Person	Agent
Event			Event	Action event	Event	
Resources	Resource	Resource	Passive Entities	Resource	Resource	

Figure A.3: Process Layer Compared to Literature Review

Appendix B

Validation Document Content

For getting insight into the applicability of the proposed method in practice, we sent a validation document to the business consultants which helped us in engineering and configuring the technical service process along the case study. The document consisted of two parts. First, we asked questions with the purpose of validating the results obtained for the configuration of the service process. Next, we asked questions related to the methodology as a whole.

1. Validation of TSP Configuration
    - Do you find the configuration rules described above in accordance to the reality?
    - What percentage of a process implemented for a particular company can be covered directly by applying the configuration rules explained above and obtaining an individualized process (90%, 80%, 70%, below)?
    - Do you consider the business aspects included in the context model sufficient for describing the business context and the configuration rules?
    - If you answered No at the question above, what other important aspects would you include and how would these aspects influence the configuration of the service process?
2. Validation of the Context-Driven Methodology
    - Give your general opinion about the method, saying if you consider the method useful.
    - Considering that the proper tool support for such a method is in place, do you see the method applicable in practice?
    - Do you see this method applicable to other core business processes (except the technical service process explained in this document)?
    - If you have any concerns about the method described in this document, please write them here.

Appendix C

Main Software Components of IEM System

In this appendix, the main IEM software components which support the business processes running within a machinery manufacturing company are detailed in Table C.1.

Table C.1: IEM Main Software Components Supporting Manufacturing Business Processes

Software Component Name	**Software Component Description**
Technical Quotation	The Technical Quotation component allows the calculation of the cost estimations corresponding to the construction of a new machinery, on a project based manner, taking into account elements as items needed, number of working hours or fixed costs.
Product Center	The Product Center supports the entire product life-cycle by allowing users to find, organize, and communicate all product related information needed for manufacturing, engineering, product development or sales departments.
Project Center	The Project Center component provides an integrated project content management environment, including products, documents, files, tasks, milestones, people, hours, etc. In this way, users have access to all project-related information from a unique form.
Document Management Studio	The Document Management Studio enables users to easily create and maintain simple and advanced external documents.
Quality Management Center	The Quality Control component implements a complete testing environment within Microsoft Dynamics AX, supporting the testing of products/services (coming from production or from suppliers) in a predetermined manner to ensure a certain quality level.
e-Con	Especially in case of BTO strategies, the e-Con is a useful component that allows companies in manufacturing, distribution or service industry to configure products, services, quotations, and prices.

Table C.1 – continued from previous page

Software Component Name	**Software Component Description**
Project Logistics	The Project Logistics component integrates all related logistical transactions (production orders, purchase orders) belonging to a certain project into one view.
Advanced Project Costing	The difference between forecasted and real costs for a certain project can be monitored by using the Advanced Project Costing component.
Service and Maintenance	The activities taking place after a machine is installed to the customer can be managed with the help of the Service and Maintenance component. The service direction handles the activities performed by a vendor in order to ensure the functionality of the products installed at the client side, including the implementation of warranty obligations, the dispatch of available resources per service jobs, follow-up jobs, etc. The maintenance direction is intended to support the internal maintenance activities of the vendor proprietary machines.
Engineering Change Management	The Engineering Change Management component can be used in all the steps of the business process presented above and deals with product changes (both in terms of product specification and product documentation) triggered by various factors such as market circumstances, technical improvements, customer demands or quality control practices.

Appendix D

TSP Variants

This appendix contains the seven variations for the technical service process discussed in Chapter 6, as summarized in Table D.1. The process variants have been modeled using the BPMN notation. These models describe the main service patterns used by service provider companies within the industrial equipment manufacturing industry sector and supported by the IEM system. The returns process (Process Variation 4) is currently not supported in the IEM system. We have chosen to include this variation as it is a common service process implemented by service providers belonging to this industry sector and will also become available in a future release of the IEM system.

The configurable technical service process is the result of the manual merging of these seven process variations and has been manually converted into a C-YAWL model.

The first variant of the service process (Process Variant 1) represents the core service process, as implemented by the majority of companies providing service. Therefore, when introducing the other variants, we will refer to this process for underlying the differences between the several variants.

Table D.1: Technical Service Process - Process Family

Process Variant	**Process Description**
Process Variant 1	The first variant describes the main steps undertaken when providing service. Initially, a customer notifies the problem to the service provider and a service request is logged in the system. The customer can notify the problem directly, via phone, email or fax, or an engineer located at the customer side can notify the problem while performing a different work order for the same machine. Further a first review of the service request is performed by an operator within the customer service department. The service request can be immediately solved or can be rejected by the service provider, case in which the service job is released and the process ends. If the problem needs further investigation, a service order is created in the system.

Table D.1 – continued from previous page

Process Variant	Process Description
	The next step is to create an estimation of the costs involved in the repair process. The necessary spare parts, the number of working hours spent by a service engineer to do the repair or the other expenses involved are estimated. Further, a service engineer is planned and dispatched for handeling this service order. Planning is mainly made by taking into consideration the availability of engineers. If spare parts need to be used for the repair, their availability in the stock needs to be checked. In case spare parts are not available in stock, the service order is put in off-line mode while spare parts are ordered and received from suppliers. Once the spare parts become available, the assigned engineer handles the repair at the customer location. The consumption of spare parts, working hours or expenses is inserted in the system after the repair is completed. The last step is to invoice the customer according to the contract policies, if a contract or warranty beween the customer and the service provider exists, or fully otherwise.
Process Variant 2	In this variant, sending a price quotation to the customer, prior to planning and dispatching an engineer for the service order, is considered. A quotation document is generated based on the estimated expenses inserted in the system. The customer can accept the quote, case in which the process goes further (in the same way as described for Process Variant 1), or can reject it, case in which the process terminates. A check of the warranty or contractual agreements with the respective customer is performed prior to generating and sending the quotation document. The reason is that the quotation is usually send to customers for which warranty and contractual policies do not apply and full invoicing needs to be performed.

Table D.1 – continued from previous page

Process Variant	Process Description
Process Variant 3	This process considers the case in which there is no need to send a technician to the customer side for repairing a machine, but rather the required spare parts are sent directly to the customer location. The customer repairs the machine and installs the replacements himself. The process starts with a notification from the customer which requires a list of spare parts to be sent. In this case, the service notification is processes using a sales order. The availability of requested spare parts is checked, and in case no sufficient materials are available in stock, these are ordered from suppliers. Once the requested spare parts are available, these are shipped to the customer location. The invoice is handled depending on the warranty or contractual policies in use for the current customer. No labor costs are involved in this situation.
Process Variant 4	The return process is a variant of the service process in which a customer can return a broken product to the vendor for repairs or crediting. After the customer notification, a return order is created in the system and a return order document with a list of the products to be returned is sent to the customer. Once the product(s) is(are) received by the service provider, a technical inspection is performed. The technical inspection can indicate that the product is damaged and needs to be scrapped. In this case the customer is credited. The technical inspection can also show that the product can be repaired, case in which a service order is created in the system and the process continues in a similar way as described in the previous variants. After repairing the product, the service provider ships it back to the customer. The final step is to invoice the customer in accordance with the warranty policies applicable.
Process Variant 5	Prior to sending a quote for repair, a company may opt to implement also an inspection phase. This implies that an engineer goes to the customer location for a technical inspection of the machine. Further, based on the inspection's result, the quotation document is generated and sent to the customer and the service process continues as described in Process Variant 1. This process variant also includes sending to the customer a quote for the inspection phase, prior to performing the inspection.

Table D.1 – continued from previous page

Process Variant	**Process Description**
Process Variant 6	Additionally to implementing the service process for providing external maintenance of machines for customers, a company can opt to implement the process for internal maintenance of his own machines also. In this case, a simplified version of the service process described in variant 1 is considered. First a service order is created, either based on an automatic schedule implemented in the system or manually created. Next a costs estimation is registered in the system, the repair task is executed and a final consumption stage is undertaken. No invoicing takes place in this situation.
Process Variant 7	In this variant, the initial logging of a service request in the system is not considered. Instead, service orders are created immediately as a customer notifies a problem. Further the service order can be processes following a 2-tier service process. Initially, the order is reviewed by a customer service department operator. If the problem cannot be solved, it is escalated to the engineering department and the process follows the same steps as described in Process Variant 1.

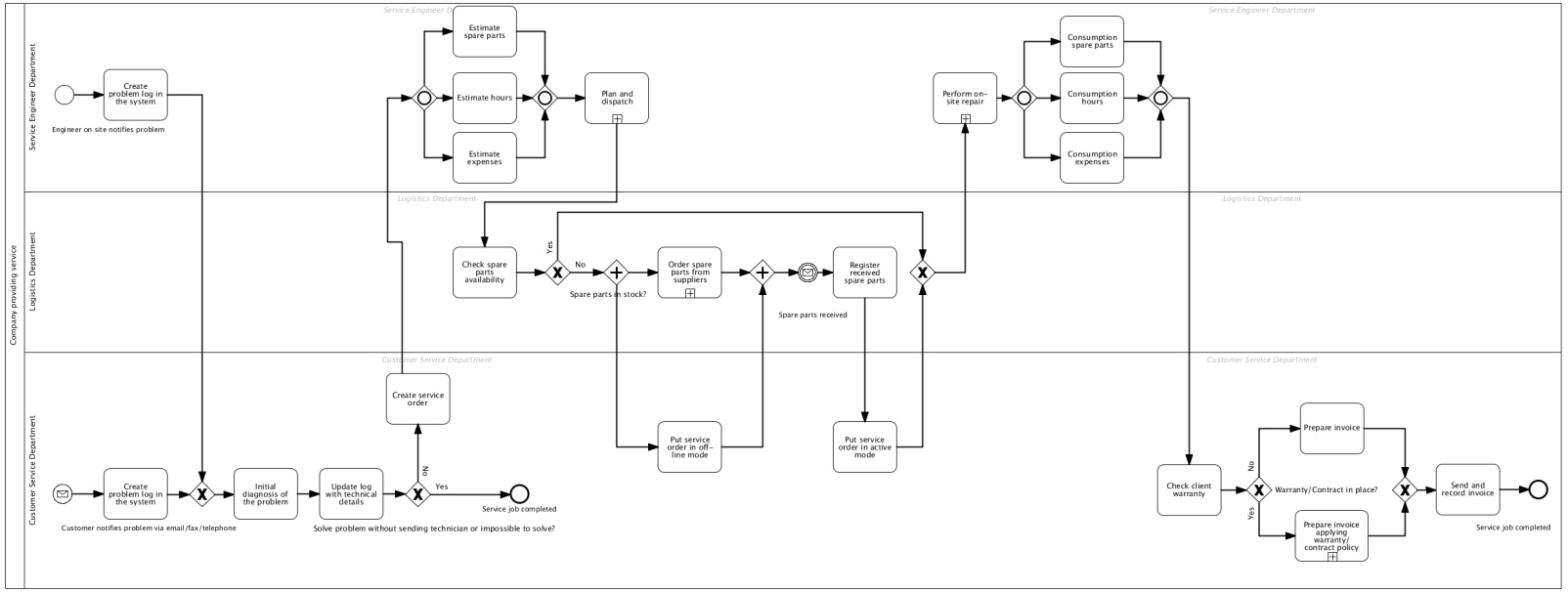

Figure D.1: Process Variant 1

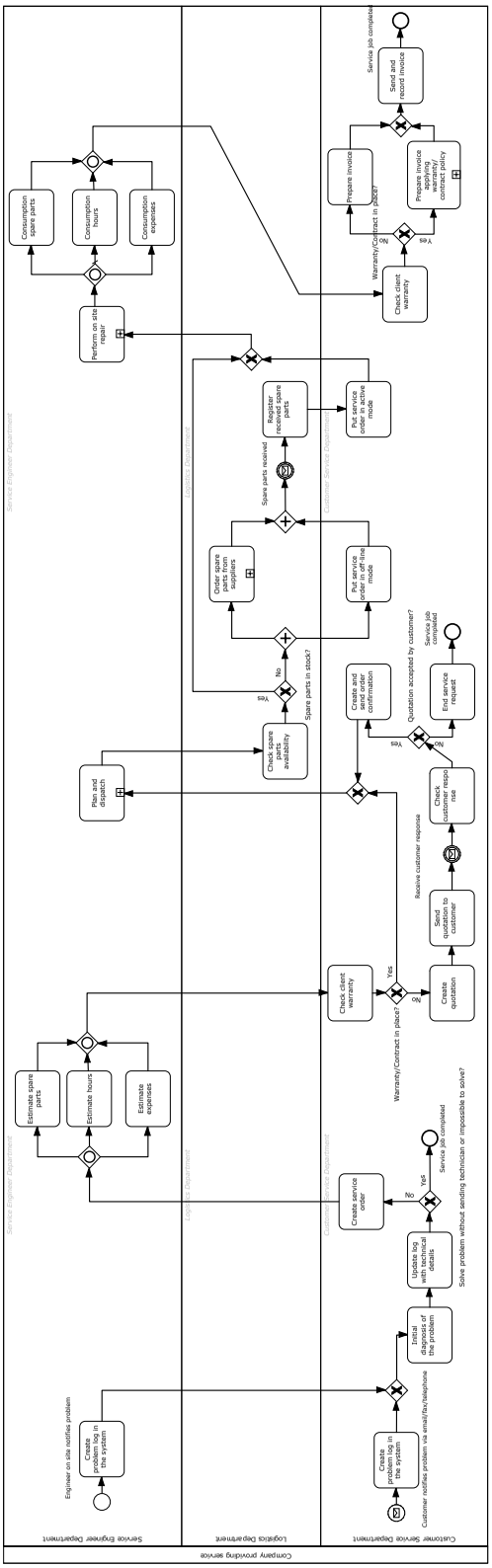

Figure D.2: Process Variant 2

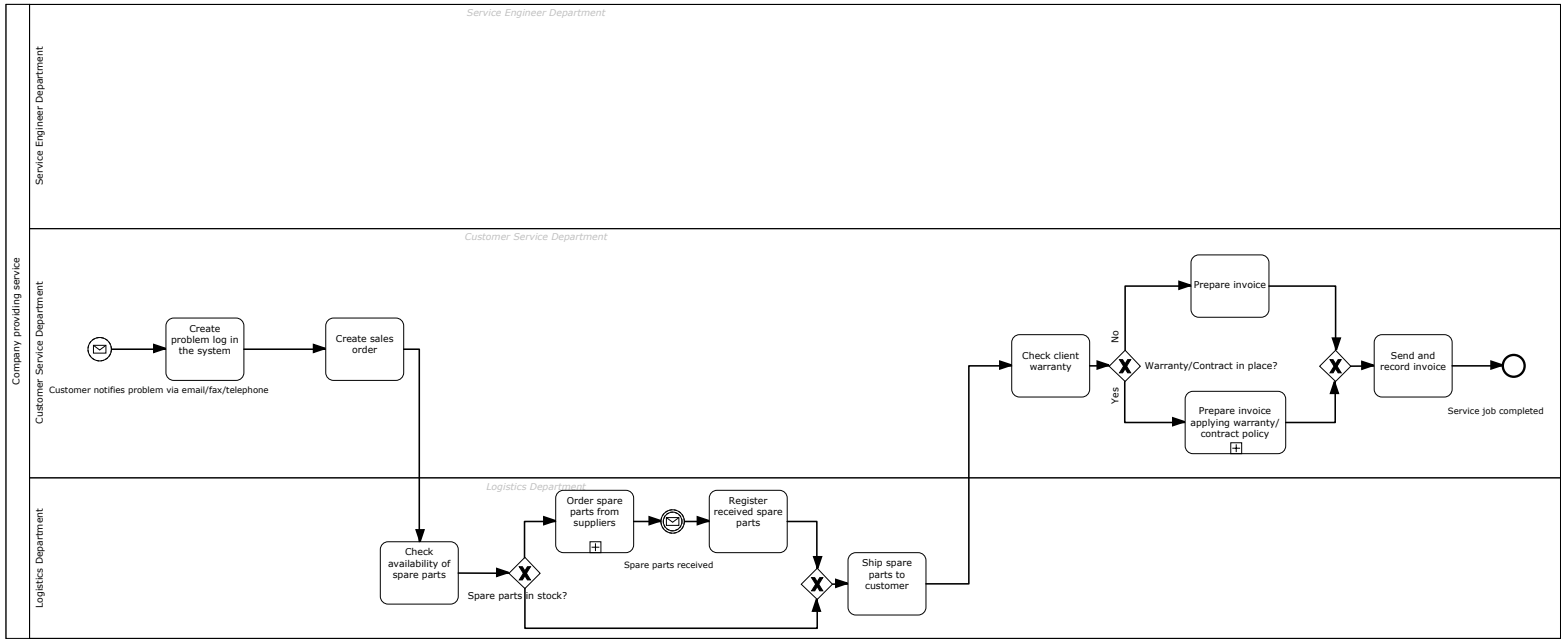

Figure D.3: Process Variant 3

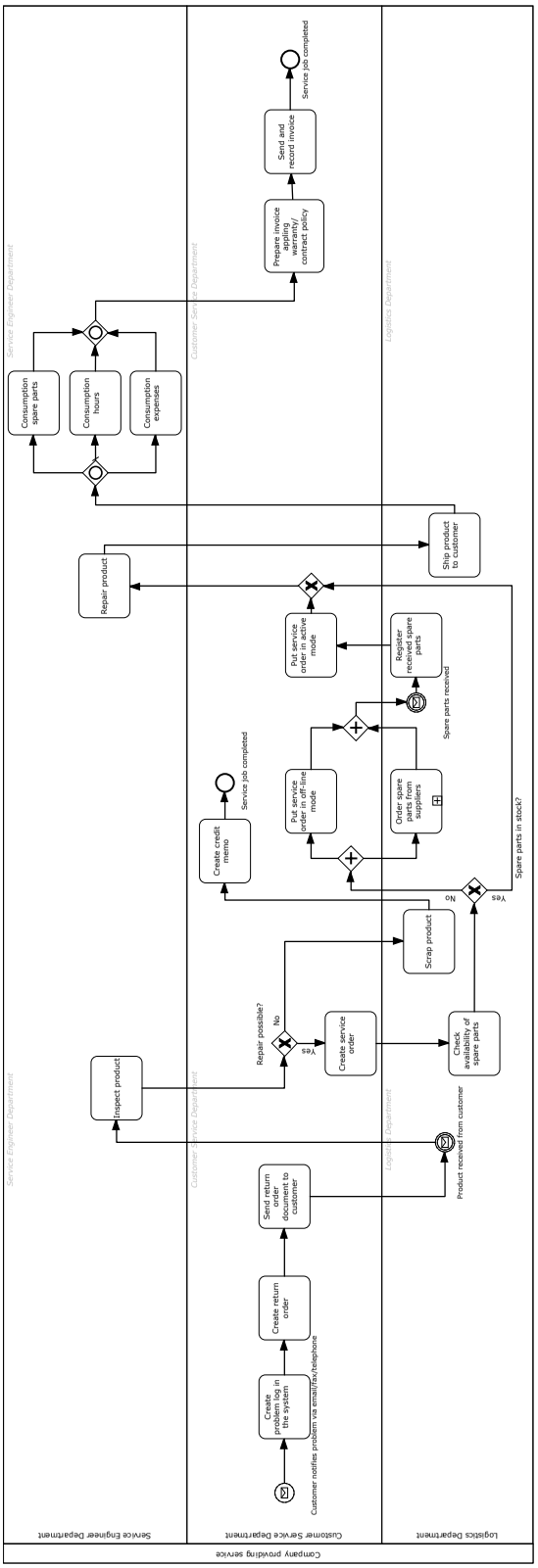

Figure D.4: Process Variant 4

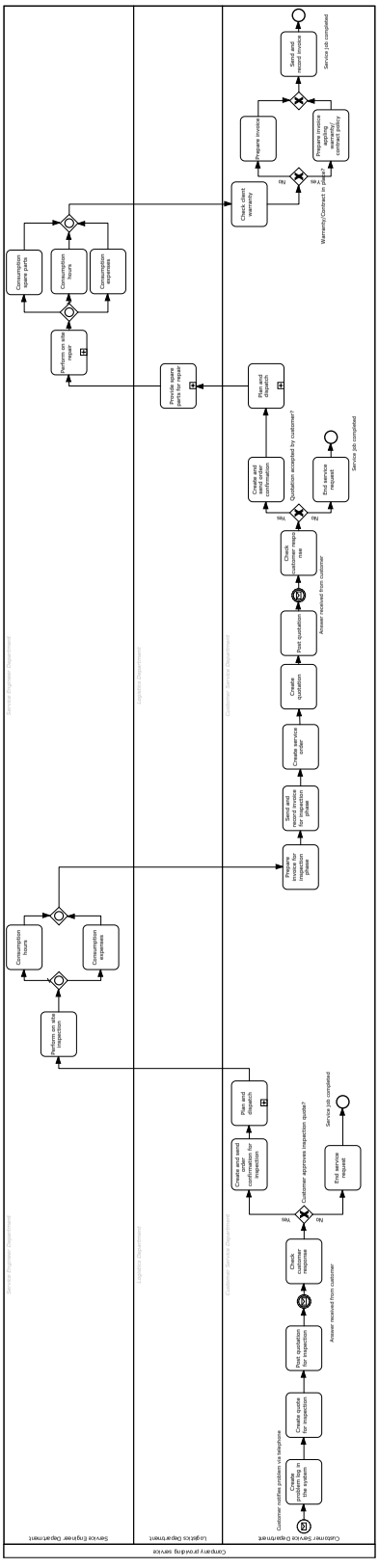

Figure D.5: Process Variant 5

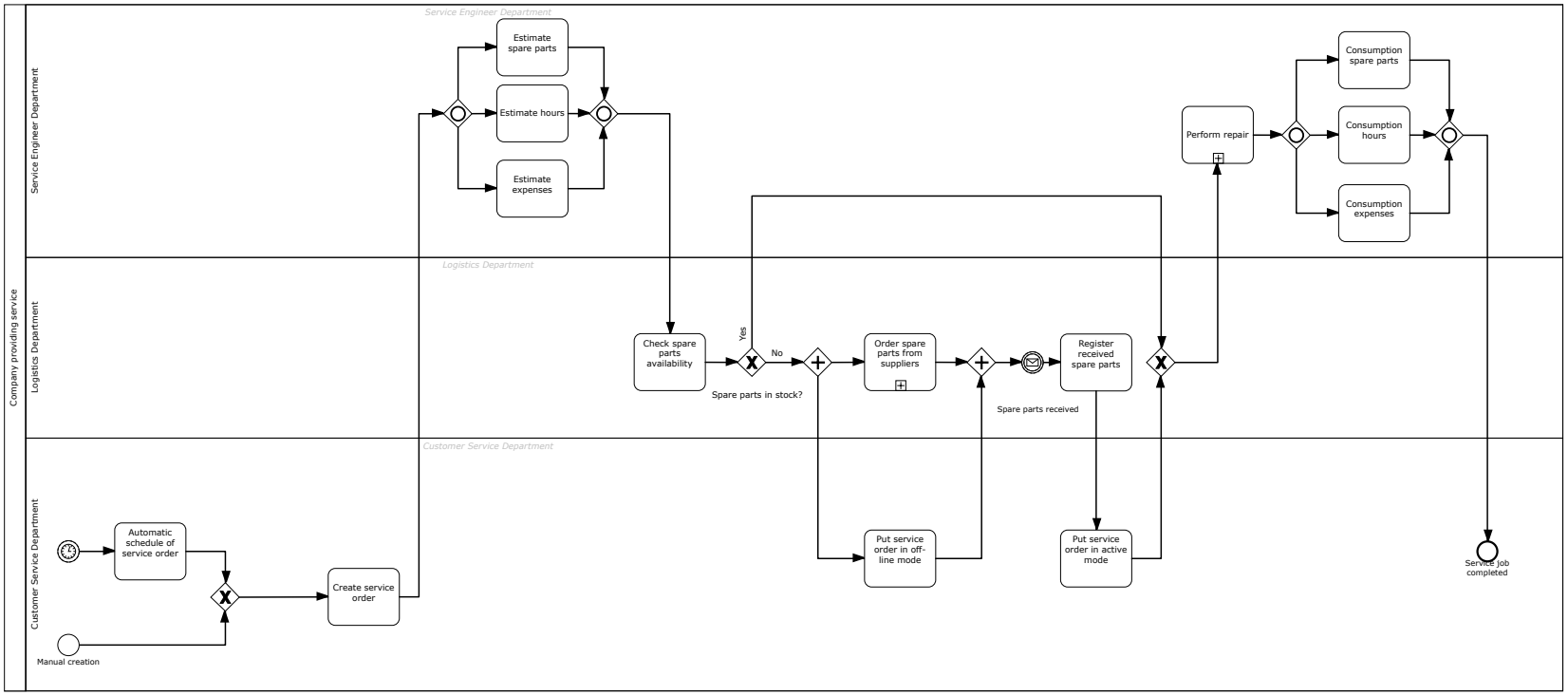

Figure D.6: Process Variant 6

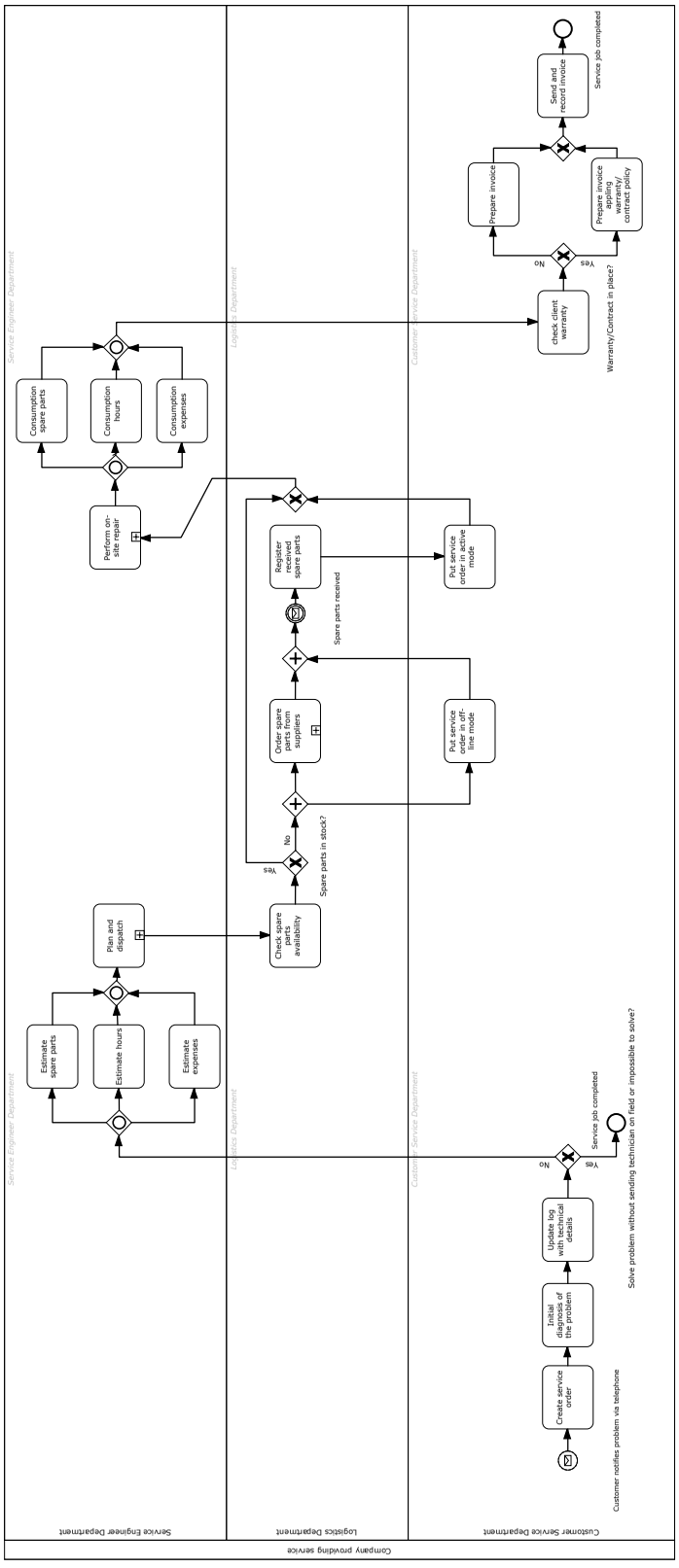

Figure D.7: Process Variant 7

Appendix E

Domain Facts for TSP

In this appendix, the domain facts defined for the technical service process detailed in Chapter 6 are described in Table E.1. We have defined 13 domain facts that handle the configuration of the process. By setting a valuation over this domain facts, an individualization of the technical service process is obtained. Dependencies and constraints between facts, as well as their setting, are resolved through configuration rules.

Table E.1: Technical Service Process Domain Facts

ID : Domain Fact	**Domain Description**
f1 : customer service	This domain fact relates to the customer service process, without considering internal maintenance and handling of returns.
f2 : returns	The *returns* domain fact configures the implementation of the returns process for a service provider company (Process Variant 4).
f3 : internal maintenance	The *internalmaintenance* fact considers the implementation of service for internal maintenance purposes.
f4 : engineer on site	This fact decides upon the type of customer service process provided, by considering the variation in which engineers are dispatched to the customer location for handling repairs. Therefore, this fact is directly dependent on *f1*.
f5 : shipment of spare parts to customer	This fact decides upon the type of customer service process provided, by considering the direct shipment of spare parts to the customer location, remaining for the customer to install the replacements and repair the machine himself. Therefore, this fact is directly dependent on *f1*.
f6 : engineer on site notification	This fact relates to one of the several starting events for the service process, namely the situation in which an engineer already at the customer location notifies the problem and loges the service request into the system.
f7 : customer direct notification	This fact considers one of the several starting events for the service process, specifically the customer directly notifying the problem via phone, email, fax.

Table E.1 – continued from previous page

ID : Domain Fact	Domain Description
f8 : preventive suggestions	The start of the service process can be also triggered directly by the system, through preventive suggestions. Including preventive suggestions into the service process is tackled by this domain fact.
f9 : manual creation of service orders	A different way of triggering the beginning of the service process is by manually creating a service request or a service order in the system. This variation can be considered in the individualized service process by setting this domain fact.
f10 : service requests	Logging service requests in the system for service notifications is considered by setting this domain fact. In this way, service orders are created in the system only when a service request could not be solved at an initial reviewing.
f11 : initial review	A first review of the problem by a front desk operator can be configured by setting this fact as true.
f12 : quotation	This domain fact can set the inclusion of service quotes in the service process.
f13 : inspection	Performing an additional inspection phase prior to handeling the service order can be considered in the individualized process by setting this domain fact.

Appendix F

Mapping of Domain Facts to Configuration Ports in C-YAWL

The mapping between the domain facts and the input and output configuration ports in the C-YAWL process model are summarized in Table F.1. Domain facts are represented by using their fact ID. For a connection between the domain fact ID and the domain fact name and meaning, please refer to Table E.1.

Table F.1: Mapping of Domain Facts to Configuration Ports: $\vee$ = OR, $\wedge$ = AND

	Configuration Port	Allowed	Blocked	Hidden
	Output Ports			
1	*dummy task 1* $\longrightarrow$ *customer notifies problem via tel*	f7		
2	*dummy task 1* $\longrightarrow$ *engineer on site notifies problem*	f6		
3	*dummy task 1* $\longrightarrow$ *automatic generation of service requests*	f8		
4	*dummy task 1* $\longrightarrow$ *manual creation*	f9		
5	*dummy task 2* $\longrightarrow$ *p14*	f13		

Table F.1 – continued from previous page

	Configuration Port	Allowed	Blocked	Hidden
6	*dummy task 2* $\longrightarrow$ *p1*	f1 $\vee$ f3		
7	*dummy task 2* $\longrightarrow$ *p12*	f2		
8	*dummy task 3* $\longrightarrow$ *p2*	$\overline{f5} \wedge$ f10		
9	*dummy task 3* $\longrightarrow$ *p6*	$\overline{f5} \wedge \overline{f10}$		
10	*dummy task 3* $\longrightarrow$ *p10*	f5		
11	*update log with technical details* $\longrightarrow$ *end event*	$\overline{f2} \wedge \overline{f3} \wedge$ f11 $\wedge \overline{f13}$		
12	*dummy task 4* $\longrightarrow$ *p16*	f2 $\vee$ f5		
13	*dummy task 4* $\longrightarrow$ *p17*	f13		
14	*dummy task 4* $\longrightarrow$ *p24*	(f3 $\wedge \overline{f13}$) $\vee$ (f1 $\wedge$ f4 $\wedge \overline{f13}$)		
15	*dummy task 5* $\longrightarrow$ *p18*	f3		
16	*dummy task 5* $\longrightarrow$ *p19*	$\overline{f12}$		
17	*dummy task 5* $\longrightarrow$ *p21*	f12		
18	*dummy task 6* $\longrightarrow$ *p32*	f2 $\vee$ f5	$\overline{f2} \wedge \overline{f5}$	
19	*dummy task 6* $\longrightarrow$ *p32, p34*	f3 $\vee$ (f1 $\wedge$ f4)		
20	*complete spare parts checking* $\longrightarrow$ *p46*	f1 $\wedge$ f4		
21	*complete spare parts checking* $\longrightarrow$ *p44*	f5		
22	*complete spare parts checking* $\longrightarrow$ *p40*	f2 $\vee$ f3		
23	*perform repair inside* $\longrightarrow$ *p43*	f3		
24	*perform repair inside* $\longrightarrow$ *p41*	f2		
25	*dummy task 8* $\longrightarrow$ *p49*	(f1 $\wedge$ f4) $\vee$ f2 $\vee$ f3		
26	*dummy task 8* $\longrightarrow$ *p48*	f5		
27	*complete consumption registration* $\longrightarrow$ *end event*	f3		
28	*complete consumption registration* $\longrightarrow$ *p56*	(f1 $\wedge$ f4) $\vee$ f2		
29	*check client warranty* $\longrightarrow$ *p60*	f2		
	Input Ports			
30	*customer notifies problem via tel* $\longrightarrow$ *create problem log in the system*	f2 $\vee$ f5 $\vee$ f10		$\overline{f10}$

Table F.1 – continued from previous page

	Configuration Port	Allowed	Blocked	Hidden
31	*engineer on site notifies problem* $\longrightarrow$ *create problem log in the system*	f2 $\vee$ f5 $\vee$ f10		$\overline{f10}$
32	*p2* $\longrightarrow$ *initial diagnosis of the problem*	$\overline{f2} \wedge \overline{f3} \wedge$ f11 $\wedge \overline{f13}$		f2 $\vee$ f3 $\vee \overline{f11} \vee$ f13
33	*p3* $\longrightarrow$ *update log with technical details*	$\overline{f2} \wedge \overline{f3} \wedge$ f11 $\wedge \overline{f13}$		f2 $\vee$ f3 $\vee \overline{f11} \vee$ f13
34	*p7* $\longrightarrow$ *initial diagnosis of the problem*	$\overline{f2} \wedge \overline{f3} \wedge$ f11 $\wedge \overline{f13}$		f2 $\vee$ f3 $\vee \overline{f11} \vee$ f13
35	*p8* $\longrightarrow$ *update log with technical details*	$\overline{f2} \wedge \overline{f3} \wedge$ f11 $\wedge \overline{f13}$		f2 $\vee$ f3 $\vee \overline{f11} \vee$ f13
36	*p37* $\longrightarrow$ *put service order in active mode*	f3 $\vee$ (f1 $\wedge$ f4)		f2 $\vee$ f5
37	*p56* $\longrightarrow$ *check client warranty*	f1		f2

Appendix G

Configuration Rules Defined for the TSP

This appendix contains the configuration rules defined for the technical service process detailed in Chapter 6. Each configuration rule has as a result setting a domain fact. We represent domain facts through their names and fact ID. For a description of the domain facts used for configuring the technical service process, we refer the user to Appendix D.

f1: Customer Service(?x, true) $\Longleftarrow$

($Organization(?x)$
$\wedge$ $belongs\_to\_industry\_sector(?x, Oil\_Gas\_and\_Mining\_Machinery)$
$\wedge$ $has\_external\_role(?x, Manufacturer\_Role)$
$\wedge$ $implements\_business\_process(?x, after\_sales\_service\_process)$
$\wedge$ $implements\_fulfillment\_strategy(?x, Engineer\_To\_Order)$
$\wedge$ $delivers\_value\_proposition\_to\_customers(?x, ?value)$
$\wedge$ $hasDimension(?value, ?dimension)$
$\wedge$ $equal(?dimension, "big")$
) (G.1)

f1: Customer Service(?x, true) $\Longleftarrow$

($Organization(?x)$
$\wedge$ $belongs\_to\_industry\_sector(?x, Special\_and\_Large\_Machinery)$
$\wedge$ $has\_external\_role(?x, Manufacturer\_Role)$
$\wedge$ $implements\_business\_process(?x, after\_sales\_service\_process)$
$\wedge$ $implements\_fulfillment\_strategy(?x, Engineer\_To\_Order)$
$\wedge$ $delivers\_value\_proposition\_to\_customers(?x, ?value)$
$\wedge$ $hasDimension(?value, ?dimension)$
$\wedge$ $equal(?dimension, "big")$
) (G.2)

f2: Returns(?x, true) $\Longleftarrow$

(*Organization*(?*x*)
$\wedge$ *belongs_to_industry_sector*(?*x*, *HighTech_Machinery*)
$\wedge$ *implements_business_process*(?*x*, *after_sales_service_process*)
) (G.3)

f2: Returns(?x, true) $\Longleftarrow$

(*Organization*(?*x*)
$\wedge$ *belongs_to_industry_sector*(?*x*, *Commercial_and_OEM_Machinery_and_Parts*)
$\wedge$ *has_external_role*(?*x*, *Manufacturer_Role*)
$\wedge$ *implements_business_process*(?*x*, , *after_sales_service_process*)
) (G.4)

f2: Returns(?x, true) $\Longleftarrow$

(*Organization*(?*x*)
$\wedge$ *implements_business_process*(?*x*, *after_sales_service_process*)
$\wedge$ *delivers_value_proposition_to_customers*(?*x*, *part*)
$\wedge$ *implements_channel*(?*x*, *direct_sales_channel*)
) (G.5)

f2: Returns(?x, true) $\Longleftarrow$

(*Organization*(?*x*)
$\wedge$ *implements_business_process*(?*x*, *after_sales_service_process*)
$\wedge$ *implements_channel*(?*x*, *direct_sales_channel*)
$\wedge$ *delivers_value_proposition_to_customers*(?*x*, ?*value*)
$\wedge$ *hasDimension*(?*value*, ?*dimension*)
$\wedge$ *hasDimension*(?*dimension*, "*small*")
) (G.6)

f3: Internal maintenance(?x, true) $\Longleftarrow$

(*Organization*(?*x*)
$\wedge$ *implements_business_process*(?*x*, *internal_maintenance*)
) (G.7)

f3: Internal maintenance(?x, true) $\Longleftarrow$

(*Organization*(?*x*)

$\wedge$ *belongs_to_industry_sector*(?*x*, *High − Tech_Machinery*)

$\wedge$ *implements_fulfillment_strategy*(?*x*, *Configure_To_Order*)

) (G.8)

f3: Internal maintenance(?x, true) $\Longleftarrow$

(*Organization*(?*x*)

$\wedge$ *belongs_to_industry_sector*(?*x*, *Commercial_and_OEM_Machinery_and_Parts*)

$\wedge$ *implements_fulfillment_strategy*(?*x*, *Configure_To_Order*)

) (G.9)

f4: Engineer on site(?x, true) $\Longleftarrow$

(*Organization*(?*x*)

$\wedge$ *delivers_value_proposition_to_customers*(?*x*, *machine*)

$\wedge$ *implements_channel*(?*x*, *direct_sales_channel*)

$\wedge$ *Customer_Service*(?*x*, , *true*)

) (G.10)

f5: Shipment of spare parts to customer(?x, true) $\Longleftarrow$

(*Organization*(?*x*)

$\wedge$ *has_business_relationships*(?*x*, *Representatives*)

$\wedge$ *implements_channel*(?*x*, *indirect_sales_channel*)

$\wedge$ *sales_through*(?*x*, *Representatives*)

$\wedge$ *Customer_Service*(?*x*, *true*)

) (G.11)

f6: Engineer on site notification(?x, true) $\Longleftarrow$

(*Organization*(?*x*)

$\wedge$ *Engineer_on_site*(?*x*, *true*)

) (G.12)

f7: Customer direct notification(?x, true) $\Longleftarrow$

(*Organization*(?*x*)

$\wedge$ *Returns*(?*x*, *true*)

) (G.13)

f7: Customer direct notification(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $Customer\_Service(?x, true)$

) (G.14)

f8: Preventive suggestions(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $Internal\_maintenance(?x, true)$

) (G.15)

f9: Manual creation(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $Internal\_maintenance(?x, true)$

) (G.16)

f10: Service Requests(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $Returns(?x, true)$

) (G.17)

f10: Service Requests(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $Initial\_Review(?x, true)$

) (G.18)

f10: Service Requests(?x, true) $\Longleftarrow$

($Organization(?x)$

$\wedge$ $Shipment\_of\_spare\_parts\_to\_customer(?x, true)$

) (G.19)

f11: Initial Review(?x, true) $\Longleftarrow$

$$
\begin{aligned}
&(\ Organization(?x) \\
&\wedge \ has\_department(?x, Customer\_Service\_Department) \\
&\wedge \ hasSize(?x, ?size) \\
&\wedge \ equal(?size, "big") \\
&\wedge \ Customer\_Service(?x, true) \\
&) \qquad \text{(G.20)}
\end{aligned}
$$

f12: Quotation(?x, true) $\Longleftarrow$

$$
\begin{aligned}
&(\ Organization(?x) \\
&\wedge \ delivers\_value\_proposition\_to\_customers(?x, purchased\_technical\_service) \\
&\wedge \ Customer\_Service(?x, true) \\
&) \qquad \text{(G.21)}
\end{aligned}
$$