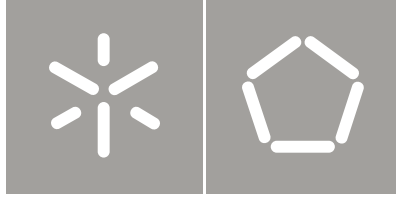Universidade do Minho
Escola de Engenharia

Francisco José Monteiro Duarte

Automated Software Systems
Generation for Process-oriented
Organizations

Francisco José Monteiro Duarte

Automated Software Systems
Generation for Process-oriented Organizations

UMinho | 2014

Março de 2014

Universidade do Minho
Escola de Engenharia

Francisco José Monteiro Duarte

Automated Software Systems
Generation for Process-oriented
Organizations

Tese de Doutoramento
Programa Doutoral em
Tecnologias e Sistemas de Informação

Trabalho efectuado sob a orientação do
Professor Doutor Ricardo J. Machado
Professor Doutor João M. Fernandes

Março de 2014

Ficha Técnica

CENTROALGORITMI

Dedico este trabalho à memória de meu pai João, e de meus avós Guima e Conceição.

À mãe Dores.

Dedico-o também, como exemplo de perseverança, ao João Nuno, ao Hugo, e à Marta.

``Man gets tired, spirit don't.
Man surrenders, spirit won't.
Man crawls, spirit flies.
Spirit lives when man dies.

Man seems, spirit is.
Man dreams, the spirit lives.
Man is tethered, spirit free.
What spirit is man can be.''

M.Scott

Contents

List of Figures

List of Tables

Prefácio

Enquadramento

Esta tese de doutoramento é o corolário de cerca de sete anos de trabalho, iniciados formalmente em Outubro de 2006. De 2001 a 2011 exerci funções como chefe de secção de sistemas de informação na Bosch Car Multimedia Portugal, e foi nesse âmbito que realizei grande parte das atividades relacionadas com os projetos estudados. Participei em todos os projetos nas mais variadas funções, tais como programador, arquiteto de software, responsável pelos testes, ou gestor de projeto, com a intenção de obter experiências o mais alargadas possível das áreas de conhecimento relacionadas com a engenharia de software. Em todos os projetos introduzi fatores de inovação, como sejam as linguagens de modelação e de programação, os modelos de processos, as ferramentas de suporte, modelo de estimativas do custo de software, ou os frameworks em código livre, tentando criar produtos da máxima qualidade com custos reduzidos e baseados em conhecimento existente na comunidade de software.

Em todos os projetos participei em equipas fora de série, quer ao nível tecnológico quer ao nível das relações humanas, o que fez com que o eventual risco resultante da constante introdução de novos fatores nos projetos fosse reduzido ao mínimo. Tenho a certeza que os produtos que realizámos e disponibilizámos à Bosch Car Multimedia Portugal, tantas vezes elogiados e pedidos para serem instalados nas mais diversas localizações Bosch no mundo, são uma fonte de orgulho para todos os que estivemos envolvidos na sua criação.

Agradecimentos

O trabalho realizado para este tese não estaria terminado sem o contributo significativo dos meus orientadores, Professor Doutor Ricardo J. Machado e Professor Doutor João M. Fernandes, aos quais agradeço todos os debates, comentários, e sugestões. No fim desta etapa da minha vida, enalteço o profissionalismo, a capacidade de trabalho, e a competência científica de ambos. Ao trabalharmos juntos, eu tive o privilégio de observar e aprender. Não consigo imaginar um agradecimento maior do que considerá-los como meus amigos.

Prefácio

No decurso da minha atividade como responsável pelos sistemas de informação da Bosch Car Multimedia Portugal fiz parte de equipas técnicas de desenvolvimento com Aurora Cameirão, Paul Vieira, Paulo Dias, José Pedro Araújo, Paulo Leite, Carlos Ribeiro, Nuno Santos, Manuel João Amaro, e Marco Couto. A todos, o meu agradecimento pela colaboração, pela capacidade tecnológica aportada aos projetos, pelos bons momentos, pelo prazer de desenvolver software, pelo brio profissional, e, acima de tudo, pela amizade com que me brindam. Um agradecimento muito especial ao José Carlos Martins pelas discussões, críticas construtivas, e por não me deixar esquecer que os valores prevalecem sempre sobre o momentâneo.

Ao Ralf Haberland, ao tempo CIO da divisão Car Multimedia da Bosch, pelo incentivo às atividades de investigação e desenvolvimento, e pelo interesse nos resultados dos projetos.

Aos especialistas de negócio José Rodrigues Oliveira, Pedro Pereira, Mariana Lima, Miguel Viana, Nuno Oliveira, Luís Lobo, Jónio Reis, e Lutz Behrend, pela partilha do conhecimento profundo do negócio e pelo espírito de equipa.

Aos diversos administradores da Bosch Car Multimedia Portugal, principalmente Carlos Ribas, Uwe Schweigert, Steffan Dick, Sven Ost, Johannes Sommerhaeuser que ao longo do tempo promoveram a criação, ou permitiram, que os distintos projetos fossem elaborados.

Em paralelo com as atividades na Bosch Car Multimedia Portugal, desde 2004 que tenho o prazer de exercer também funções como assistente convidado no departamento de Sistemas de Informação, na Universidade do Minho. Neste âmbito, privo com colegas que são seguramente fontes de inspiração e de conhecimento, como o Professor Doutor Ricardo J. Machado e o Professor Doutor Pedro Ribeiro. Em vários projetos utilizei, com proveito, os conhecimentos que partilharam comigo.

Aos criadores dos frameworks de software utilizados por mim (listados no Anexo "List of Software Tools and Frameworks"), entre outros ServiceMix, ODE, Lyx, JabRef, Eclipse, por permitirem que o produto do seu intelecto seja usado livremente e permita a criação de novo conhecimento.

À Isabel, ao João Nuno, ao Hugo, e à Marta pela paciência pelas imensas horas que lhes roubei.

Sumário

Cada vez mais, as organizações suportam as suas operações em sistemas de software. Torna-se, portanto, muito relevante o correto mapeamento das operações nos sistemas de software.

Esta tese foca-se em organizações orientadas a processos de negócio, devido à relevância dada pelas normas de qualidade, pelos modelos de excelência, e pelos requisitos dos clientes, a esse tipo de estruturação interna das organizações. Nas organizações orientadas a processos de negócio existem diversos fatores, como o tempo envolvido nos projetos de implementação de processos de negócio em software, as diferenças existentes entre os modelos de processos de negócio e a sua implementação real, ou a quantidade e o tipo de recursos envolvidos nesses projetos, que fazem com que os projetos de desenvolvimento de software sejam demasiado dispendiosos, demorem demasiado tempo, e não garantam que o produto de software resultante seja o mais adequado à realidade da organização que o vai usar.

Esta tese propõe que os sistemas de informação e de software devam ser desenvolvidos, desde o início, incorporando os modelos das organizações onde irão ser usados. Além disso, e como existem disponíveis modelos de referência de processos de negócio, esta tese também propõe o seu uso explícito aquando da recolha de requisitos.

Assim, o objetivo principal da tese é propor uma metodologia que se inicie com modelos de processos de negócio e que termine com a geração de sistemas de software, para organizações orientadas a processos de negócio. A metodologia denomina-se BIM e é formalizada através do metamodelo EPF.

Dada a abrangência dos temas a tratar, a tese foi conduzida tendo em atenção que o processo de desenvolvimento de software para suportar organizações orientadas a processos pode ser otimizado. Para melhor mostrar os diversos passos e resultados intermédios, usamos a metodologia de investigação Action Research. A tese propõe que as atividades de investigação sejam terminadas quando uma dada condição de paragem seja atingida, e para isso usa uma avaliação baseada num conjunto de indicadores para os resultados do produto e do processo, e uma adaptação do modelo de excelência EFQM para a forma como foi executado o processo de desenvolvimento. O foco das Action são os sistemas de software MES, essenciais na ligação entre sistemas de software embebido e sistemas ERP.

Nesta tese, as Action iniciam-se com modelos de processos e com arquiteturas de software standard, e terminam com uma proposta de modelo de processo e com arquiteturas de software e tecnologias adaptadas à execução de processos de negócio. A tese propõe também alguns conceitos como IAvO (extensão de modelos de processos de negócio), OBO (componentes de software intermutáveis e não-proprietários), OA (aspetos organizacionais), e PF (framework de processos) para aumentarem a eficiência e eficácia na implementação em software de processos de negócio.

Palavras-chave: UML, RUP, EPF, SPEM, BPM, MDA, BPEL, MES, ServiceMix, ODE, Spring, EFQM, Action Research

Áreas Temáticas: Engenharia de Software; Sistemas de Informação; Gestão de Processos de Negócio

Abstract

Increasingly, organizations support their operations by using software systems, turning very relevant the proper mapping of operations into software systems.

This thesis focuses on organizations oriented to business processes, due to the importance that quality norms, excellence models, and customer requirements put on this type of internal structures of organizations. Process-oriented organizations have characteristics, such as the time needed to implement business processes in software, the differences between the business process models and the real business processes, or the quantity and type of the required resources, that lead to development projects too expensive, taking too long to complete, and that do not assure that the resulting product is the most adequate to the reality of the client organization.

This thesis proposes that the development of information and software system embodies, since the early stages, the models of the organization where they operate. In addition, and since business process reference models are available, the thesis also proposes to use explicitly such reference models by requirements collection time.

Thus, the main goal of the thesis is to propose a methodology that picks business process reference models and ends with software systems, for process-oriented organizations. The methodology is denominated BIM and is formalized by using the EPF metamodel.

Due to the wide scope of the studied areas, the thesis is tailored considering that the development process for process-oriented organizations can be optimized. To express better the intermediate steps and results, we use the Action Research methodology. The thesis proposes that the research activities terminate when a stopping condition is met, based on a set of indicators for the product, and a tailoring of the EFQM model for the development process. The Actions are focused on MES, crucial for the linking of embedded software systems with ERP systems.

In this thesis, the Actions start by using standard process models and software architectures, and end by using a proposed process model, and software architectures and technologies adapted to the execution of business software. The thesis also proposes new concepts like IAvO (extension to business process reference models), OBO (interchangeable and non-proprietary software components), AO (organizational aspects), and PF (process framework) to increase the efficiency and the effectiveness of the implementation of business processes in software.

Part I - Prologue

1. Introduction

This chapter starts by presenting the first introductory notions about information and software systems. Afterward, we explain the surrounding environment of the thesis. Here, we present the thesis as a project with its motivations, goals, and a plan of activities. We also explicate the choice of the adopted research method. Later, we present a basic taxonomy, introduce the scope of our thesis, and state the problem addressed in our work. The chapter terminates by explicating the content of the written document.

1.1. Information and Software Systems Development

Organizations commonly use information and software systems to support their operations. The importance of information and software systems is so critical that some organizations may paralyze when those systems do not work properly. Complementing this important supporting role, information and software systems bring benefits to organizations by acting as tools to increase their performance, for instance, by reducing the execution time of business tasks, by reducing the amount of resources used to execute business tasks, or by increasing the correctness and availability of the generated data when those tasks are executed. Moreover, information and software systems also play a key part in the integration of the organization within its surrounding business environment, by supporting business interfaces with suppliers, clients, or partner organizations.

Within an organization, it is common to discover activities occurring on a daily basis but not explicitly described in any information system. This lack of coverage endangers the integration of the business reality with the data hosted in computer systems and with business processes. It may also happen that software systems are even more limited than information systems in the business scope. In such case, software systems may not support all the tasks modeled in the information system. The shorter scope of software systems can occur because it was planned by the organization (e.g. if the aim is to have a certain amount of manual tasks), but it may also occur because the organization has a deficit in the

1. Introduction

implementation of its software systems. In the latter case, organizations lose business effectiveness.

Software systems must have the ability to adapt their behavior to the business conditions of the organization they support. Normally, the adoption of configuration and customization mechanisms resolves the adaptation challenge. This way, software systems incorporate variable behaviors, without the need to change their source code when some business change occurs. Despite this important characteristic, it is always a difficult task to design and implement proper processes and tools to put into business operation the output of the configuration and customization mechanisms.

The correct configuration, customization, and use of the information and software systems may also not bring into the organization all the expected business benefits. Poor synchronization or misalignment of the information and the software system are the main causes for this lack of results. Inside the same organization, it is possible to discover the existence of different software systems to support exactly the same business activity, possibly used by different departments, and even implemented with different software technologies. Normally, the cause for such redundancy and waste is the existence of non-communicating departments inside the same organization.

It is desirable that an organization has explicitly described and managed information systems covering all its business activities. Such organization can better conduct small improvements or reengineering activities. When there is no explicit description of the business activities, and when there is no explicit management of the information system, then an organization lowers the chances of accomplishing its strategic goals as well of performing expedite daily business tasks. When the description of the business activities is missing, an organization must only trust the ability of its employees to perform those tasks. If the organization has outstanding employees, it may achieve also outstanding results, but one may argue that outstanding achievements based only in personal information is not sustainable. Thus, and since humans are the driving force of an organization, we must design an information system in such a way that it overcomes the weaknesses and fosters the strengths of its users. People make mistakes, so descriptions of business tasks are helpful. Managers need concrete data related with business activities. Otherwise, they use perceptions and hunches instead of reasoning.

If we properly use information systems, we can use rationality and intuition to manage a business, otherwise we are only limited to intuition.

The difficulty to coordinate the information and the software systems inside an organization may be a sound cause for the success of Commercial Off-The-Shelf (COTS) systems, like the case of Enterprise Resource Planning (ERP) systems. For instance, in ERP software architectures a package of cooperative software modules exchanges data to achieve a business-integrated behavior, perceived as homogeneous by the users. This integrated approach has the potential to eliminate the problems that

4

occur when a constellation of non-cooperative, and heterogeneous, information and software systems exists inside an organization.

Nevertheless, the existing synchronization between the behavior and the data of a software system supporting a complete organization, like the case of ERP systems, it is not a definitive guarantee that an organization has solved one of the top problems when using business software systems: the misalignment between the running software systems and the real-world business tasks. Usually, this misalignment problem manifests by a profusion of textual or slide show presentations supposedly modeling business processes but not implemented by any software system. Another symptom is the proliferation of worksheets that store uncoordinated data.

For an organization, the acquisition cost and utilization fees of a COTS system are not negligible. Also relevant are the effort and expertise needed to implement successfully COTS systems. For COTS implementation projects, we can use some indicators to track and evaluate the project, such as:

- the elapsed time between the idea of implementation and the start of productive use;
- the amount of human effort used from business experts inside the organization;
- the effort and expertise of external support for IT technical activities;
- the effort and expertise of external support for business tasks;
- the effort of the project team to test the information and software systems.

In addition, during a project to implement a COTS system, and even when constraints related with the previous five indicators do not exist, the project team cannot guarantee that:

- the resulting customization of the COTS system copes with the current business needs of the organization;
- the implemented COTS system supports the strategy of the organization in short, medium, or long-term periods;
- the project team considers excellent business models for implementation, substituting, or extending the existing ones, besides those hypothetically already provided by the reference model of the COTS system;
- the organization properly understands and uses the business content provided by the COTS system.

During the life-cycle of a COTS, or any other organization-wide software system, the need for adaptations always emerge and it is normally triggered by the changing business environment. The subsequent software tailoring activities require significant efforts, similar to those of a COTS implementation project, and present the same lack of guarantees. To cope with a changing business environment, organizations must continuously invest in update cycles for their information and software systems.

In the project of new information systems and in the adaptation of existing ones, it is always crucial to take into consideration the internal structure of the organization. Thus, an information system has

1. Introduction

to explicitly embody and implement some model of the organization it supports (Fernandes & Duarte, 2004). This characteristic is also valid for business software systems. If the elicitation of requirements does not provide a proper understanding of the business environment and the internal structure of the modeled organization, then serious disparities will exist between the expected and the implemented software behaviors.

The internal structure of organizations traditionally follows a hierarchical model and static organizational charts are their normal form of visualization. This static view may induce that organizations are inactive and the main purpose of the employees is to please their chain of command. Such a static view is not the best choice to represent the needs of the customers, neither the needs of the internal clients.

Certainly, the main problem of an organization is not how it is represented, but how it behaves in the business reality. Hierarchical organizations and, thus, with internal fragmentation on cross-departmental activities, are focused on departmental goals rather than on the customer's needs. These kind of organization is less able to cope with changing requirements from customers and have a higher risk of not addressing adequately changes in their markets.

The concept of business process is an answer to the hierarchical view and to shape the operation of organizations. Hammer & Champy (1993) consider a business process as "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer". This definition points out the need to focus the organization on traversal processes that provide business value to the ones who most need them: its customers.

The development of information and software systems is a complex task. Despite new technologies, methods, tools, and computer skills widespread, it is a difficult task to understand the business environment of an organization, to elicit proper requirements, to steer them until the final product expresses characteristics according to the customer needs, or to fulfill explicit and implicit functional and non-functional requirements. An additional increase in the complexity of development projects is that all the activities must be executed keeping costs, time, capacity, and have the quality controlled in accordance with restrictive, and sometimes conflicting, goals. This scenario increases the likelihood for project failures.

For the execution of projects of information or software systems, some hints to measure the degree of success can be devised by checking:

- the fulfillment of the captured requirements;
- the fulfillment of both explicit and implicit requirements;
- the number of new business processes installed;
- the number of improved business processes installed;
- the needed time for the implementation;

6

- the way the system deals with future changes.

During development projects, the business experts are not the only responsibles for the improvement of the current business processes. The project team must have a pro-active approach by proposing enhancements for business processes and not only expecting requirements to be verbalized by the client.

The possibility to introduce new, or redesigned, business processes during project time is desirable, not only for the sake of the client organization but also for the overall project success. At the end of a project, the way the resulting software product is able to cope with the business needs of the client organization is the main measure of the success of the project team.

To bring business-specific knowledge into software products, like the knowledge on how to organize a supply-chain, a project team should consider excellent business process reference models embodying best practices. In such a way, the project team can compare the requirements expressed by the client for the business processes with those stated in the business processes reference models. Thereby, the project team can explicitly manage the differences between both models.

The client must decide its strategic options having in mind that other options for its business processes exist, probably better than those of its knowledge, and are usable. Nevertheless, for a particular organization, the processes contained inside a business process reference model (BP-RM) are not always better than own-developed processes.

Organizations should always create a balance between own-developed and reference model business processes. The best suited for its needs, are the ones to implement.

This thesis proposes **a methodology that can bring into business software development projects the better of these two different alternatives: the explicit consideration of excellent business processes and the particularities found in every organization**. The existence of well-defined development processes, methods, and techniques that allow the automated implementation of a given set of business processes in a software system brings significant advantages to the organizations that use them. The advantages can imply reductions on the implementation time of business software systems or increases on the quality, understood as the degree of fulfillment of the client requirements.

Currently, models of business processes are materialized by using different notations, ranging from natural language, with a reasonable degree of subjectivity, to formal descriptions, like Colored Petri Nets (CPN) (Jensen, 1992), which are too specialized to be properly understood by the vast majority of business experts. This way, there is a necessity to use unambiguous but commonly understandable business processes modeling languages. The use of semi-formal languages, like the Unified Modeling Language (UML) (Rumbaugh et al., 1998), is a step forward, although insufficient, to include the expected business behavior into a business software system. In addition, the use of workflow-like

systems to back up business processes is not a complete solution to implement an adequate support for business processes in software, due to the lack of business semantics in workflow systems. Workflow systems can be the basis to model sub-processes inside an organization. However, by missing business semantics, workflow systems can also be a driver to implement misaligned business processes in relation to the strategy of the organization.

In productive software environments, the existence of executable models of requirements, understandable and operable by both business experts and software engineers allows less model transformations, faster implementation times, and improvements on the resulting software product, because the same model could pass through all the development project phases without any transformation.

Currently, many business software systems are of COTS type. In the implementation of COTS systems, the customization tasks prevail over the programming tasks. In this situation, a detailed technical knowledge is required from the project team members to deal with the COTS software architecture. Even when this detailed knowledge is available, it is not straightforward, neither immediate, that the transformation of business requirements into software characteristics can occur.

One of the key aspects to improve the development of business software is the use of methodologies that embody lean concepts and, at the same time, consider business best practices. Therefore, during the development of software systems, the project team should consider the following aspects:

- a software system must embody and implement a model of the organization it supports;
- when capturing the requirements, the project team should consider excellent business processes available as benchmarks;
- the use of automated techniques that transform models of business processes into software systems brings competitive advantages to organizations;
- the use of models of business processes understandable by the business experts assigned to the project;
- currently, the use of COTS software systems requires significant customization and less traditional programming efforts, in both cases, demanding specific technical knowledge from software engineers;
- even for skilled software engineers, COTS software customization tasks are not straightforward, mainly due to the complexity and interdependency of the business processes, but also due to the specific expertise needed to perform those tasks.

As a response to the inefficiencies of information and software systems technological and methodological approaches previously described, we think that project teams should use a proper development environment. This environment should consider all the relevant bodies of knowledge, ranging from generic business process models to software systems architectures and implementation technologies. Additionally, it seems to be worthwhile to bring automation into development methodologies in order to accelerate the process and to improve the quality of the final product. Therefore, **this thesis presents a holistic methodology to obtain a software system compliant, and deeply tied,**

with the set of business processes of a given organization. The methodology adopts BP-RMs as input and can use a customizable software stack, composed of free software frameworks together with own developed software, to allow the immediate execution of business processes in a software system.

1.2. Motivation

Business software development is a complex task and does not guarantee the quality of the resulting product. The existence of models of the organization where the business software is supposed to run, namely in process-oriented or in matrix organizations[1], is only the first step to reduce the complexity of business software development. During a project, the requirements for the software system may change due to business conditions. Additionally, each project team executes a set of tasks impacting quality, time, methodology, and cost. The success of the project is highly dependent on the quality of the human resources.

The existence of automated mechanisms to transform the requirements into characteristics of the software brings value into the resulting product of a development process in the following dimensions:

- quality, since requirements are better fulfilled;
- time, as the duration of the development efforts is shorter;
- methodology, because the project team uses a defined set of tasks;
- cost, because automated tasks tend to require less human effort.

Thus, business experts can manipulate the requirements by using a model easily understandable and with an immediate impact on the behavior of the software system. The use of such mechanism is appropriate in two contexts:

- during the development of a software system, independently of being a new system or an update of an existing one;
- during the support and maintenance activities.

Inside the software industry, the focus on system and personal productivity software is changing towards the automation of business processes. Greenfield & Short (2004) state that Business Process Management Systems (BPMS) are the next paradigm shift in software systems, making process-oriented applications the backbone of application portfolio. Within this context, several questions and challenges triggered the motivation to perform the present thesis, namely:

- why does it take months to complete a project to implement business processes in an organization?
- why there are so many differences between the business reality and running business software?

---

[1]Matrix organizations are organizations where processes and departments co-exist.

- why there is the need to have dozens of business experts and IT personnel in ERP implementation projects?

- in 10, 20, or 100 years will we still use the current software development approaches? So, what is the right investigation path?

- when creating information systems, why does automation seems not feasible to be implemented?

- will we continue to implement high-level concepts, like "Delivery Note" or "Invoice", mainly by using primitive concepts such as if-then-else rules and loops?

1.3. Ambit

If the concepts, methodologies, and tools proposed in this thesis are to be used outside the scope stated in Sub-section 1.3.2, then their applicability must be first checked.

We also consider important to define the problem treated in the thesis, in order to obtain more easily sound solutions for its root causes.

1.3.1. Taxonomy and Basic Definitions

We use several expressions and terms throughout this thesis. Therefore, it is essential to reach a common understanding for their basic semantics.

APDSI (2005) proposes the following definitions:

- information system: it is a system comprised of human resources, material resources, and procedures allowing the acquisition, storing, processing, and diffusion of the needed information for an organization, either being or not an informatics system.

- informatics system: it is a system made of one or more computers, peripheral devices, and software that processes data.

In the scope of this thesis, we consider a software system as a subset of an informatics system. We do not explicitly address the remaining parts of an informatics system.

We use a similar approach to distinguish between information and software systems, knowing that sometimes they may have the same scope, due to the degree of IT implementation.

Figure 1.1 depicts the differences that may exist between the business scope of an information system and the scope of a software system, within the same organization. An information system has more business content and components than the corresponding supporting software system. Nevertheless, software systems may exist with components not used within the scope of the organization where it runs (e.g. the case of business processes available inside an ERP, but not used by the organization).
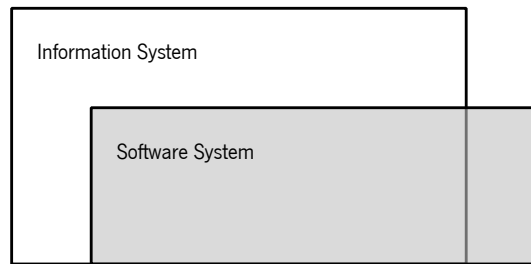
Figure 1.1.: Scopes of information and software systems within an organization.

Our proposals are applicable independently if someone considers that an organization has just one or a multiplicity of information systems. The same applies to software systems.

Next, some of the most used terms are defined, to allow a better understanding of what we intend to mean with them within the scope of this thesis.

organization:

It is any kind of public or private association that pursues a collective goal, either in the economic, social, or public fields of activity, with or without profit purposes. An organization has some form of arrangement of resources (e.g. social, technical), and it embodies and pursues a set of objectives, which somehow controls its own performance. Organizations have boundaries separating them from the surrounding business environment.

business process:

It is a set of coordinated traversal tasks taking place inside an organization, but that may go beyond its boundaries. The requirements of the clients should be the main drivers to design a business process.

process-oriented organization:

It is an organization that defines and manages business processes explicitly. This definition includes matrix organizations because they incorporate business processes, similarly to a pure process-oriented organization, but intersecting with departmental responsibilities.

information system:

It is a system where people and software interact through the manipulation of data and processes and by using some kind of technology. In the scope of this thesis, information systems have a broader range than software systems, because the first may include tasks handled without the intervention of computers. Information systems should also embody a model of the organization it supports.

immediately executable information system:

It is an information system that can be used, and is available, to the organization immediately after a new design of a business content. For that, a business process depicted in a slide show presentation does not belong to an immediately executable information system.

software system:

It is a set of software components supporting the business operation of an organization, in a coordinated and cooperative way. A software system should embody a model of the organization where it runs. In this thesis, the scope of software systems is restricted to software supporting business tasks.

immediately executable software system:

It is a kind of software system that can provide immediate support to a new design of business content, possibly through the use of a mechanism such as orchestration. In this sense, a software system developed from scratch to support a new design of a business process can not be considered an immediately executable software system.

business process reference model (BP-RM):

It is a set of industrial, or academic, accepted definitions of business processes, incorporating process best practices, business actors, and possibly business indicators. BP-RMs are organized by vertical business markets (e.g. automotive industry) or by horizontal business functions (e.g. management of information technology services).

project:

It is a time-based event that produces a unique product. It must have defined start and end times. Projects must be planned, executed, and controlled. A team of persons, with limited resources, executes them. We follow the definition provided by the PMBOK (Duncan, 1996).

In business software development, a software system should always be part of an information system. When an information system is completely implemented in software (i.e. there are no business activities without software implementation), the information system and the software system are indistinguishable.

Inside a process-oriented organization, where a model of its business processes exists, it is appropriate that only one supporting information system and one business software system exist. Thus, organizations should avoid partial, and non-communicating, information systems or software systems. In such

case, it indicates that the managers do not govern the organization in an integrated way. The existence of a myriad of disaggregated software systems, not managed in an integrated way, is not considered a best practice since knowledge is not shared and, at the end, not all the potential business value is materialized.

1.3.2. Scope of the Thesis

The scope of the thesis is limited to information and software systems for process-oriented organizations. The proposals can be used either during project time or, alternatively, during normal operation, either in reengineering contexts or in continuous improvement initiatives.

For the software framework that supports the proposals, whenever feasible, we use standard and free software, together with the software developed by ourselves. Currently, the free software frameworks available in the market are able to support business critical operations. Nevertheless, our proposals are also valid for proprietary software solutions.

1.3.3. Definition of the Problem

As stated by Berry (2004), a PhD proposal targets a real and unsolved problem. According to Carvalho (2004), a PhD proposal should also exhibit the following characteristics:

- the problem is relevant;
- the outcome is valid;
- the student as a notion of the strengths and weaknesses of the work;
- the student knows how to carry on the work.

In this thesis, the problems dealt relates with the efficiency and the effectiveness of development projects of information and software systems. The thesis also addresses the resulting software products and their use after project time.

We foresee six different perspectives on the constraints of the problem, namely:

- project duration, because the quickness of reaction of an organization to a changing business environment may be crucial for its survival;
- human resources, because as much the used capacity increases as much as the project bill grows. Staffing is a cornerstone of a project, due to later impacts in time and costs, both caused by unskilled members;
- project cost, because it is always a constraint in any engineering context;
- qualifications of the team members, because the needed expert skills can make difficult to staff properly the project teams, and may affect the capacity, costs, and time;
- quality of the resulting information and software products, because the requirements must be

> met;

- cost of use and maintenance, after project time.

Thus, the problem investigated is:

How to develop software systems that support business processes, in an efficient and effective way.

1.4. Goals

The main goal of the thesis is:

(MG) To propose a methodology to generate immediately executable information and software systems for process-oriented organizations, starting with models of business requirements.

The secondary goals, perceived also as success criteria, are:

- (SG1) The resulting software products must have quality.
- (SG2) The proposed methodology should allow development projects with a short execution of time.
- (SG3) To specify and to implement a software framework to support the development of business software with the methodology.
- (SG4) The resulting methodology and software products should be able to be used by business experts without the intervention of software experts.
- (SG5) The proposed methodology should be holistic, ranging from business process models to running software.
- (SG6) To promote the composition of software solutions based on free software and originated in different vendors.

The thesis intends to permit business experts to generate software systems for a process-oriented organization, if we provide them a set of proper mechanisms, tools, and guidelines.

1.5. Research Method

Gable (1994) identifies two different basic types of research methods related with information systems research: qualitative (e.g. the case study method), and quantitative (e.g. the survey method). In qualitative methods, the researcher collects data from a small number of experiences that lead the researcher into the details of specific environments. These methods have the advantages of studying information systems in their natural environments, or, due to its deeper analysis, to discover hidden

knowledge. Difficulties may arise to prepare proper environments to allow the deduction of conclusions, to repeat the case study, or to generalize the proposals. Yin (1984) suggests that qualitative research methods are appropriate when the objective is to study contemporary events, like in the case of this thesis.

In quantitative research methods, the researcher collects data from a large number of organizations, using questionnaires or interviews. After having a representative sample, relationships can be inferred and proper conclusions can be drawn from the data analyzed. These kinds of methods have drawbacks such as a short analysis validity time.

Gable (1994) proposes the combination of these two research approaches, mainly by conducting a multiplicity of case studies after a pilot case study, by conducting sampling surveys, and by producing descriptive statistics. Later, the researcher validates the models and the hypotheses. This thesis materializes the proposal of Gable through the use of the Action Research method (Lewin, 1947).

Avison et al. (1999), recommend the use of the Action Research method for the information systems research, as Action Research is seen as a unique way to associate research and practice. The Action Research is a method where the "research informs practice and practice informs research synergistically". Baskerville (1999) states that complex social processes, as the ones occurring in the context of organizations and their information technology use, "can be studied best by introducing changes into these processes and observing the effects of these changes".

The Action Research method includes several planned, executed, and studied cases (called Actions), where theory moves into practice. Later, the researcher evaluates the theory in order to improve the next Action. Since the main purpose of this thesis is to propose an automated way to generate information and software systems for process-oriented organizations, it is more appropriate to focus on the validation of the proposals rather than to make a comprehensive analysis of the different ways of generating information and software systems. We conduct several Actions to demonstrate the validity of the proposals and we check their convergence with the goals of the thesis.

In our opinion, the Action Research method also embodies the main characteristics of a PDCA (Deming, 2000) continuous improvement cycle.

The adopted approach to accomplish the goals of the thesis is composed of the following macro-steps:

1. Study and analysis of the state-of-the-art related with the organizational, information and software systems domains. This theoretical work complements our practice of several years of work experience in the development of information and software systems for process-oriented organizations, and in the governance of business processes.

2. Post-mortem analysis of an Action. We conduct a qualitative, yet quantified, analysis of one software development project in a process-oriented organization.

3. Action Research setup and execution. Based on evidences collected from the previous Action, and based on our working experience, we propose improvements for methodologies, technologies, and tools to allow a more automated generation of information and software systems for

> process-oriented organizations.

4. Thesis validation. At the end of each Action, we check the fulfillment of the thesis goals.

5. Learning feedback. The comparison of Actions, expressed in lessons learned, is the basis to propose future enhancements on how to conduct Action Research studies, and mainly on how to develop software for process-oriented organizations.

We do not perform the previous sequence of macro-steps using a pure waterfall process. Instead, we use an iterative and incremental approach by creating a succession of Actions. This way, it is acceptable that overlapping macro-steps occur. In Section 4.2, we explain in detail the adequacy of applying Action Research to the development of business software.

1.6. Plan of Activities

To reach the goals, and complying with the chosen research methodology, we adopt the activity plan presented in Table 1.1. The first phase ("preparation") is where the theoretical research activities take place, namely for business organizational models, languages to model business processes, and software engineering areas, in this latter case according to the SWEBOK (Abran & Moore, 2004). The goal of this first phase is to elaborate the state-of-the-art. By the end of this phase, we should know the main authors and work in the related research fields.

In the second phase ("thesis proposal"), we present contributions for theory, methodologies, and tools. This is where we show the core proposals to address the automated generation of software systems for process-oriented organizations. It is our intention to develop a methodology able to generate automatedly information and software systems, in a short period of time, and with a resulting quality product. This generation should be accomplished by a development team staffed not only with software engineers. Due to automation, the number of business experts in the development teams tends to increase, and, in opposition, the number of software experts tends to decrease. The generation of new software systems can be accomplished without major interventions from software engineers, unless the project requires new business activities not yet implemented in software, as explained in Chapter 7. The goal of the second phase is to define the first proposal for the knowledge body to support an automated generation of information and software systems for process-oriented organizations. The success criteria for this phase are to have the first proposals for a body of knowledge and a methodology ready for use in the setup and execution of an Action.

In the third phase ("execution of Actions"), all the Actions are planned and implemented. We validate, in a real business environment, the proposals made in the "thesis proposal" phase and the lessons learned from previous Actions. The goals of this third phase are the participation in development projects for information and software systems, to derive conclusions about current theories and practices, and to suggest improvements for the subsequent Actions. The success criteria for this phase are to have running software systems, to participate actively in their development, and to have, at the

Table 1.1.: Planned thesis phases.

Phase #	Name	Begin Date	End Date	Description
1	preparation	01.11.2006	31.12.2007	theoretical research to draw the state-of-the-art
2	thesis proposal	01.01.2007	31.12.2009	define the contribute, in theory, methodologies, and tools
3	execution of Actions	01.11.2006	31.10.2010	implement and analyze the Actions
4	conclusions and text revision	01.01.2010	15.12.2010	extract conclusions from the Actions and make proposals for improvements
5	thesis presentation	$1.^{st}$ semester 2011	$1.^{st}$ semester 2011	public discussion of the thesis

end, an implementation of a software system that meets the thesis goals.

We use the fourth phase ("conclusions and text revision") to derive the conclusions and to compare the Actions, including the lessons learned, the analysis of the results, the validation, and scope of the proposals, and some insights about possible future work. In this phase, we also review the full text. The goal for this phase is to think over the proposals and to incorporate potential new ideas obtained from the execution of Actions into the future work. The success criterion for this phase is to have the full thesis document completed.

The fifth, and last phase, is the public presentation and discussion of this thesis.

1.7. Organization of the Document

We organize this thesis with four parts, each one with one or more chapters. The first part ("Prologue") contains the thesis scope, and the introduction to the proposals. The second part ("Fundamental Concepts") comprises the state-of-the-art for the key areas under study and a critical analysis about the presented state-of-the-art content. The third part ("Contribution") is composed of the proposed concepts and methodologies, the supporting technology, and the reports about the execution of the Actions. The last part ("Epilogue") contains the lessons learned, comments, final critical analysis of the thesis, and the future work.

Table 1.2 presents the distribution of the chapters by the four parts. The underlying baseline of this thesis is to analyze the key concepts inside the domains of business, information, and software systems in order to create a crossing bridge throughout all these domains at software systems development time and during their productive use. With this approach, we assume that a business software system should always embody the model of the organization it supports. Afterward, we present the thesis proposals in the form of new concepts, a methodology, and a set of supporting technologies to allow a practical use of the theoretical aspects of the thesis.

Besides the first chapter, all other chapters contain a "Conclusions" section, where we present a critical

Table 1.2.: Structure of the thesis

Part #	Part Name	Chapter #	Chapter Name
I	Prologue	1	Introduction
II	Fundamental Concepts	2	Organizational and Information Systems Domains
II	Fundamental Concepts	3	Software Systems Domain
III	Contribution	4	Action Research in Action
III	Contribution	5	Building Software to Support Business Processes
III	Contribution	6	Running Business Process Reference Models in Software Frameworks
III	Contribution	7	The Business Implementation Methodology
IV	Epilogue	8	Conclusions

analysis. Next, we list the themes discussed in the chapters of parts "Fundamental Concepts" and "Contribution":

Chapter 2 - Organizational and Information Systems Domains We start by presenting the concepts related with the internal configuration of organizations and the role that such structures play in the development of software. Afterward, we describe in detail process-oriented organizations and we present the business strategy concept and the impact it has in the structure of organizations. We then discuss the type of internal structure that organizations can assume. Later, we show in more detail issues related with business process-oriented organizations, including BP-RMs and business process management. We also present ontologies and domain models, as tools to add semantics for process descriptions. We describe metamodels for process-oriented organizations with the intention of adding some formality to an organizational model. Finally, we present concepts from the information systems domain, namely processes, business modeling, and languages to support them.

Chapter 3 - Software Systems Domain This chapter contains core concepts related with software engineering, like software development processes, requirements engineering, or software patterns. It is also emphasized the importance of software frameworks. We terminate addressing concepts, like aspect-oriented software development, due to their expressiveness, and the ability to use them in the business processes domain.

Chapter 4 - Action Research in Action This chapter presents the Action Research and discusses its adoption for the development of software to support organizations. We also formulate a proposal to evaluate Actions combined with metrics for the resulting software product. Finally, we present an initial research Action that acts as the trigger to use such kind of research methods and as motivation to continue the research on how to develop software for process-oriented organizations.

Chapter 5 - Building Software to Support Business Processes This chapter presents two Actions. We focus the Actions on the development of software to support process-oriented organizations with general-purpose languages, processes, and tools.

Chapter 6 - Running Business Process Reference Models This chapter contains the last two Actions, which improve the software support for process-oriented organizations by using process reference models. In the last Action, we use our proposed methodology. We also explain an own-developed software environment that allows some automation and software reuse during the development and execution times of software systems.

Chapter 7 - The Business Implementation Methodology This chapter explains in detail the main outcome of our work: a new methodology, which includes process reference models as input and delivers a business software product as output.

The thesis also contains an Appendix part, where we show deliverables of the Actions and details of the proposals. We also present source code of the software developed during the last Action.

Part II - Fundamental Concepts

2. Organizational and Information Systems Domains

The main purpose of this chapter is to discuss the internal structure of organizations, and the role that such structures play in the development of information and software systems.

This chapter starts with the presentation of business strategies and the impact they have on the generic configurations for organizations. We explain and compare the internal formats that organizations can assume. Then, we show topics related with process-oriented organizations, including BP-RMs and Business Process Management (BPM). Due to their importance for the holistic view of an organization, we present and evaluate Enterprise Architecture models, incorporating, among others, business processes and IT architecture views. We also present ontologies and domain models, as tools to add semantics for processes descriptions, and we describe metamodels for process-oriented organizations with the intention of adding formality to organizational models. We present concepts from the information systems domain, namely processes, business modeling, and languages to support it, and methodologies. This chapter ends by presenting a particular type of software systems, the MES, later used in the Actions.

2.1. Introduction

Organizations have different purposes. Some exists in business environments, like the ones related with commercial or industrial activities, some are governmental, and some others do not have profit purposes. All types of organizations have internal structures representing, among others, their goals, staff, or activities. These internal structures are models of the reality existing inside an organization (e.g. organizational charts representing hierarchical relations between roles and the high-level tasks assigned to the associates). Complementary to the static models, such as the organizational charts, other models exist to represent the dynamics of the activities of an organization (e.g. work instructions, quality procedures). These latter models include explanations on how to perform activities, in order to guide the execution of tasks by associates.

2. Organizational and Information Systems Domains

The models of an organization represent its structures and activities and are related with the size, the maturity, and the surrounding environment of the organization. In an organization with only a few associates, it could be admissible to have an organizational chart communicated orally. In opposition, in a multinational company it is likely that an intranet environment provides electronic information. In addition, quality standards may raise the maturity level of an organization by promoting the explicit sharing of knowledge between associates performing the same tasks (e.g. creating a written working instruction for some activity occurring in a factory shop-floor performed by distinct employees in different shifts). In the same way, the business environment, like the industry where the organization operates, may force internal procedures of information storage and retrieval, like traceability systems for products or processes in medical or automotive areas.

For all that, business information characteristics, like the volume or the complexity of the information, drive organizations into the creation of information systems. The characteristics of the organization, like the availability of information or the automation of data recollection, may result in a computer-based implementation of the information system, namely by recurring to a business software system. It may also occur that the organization uses some software system to manage the knowledge existing inside it.

Table 2.1 presents a knowledge taxonomy and the different types of knowledge are mapped to the knowledge related with business processes.

Table 2.1.: Knowledge taxonomies and examples (adapted from Alavi & Leidner (2001)).

Knowledge Types	Definitions	Examples
Tacit (either Cognitive or Technical)	Knowledge is rooted in actions, experience, and involvement in specific context. Cognitive knowledge is based on mental models whereas Technical knowledge is based on know-how applicable to specific work	Best means of dealing with a specific customer. Cognitive: individual's belief on cause-effect relationships. Technical: surgery skills.
Explicit	Articulated, generalized knowledge.	Knowledge of major customers in a region
Individual	Created by and inherent in the individual	Insights gained from completed project
Social	Created by and inherent in collective actions of a group	Norms for inter-group communication
Declarative	Know-about	What drug is appropriate for an illness
Procedural	Know-how	How to administer a particular drug
Causal	Know-why	Understanding why the drug works
Conditional	Know-when	Understanding when to prescribe the drug
Relational	Know-with	Understanding how the drug interacts with other drugs
Pragmatic	Useful knowledge for an organization	Best practices, business frameworks, project experiences, engineering drawings, market reports

Organizational knowledge can be managed according to different strategies (Von Krogh et al., 2001): organizations can leverage their knowledge throughout the organization; expand their knowledge further based on existing expertise; appropriate knowledge from partners and other organizations; or develop

completely new expertise by probing new technologies or markets. Alavi & Leidner (2001) suggest high-level processes, namely "Knowledge Creation", "Knowledge Storage and Retrieval", "Knowledge Transfer", "Knowledge Application", to put into operation the chosen knowledge management strategy. The proposals of the thesis, derived from the goals (stated in the Section 1.4), can promote and enable the high-level knowledge management processes, namely :

- In "Knowledge Creation", by introducing, discuss, and reshape business process models coming from reference models;
- In "Knowledge Storage and Retrieval", by providing central repositories to explore procedural knowledge;
- In "Knowledge Transfer", by providing explicit and documented knowledge;
- In "Knowledge Application", by automating the execution of models that explicit state the procedural knowledge of the organization.

The use of the same human-readable model to cross the life-cycle of the software implementation of business process increases an organization awareness of what is running inside the business software system, eliminates the need of the collaborators to learn different models depicting the same business reality, and positively contributes to avoid the knowledge lost when model transformations performed. This way, we think that it is possible to create an organization more focused on the contents of the business processes and that do not wastes the major part of its resources to manage the environment of business processes.

To better support the implementations of business processes, organizations use business software systems. Business software systems are generic software systems that implement business process designs to enact and manage operational business processes (van der Aalst et al., 2003). One key characteristic of those software systems is the language used to model the business processes. A project team can use distinct selection criteria to pick an adequate language based on the expressiveness of the language, on the skills existing inside the organization to deal with a certain language, and on the adequacy of the language to the business environment (Santos et al., 2013).

2.2. Business Organizations

A business organization is an entity formed for the purpose of carrying on commercial enterprise. Such an organization is predicated on systems of law governing contract and exchange, property rights, and incorporation (Encyclopaedia Britannica, 2010).

Business organizations do not stand alone. They are part of constellations of organizations. It is a key goal to combine their internal business processes with the processes of their suppliers, customers, and partners. Higgins (2005) suggests integration rules, like to standardize organizational processes and tools, establish a coherent organizational structure, or optimize the requirements and change
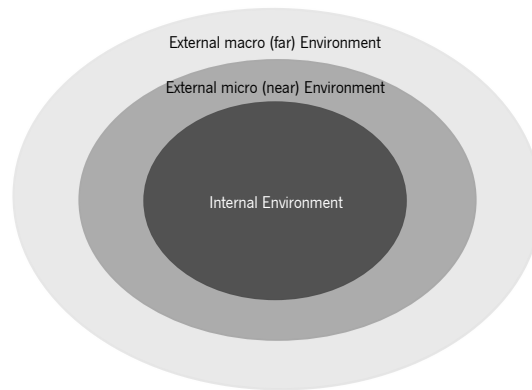
Figure 2.1.: Environments of organizations (adapted from (Campbell et al., 2002)).

management processes. Additionally, business organizations live inside some business environment where external and internal constraints shape their internal structures and behaviors. Since business organizations are not freestanding entities, they must operate obeying to constraints imposed by various forces, such as technological imperatives or the public interest. Due to that, business organizations are also social systems (Kalleberg & Berg, 1987).

2.2.1. Strategy of Organizations

As pointed by Kay (1995), the concepts of industry and market are distinct and should not be confused when defining a business strategy. Industries produce services or products that markets consume. Thus, we place business organizations inside some industry or set of industries. The external environments surrounding an organization have a decisive impact on the structure and behavior of an organization. Campbell et al. (2002) state that external environments can be divided into two layers: macro and microenvironment (see Figure 2.1). External macroenvironment may have impacts from socio-demographic, technological, economical, legislative, natural environment, or even political factors. The macroenvironment is generally out of the influence of a single organization, but may influence decisively the inner microenvironment. The industry, to which the organization belongs, and the market, where it sells its products or services, both compose the microenvironment of an organization.

The ability to extract better results from a given market, or to foreseen its behavior, are probably the most important success factors for an organization, either in the short, mid, or long term time frames. To cope with external environments, organizations must shape themselves in such a way that they overcome external and internal constraints with success. For that, organizations draw strategies to achieve their business goals. According to Chaffee (1985), three different business strategy models exist:

1. Linear Strategy, focused on the planning. This model has its basis on the definition of strategy by Chandler (1962): strategy is the determination of the basic long-term goals of an enterprise, and the adoption of courses of action and the allocation of resources necessary for carrying out

these goals. Usually, this model is associated with strategic planning. This model is methodical, directed, and relates with sequential planning activities. Thus, strategy consists of integrated decisions, actions, or plans that set organizational goals. To reach these goals, organizations change their interfaces with the surrounding business environment, like by changing their product portfolio, by moving into new industries and markets, or by reformulate themselves;

2. Adaptive Strategy, based on the definition provided by Hofer (1973), is concerned with the definition of a viable match between the opportunities and risks present in the external environment and the organization's capabilities and resources. In this model, the organization continuously monitors external and internal conditions in order to properly address changes in its internal structure or to create a change in the external environment, in order to align external environment opportunities and risks with internal capabilities and resources;

3. Interpretative Strategy, characterized by a social perspective of an organization. In this model, free will individuals contribute to the organization strategy by creating some kind of social contract by believing and acting in a way that will bring benefits for the organization. The strategy is composed of orienting metaphors constructed to guide individual attitudes of organizational stakeholders.

More recently, the Blue Ocean strategy model (Kim & Mauborgne, 2005) created a breakthrough with former strategy models. In the Blue Ocean strategy approach, organizations focus on creating new markets for their products or services instead of focusing on understanding the current markets and fighting the competitors. Within this approach, Red Oceans represent all the known industries and markets where boundaries and competitive rules exist. In Red Oceans, organizations try to outperform their competitors to obtain a greater share of an already existing demand. By opposition, Blue Oceans represent all the industries and markets not yet created. With the Blue Ocean approach, the organizations expect high revenues due to the novelty of the markets and because there is no or little competition. One success story with this approach is Cirque du Soleil. They created a new circus market, targeted for adults, based on a transformation of the traditional circus acts, and on the presentation of different shows to allow the same audience to assist more than on act in a short period.

The relevance of business strategy comes from the fact that organizations, in order to produce products or services, must take some set of actions. As pointed by Dixit & Nalebuff (1991), not all acts and decisions are made isolated. Either in personal or business environment, all our decisions interact with other decisions made by our surrounding environment. This interaction has an important effect on the thinking and on the actions of an organization. For that, a branch of applied mathematics, named Game Theory (von Neumann & Morgenstern, 1944), is the basis for many of the specific business strategies.

Independently of the strategy model adopted, an organization need to change its internal environment in order to deal with the external business environments. Usually, in process-oriented organizations, a strategic management process materializes some form of business strategy.

To transform properly the strategy into operations, and to maintain a continuous link between both, organizations need a defined and structured approach. Among others, Davenport & Short (1990), and Kaplan & Norton (2008) propose a closed-loop management system, containing an integrated set of processes and tools that an organization can use to develop its strategy, translate it into operational actions, and monitor and improve the effectiveness of both. Five steps compose the management system of Kaplan and Norton:

1. Develop the Strategy. Even prior to the strategy definition, managers need to define the mission (company's purpose), vision (aspiration for future results), and values (guide to the actions of the organization). Then, managers perform a strategic analysis, including political, economic, social, technological, environmental, and legal factors. Finally, they formulate the strategy by stating it and planning how the company proposes to achieve it;

2. Translate the Strategy. In this step, managers need to translate the strategy into objectives and measures that are after clearly communicated to all units and employees;

3. Plan Operations, by improving important processes, develop sales plan, plan the needed capacity resources, and plan the budget;

4. Monitor and Learn, by conducting strategy and operational reviews;

5. Test and Adapt the Strategy, because assumptions underlying strategy are often flawed or obsolete.

The governance of the business processes after their deployment into the business reality should be resilient. For that, a management system needs to be in operation, for instance by recurring to an organizational change management process. A business process landscape containing this process is later presented in Section 2.2.3.

An information system must absorb the changes in the organization derived from the strategy and learned from the operations. For that, it is crucial that a project team understands the business strategy and its operationalization, in order to adapt properly the information and software systems and to design them recurring to technologies that backup the business strategy.

2.2.2. Internal Configuration of Organizations

Porter (1980) states that to win a competitive advantage in any market, a firm needs to be able to deliver a given set of customer benefits at lower costs than competitors, or provide customers with a bundle of benefits its rivals cannot match.

To achieve these goals, organizations need to focus their activities on customers, and need to increase their ability to innovate continuously.

This demanding and changing environment drives the call for reorganizations of the internal structure to keep the changing pace with external factors. Hamel (2000) points out that the fundamental challenge that organizations face is reinventing themselves and their industries not just in times of crisis,

but continually. In addition, the focus on customer requirements drives the need to change internal structures, focusing and organizing activities to answer properly to customer requirements. This also demands a proper business model.

Based on strategy, and on the identified strategic success factors, an organization needs to draw a business model. This model is a conceptual tool that contains a big set of elements and their relationships and allows expressing the business logic of a specific organization (Osterwalder et al., 2005). The business model also contains the architecture of the organization and its network of partners. Thus, managers need to conceive the internal structure of their organizations in such a way to provide value to customers. Due to the awareness of the business environment, the business model should be able to support changes, namely those on the internal structure of the organization.

Information gives organizations competitive advantages (Porter & Millar, 1985). Information technology has strategic significance to organizations from allowing computer-aided design in technology development to incorporating automation in warehouses (Porter & Millar, 1999). This way, when an organization organizes itself to deliver the best value to customers, a proper way to accomplish it is to look for value-added activities, as in a value-chain analysis (see Figure 2.2).

According to Porter (1985), a value-chain is a chain of activities. Products or services pass through activities to gain value. The value-chain categorizes the generic value-adding activities of an organization. As shown in Figure 2.2, the primary activities of an organization having a supply-chain include: inbound logistics, operations, outbound logistics, marketing and sales, and services. The support activities include administrative infrastructure management, human resource management, R&D, and procurement. A value-chain identifies the costs and value drivers for each value activity.
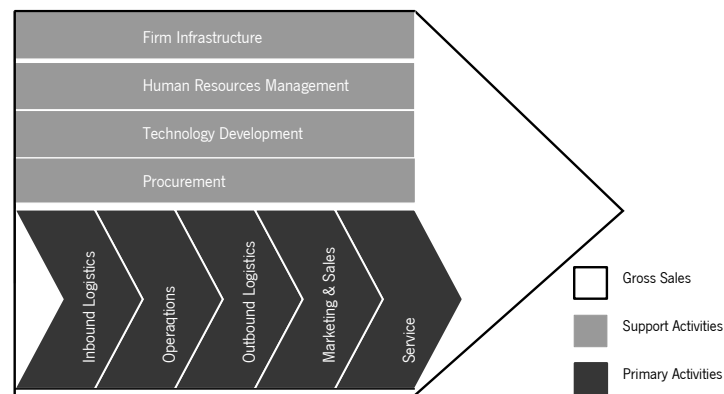
Figure 2.2.: Value-chain (adapted from (Porter & Millar, 1985)).

The value-chain framework is an analysis tool for strategic planning. Its ultimate goal is to maximize value creation while minimizing costs.

If we use a traditional hierarchical organization chart to materialize the value-chain, then problems and indecisions start to arise from the covered area of primary and support activities (e.g. human resources (HR) management would be a responsibility only for the HR department, instead of being a shared responsibility for HR and all other departments).

The explicit existence and management of business processes in an organization is a suitable approach to materialize a value-chain.

2.2.3. Process-Oriented Organizations and Management Approaches

The main idea behind process orientation is to drive an organization into value-added activities focusing on client's requirements and needs (Hammer & Champy, 1993). In a more traditional arrangement, organizations are composed of departments, each one with concrete, but separated, responsibilities and tasks. This splitting may lead into an excessive focus on the departmental goals and tasks, and, this way, missing the most important business driver: the client. Department-oriented organizations may face problems when they need to deploy internally the requirements of the clients (e.g. for the delivery quantities, the schedules, or the dates). For instance, the organization may deploy a goal for "delivery quantity" into the production department; "delivery schedule fulfillment" into the logistics department, and "delivery quality" into the quality department. This splitted deployment of goals causes internal conflicts, like when the logistics department wants to deliver at a certain date some quantity of products that may face a quality risk. Therefore, this internal conflict will absorb precious resources causing the organization to spend energy on internal discussions instead of converge into the best solution for the client. In a process-oriented organization, it is likely that a process owner receives all the client requirements. Since he owns all the goals, it is much simpler to derive the best solution for the client, particularly when restrictions are present.

2.2.3.1. Process-Oriented Organizations

Several definitions for business processes can be found:

1. Davenport (1993) defines a business process as "a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action". Processes are as structures by which an organization does what is necessary to create value for its customers. For that, an important measure of a process is the customer satisfaction with the output of the process.

2. for Hammer & Champy (1993), a business process is "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer".

3. the definition of Rummler & Brache (1995) is "a series of steps designed to produce a product or service. Most processes (...) are cross-functional, spanning the white space between the boxes on the organization chart. Some processes result in a product or service received by an organization's external customer. We call these primary processes. Other processes produce products that are invisible to the external customer but essential to the effective management of the business. We call these support processes".

4. the definition of Johansson et al. (1993) is "a set of linked activities that take an input and

transform it to create an output. Ideally, the transformation that occurs in the process should add value to the input and create an output that is more useful and effective to the recipient either upstream or downstream".

Common to all business processes definitions are action-related elements (either called steps or activities) that start with some input and produce an output with value for the client. The focus on the client requirements is also a common characteristic of the definitions, making business processes an adequate model to cope with new challenges that daily occur during the normal business operation of an organization.

The ability to cope with incomplete and inconsistent requirements from clients, and also with their continuous change, it is better handled inside a process-oriented organization than in a department-oriented organization, because of the continuous focus of process-oriented organizations on the clients' needs (normally expressed in the higher percentage of internal resources allocated to directly satisfying the client).

In a process-oriented organization, the management designs and coordinates the business processes in order to eliminate internal barriers (Hammer & Champy, 1993). The existence of departments is one of the main causes for the existence of those barriers. Hierarchical departmental structures are as a slice-in-time view of responsibilities and reporting relationships (Davenport, 1993).
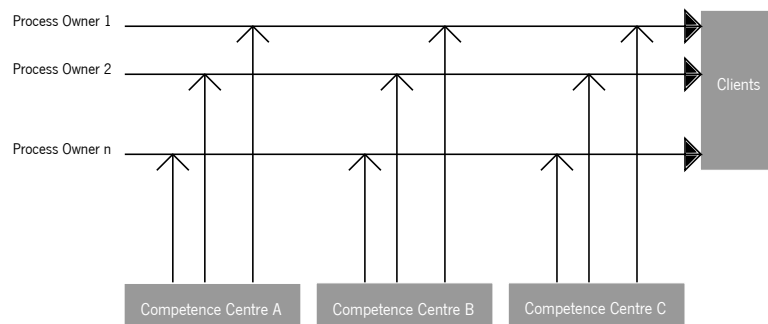
Figure 2.3.: Process-oriented organizations (adapted from (Hammer, 1997)).

As showed in Figure 2.3, there are no departments inside a pure process-oriented organization. Instead, the organization embodies competence centers to provide experts with skills that can contribute to execute properly a business process. Accordingly, the organization designs processes in such a way to bring the best possible value for clients. Competence centers are nucleus where the managers have the tasks to organize the human resources (e.g. decide the appointment for the human resources in his competence center for each instance of a business process) and to train them in the specialized tasks related with the expertise of the competence center. Processes are coordinated sets of activities where all the needed experts, managed by a process owner, collaborate in order to provide value-added to the client.

Having an organization with a business process-oriented approach is not a guarantee of business success. Business processes inside organizations should reflect and support the strategy of the orga-

Figure 2.4.: A typical cross-functional business process (adapted from (Davenport, 1993)).

nization. The organization should design business processes according to the clients' needs, should appoint adequate process owners, and should staff properly the teams.

The process owners should quantify and regularly check the indicators of their processes. Process owners must redesign the business processes when the current values do not match the process goals.

Organizations should also have the ability to redefine their process landscapes according new strategies or new clients' needs. A process landscape is not a static model, and for that, the organization must regularly review it.

Due to all the previously mentioned factors, the design, the control, and the improvement of the business process landscape of an organization can only be efficient and useful if exists a business process model defined, managed, and with quantifiable indicators for the organization.

As depicted in Figure 2.4, a typical business process is cross-functional. The business process, "New Product Development", organizing the analysis of competitors and market research inputs tasks, embodies activities belonging to three distinct functional areas (Research and Development, Marketing, and Manufacturing) in order to create, as output, a new product or a new prototype.

The structure of a business process model is crucial to the success of an organization. The modeling of processes in organizations is the most vital design element in the ERPs (Dalal et al., 2004). An organization should also pay attention to the amount and to the shape of all business processes. Business processes do not exist alone inside an organization. Therefore, the organization must design a global environment, the process landscape, to manage its business processes. A process landscape is a set of hierarchically structured business process models, related via interfaces (Gruhn & Wellen, 2001).

According to Ould (1995), organizations can embody two types of processes:

1. The sort that starts when necessary and finishes sometime in the future;

2. The sort that is constantly running.

An organization should consider this time-based classification when designing the process landscape and when staffing the processes. After a proper process landscape is set up, the execution of well-defined business processes in not trivial. Immature organizations execute business processes generally

by improvisation, done by practitioners and managers as they perform the work. Even when a process is well defined, it may happen that practitioners and managers do not rigorously follow it.

An immature organization reacts. Managers of immature organizations usually focus their attention on solving immediate crises - the "fire fighting" approach. Such organizations may not achieve their global goals for budget, schedule, and service level agreements because the managers focus on their own goals, based on their partial views. When a customer imposes hard deadlines, such immature organizations often compromise the product and service functionality and quality to meet the schedule or throughput expectations (OMG, 2008a).

2.2.3.2. Management Approaches for Process-Oriented Organizations

In the last decades, the business conditions pushed organizations into markets where short product life-cycles and rapidly changing technologies are key factors for their survival. Organizations carried out efforts to cope with the stakeholders expectations. These setting lead organizations to rationalize their activities, increase the level of automation, improve productivity, increase flexibility, decrease costs, or reduce internal headcounts. More radical approaches, like BPR (Hammer & Champy, 1993) were proposed, where the focus is to use computers as support for redesigned business processes, instead of only use computers to support automation. Automation efforts may conduct organizations into the trap of automating the wastes, rather than eradicating them.

BPR approach proposes to organizations some principles to apply into the design of their internal structures (Hammer, 1990):

- organize the work around their outcomes, not around tasks;
- have those who use the output of the process perform the process, suggesting that some tasks can be performed by all organization members instead of by only a specialized set of workers (e.g. buying a pencil does not need a purchasing expert);
- include information-processing work into the real work that produces the information. This principle advises to eliminate departments of just information processing and handle the information processing to the ones who perform the real work;
- treat geographically dispersed resources as thought they were centralized, facilitated by current communication technology available;
- link parallel activities instead of integrating their results, promoting parallel work to speed up processes execution;
- put the decision point where the work is performed, and build control into the process, stating that monitoring and controlling activities can be performed by the ones who execute the process;
- capture information once and at the source;

BPR efforts can lead organizations into radical changes and thus, if badly executed, can have negative impacts on headcounts. For that, Hammer (1997) putted effort to explain better the concept.

More human-aware approaches were also proposed for the business process management, by managing changes in a more holistic approach considering technological, human, and organizational dimensions (Davenport, 1993), or, in TQM, by linking the root cause of the failure of organizations to top management "It is no longer socially acceptable to dump employees on the heap of unemployed. Loss of market, and resulting unemployment, are not foreordained. They are not inevitable. They are man-made. The basic cause of sickness in American industry and resulting unemployment is failure of top management to manage" (Deming, 1982).

Management models based on TQM, like the EFQM model (EFQM, 2001) or MBNQA (Steeples, 1993), provide frameworks for organizations to reach performance at excellence level based on business processes. Other management approaches focuses on the reduction of wastes and improving quality, like the JIT, where the inventory is intended to be reduced to the minimum, closely related with the TPS (Shingo et al., 1989). In TPS, the products are only manufactured when they are ordered by clients, production is organized in small lots triggering the need for quick change-over in production tools, and productions steps are pulled backwards in the supply-chain.

Another business process management approach started in the beginning of the 1980s, Six Sigma, aims to improve the quality of the outputs of processes by identifying and removing the causes of defects (errors) and minimizing the variability in the business processes (Antony, 2004).

Either by incorporating continuous improvement approaches, by introducing radical changes, or by doing both at the same time, the capacity of organizations to change and adapt to the surrounding business environment is a key success factor for their survival. To introduce modifications into organizations, BPR emphasizes the role of IT as a vehicle to introduce radical changes, either by allowing new organizational formats (e.g. distributed organizations) or by promoting collaboration within and between organizations (Hammer & Champy, 1993).

Process-oriented approaches enhance the efficiency of organizations to be more efficient. Smith & Fingar (2002) state: "A process-managed enterprise makes agile course corrections, embeds Six Sigma quality, and reduces cumulative costs across the value-chain. It pursues strategic initiatives with confidence, including mergers, consolidation, alliances, acquisitions, outsourcing and global expansion. Process management is the only way to achieve these objectives with transparency, management control, and accountability".

The BPM approach is intended to align the business processes with the strategic objectives and the customers' needs, but requires a change in a company's emphasis from functional to process orientation (Lee & Dale, 1998).

Models of business processes are the basis for BPM. Process stakeholders can communicate about process structure, content, and possible improvements. BPM includes areas such as business process modeling, formal models, analysis and verification of business processes, process mining and workflow management. BPM has case studies from various domains including medicine, technology, and logistics (Pernici & Weske, 2006). According to Zairi (1997), BPM is a structured approach to analyze and
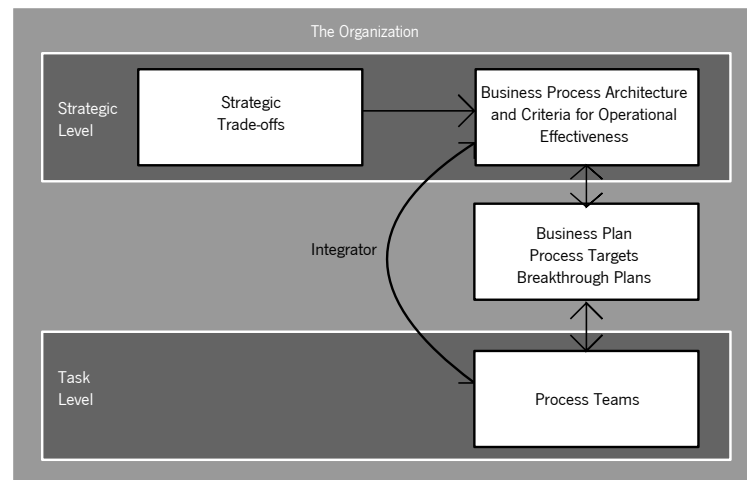
Figure 2.5.: Organizational framework for BPM - generic levels of activity (adapted from (Armistead et al., 1999)).

continually improve fundamental activities such as manufacturing, marketing, communications and other major elements of a company's operations. Elzinga et al. (1995) define BPM as a systematic, structured approach to analyze, improve, control, and manage processes with the aim of improving the quality of products and services. Common to most BPM definitions are the cross-functional characteristics of the business processes that BPM handles, and the structured and analytical nature of BPM to manage business processes.

An organizational framework to explicate the generic levels of activities related with BPM is presented in Figure 2.5. According to Armistead et al. (1999), the integration link between the strategic organizational level and the task level (daily activities) materializes throughout business planning, definition and measurement of process targets, and improvement actions. The integration link between the abstract strategic level and the operational tasks level is important to guarantee a proper alignment between business intentions and their practical implementations.

Several themes can be associated with BPM (Armistead et al., 1999):

- how strategy is developed, choices are made, and plans for deployment are generated within the goals and targets of the organization;
- how BPM is incorporated into the design of the organization, namely the empowerment given to the business processes;
- the maximization of market value-chain recurring to processes descriptions;
- the performance management;
- the organizational coordination, both internally and with the surrounding environment;
- the organizational learning and the knowledge management;
- the organizational culture impact on processes and vice versa;

These multifaceted themes promote the need to install a proper support for the BPM activities inside an organization. For that, organizations need BPM systems.

A BPM supporting system is a generic software system driven by explicit business process designs to enact and manage operational business processes. A BPM system should also allow the modification of the processes it supports, the deletion of processes, or the inclusion of new processes (van der Aalst, 2003a).

In the mid 1990s, organizations considered especially workflow management systems (Georgakopoulos et al., 1995) as significant contributors to improve the efficiency of the processes. Also, ERP vendors, such as SAP, positioned their solutions as vehicles for business process redesign and improvement. Because business processes cross the boundaries of organizations, and because organizations have a global presence, ubiquitous technologies are increasingly necessary to support business processes.

Currently, BPM approach is receiving major attention from business administration, to improve the operations in the organizations, and from computer science communities, to investigate the structural properties of the business processes and to investigate robust and scalable software systems to support the integration of business processes into complex information systems (Weske, 2007). As stated by Greenfield & Short (2004), each business process can be implemented as a web service.

Independently of the management approach used for business processes, there is a need to assess the compliance with some reference structure (e.g. like with the strategy of the organization, or with some quality standard), and the differences between the models of the business processes and the real business processes practiced. Many frameworks exist to assess the maturity of the business processes (see Roseman (Rosemann & de Bruin, 2005), BPMM (Lee et al., 2007), Gartner's Maturity Model (Melenovsky & Sinur, 2006), or even EFQM (EFQM, 2010)), being the most noticeable the SEI Capability Maturity Model Integration (CMMI) (Ahern et al., 2008). CMMI appraisal schema evaluates organizations, in its discrete evaluation, in five levels, namely: Initial, Repeatable, Defined, Managed, and Optimized. Advantages of using maturity models to assess implementation of the business processes are the creation of a concrete conscience within the organization about the current situation, the avoidance of too optimistic endeavors to come from not-managed processes into top levels of maturity, and, thus, resulting in failures.

2.2.4. Business Process Reference Models

Models are abstractions of reality. They represent views of reality and, thus, they allow its users to focus only on the needed characteristics of the modeled reality. Models can minimize the complexity of reality they represent by including representations understandable and more easily handled by humans or machines. Models are also used to transfer pieces of knowledge between reality domains, namely from one organization into another organization sharing some characteristic (e.g. both operating in the same market).

2.2.4.1. Definition and Characterization

Reference models are abstract models representing types of organizations. Reference models are abstract in the sense that they do not represent a particular organization. According to Scheer (1999), a reference model is a model which can serve as the starting point for the development of solutions based on concrete problems. Hars (1994) states that every reference model is a model which can be consulted for the development of other models.

Reference models for process-oriented organizations are reference models that promote and include some form of business process-oriented approach. BP-RMs, like the SCOR (Council, 2008a) or the Fernandes/Duarte reference model (presented in Figure 2.6), can include processes definitions, business roles allowed to execute business process, best practices to implement the reference model in a specific organization, or indicators to track the performance of the business processes.
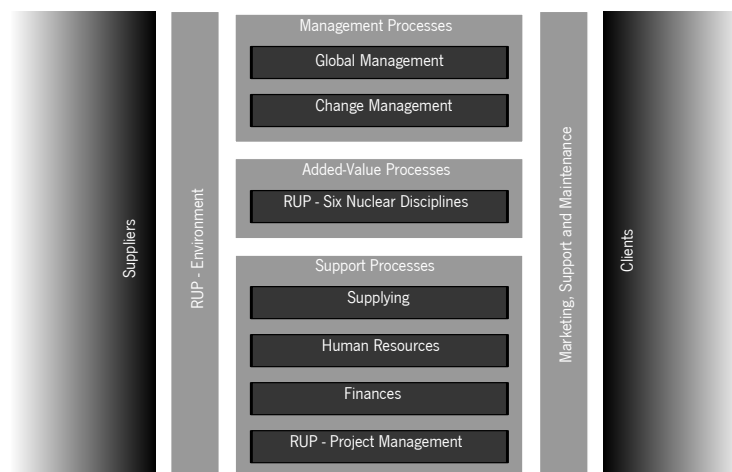
Figure 2.6.: The Fernandes/Duarte reference model: A BP-RM for an organization that develops software, based on RUP disciplines (adapted from (Duarte et al., 2007)).

Organizations can use BP-RMs to derive concrete models of their business processes. According to Thomas (2006), a reference model, in the sense of an initial conceptual approach, is a point of reference for the development of specific models because it represents a category of applications. From this definition, we may infer that reference models are not universally applicable. Some sort of restriction on their usage should be stated, like the vertical market where the organization operates (e.g. insurance, or banking industries), or some set of traversal organizational function (e.g. IT management, or logistics).

BP-RMs can also include general indicators for each process to measure the business process throughput (Marshall, 2000), namely:

- productivity: relation between the output value and the input cost;
- added value: the output value minus the input cost;
- cycle time: difference between the initial time and the end time collected from each execution

of an instance of a business process;

- queue size: average size of the queue containing the requests to execute the processes;

- quality indexes: percentage of the number of defects collected out of the business process instances executed.

The intrinsic properties of BP-RMs describe and characterize the BP-RM. Fettke et al. (2005) propose a framework to describe reference models, either process-oriented or not (see Table 2.2). Using such standardization to describe and characterize reference models eases the work to pick an adequate reference model for a specific organization.

Table 2.2.: Criteria to describe reference models (adapted from (Fettke et al., 2005)).

Element	Group	Criteria
Reference Model	Identification	Number
		Name
		Primary/Secondary Literature
	General Characterization	Origin
		Responsibility for Modeling
		Access
		Tool Support
	Construction	Domain
		Modeling Language(s)
		Modeling Framework
		Size
		Construction Method
		Evaluation
	Application	Application Method(s)
		Reuse and Customization
		Use Case(s)

A catalog of known reference models together with their characterizations, according to the Table 2.2, can be found in the catalog provided by Loos & Fettke (2010). Among many others, the characterization for SCOR and for the Fernandes/Duarte reference model can be consulted. Appendix Characterization of Reference Models provides the detailed characterization of both reference models.

The existence of BP-RMs helps the software engineering activities, primarily the ones related with business modeling, software requirements, and software design. The model of an organization, as well as the management of the model, are shared work areas where software engineers and business experts can discuss the present and can design the future shape of the organization. This common basis can avoid discrepancies between the organizational requirements and the characteristics of the business software systems.

According to Scheer et al. (2007), reference models are developed for the aim of using the CIM concept. This perception emphasizes the tight connection between reference models and software systems. Indeed, if a reference model is properly picked from a catalog, like the one from Loos & Fettke (2010), and then is instantiated into a model for a specific organization, then software engineers can use the model as a major input for the development of information and software systems. A methodology to help carry out this steps is later on proposed in Chapter 7.

During software development, the project team needs to compare the characteristics of the processes in the BP-RMs with those needed for the organization. The project team needs the results of such comparison to verify if the business model representing the organization is ready to allow the project to start.

2.2.4.2. The SCOR Reference Model

SCOR is a BP-RM for the supply-chain operations. The SCOR has been developed to describe the business activities associated with all phases of satisfying a customer's demand (Council, 2008a). Business experts can use SCOR across the boundaries of vertical markets. SCOR is also used as the basis for the Action presented in Section 6.3.

SCOR provides the standard language to describe the performance, configuration, activities, practices, and workforce assets of a supply-chain. The SCOR consists of four key pillars (Council, 2010):

1. performance metrics, to help to describing the performance of the supply-chain;

2. processes, to help explaining how the supply-chain is configured (what activities are taking place);

3. practices, as they are unique ways to configure supply-chain processes;

4. people, to assess the needs, availability, and gaps of skills in the supply-chain workforce.

Each pillar has a hierarchy of elements that may have connections with other pillars (e.g. metrics link with processes).

The top three process levels (see Figure 2.7) are the focus of SCOR. SCOR does not attempt to prescribe completely how a particular organization should conduct its business or tailor its systems and information flow. Every organization that implements supply-chain improvements using SCOR will need to extend the SCOR model, at least to the Implementation Level (see Figure 2.7), using organization-specific processes, systems, and practices (Council, 2008b). Below the Process Element Level, SCOR uses classic hierarchical process decomposition to describe each process. This way, at the forth level, tasks describe each process element. At the fifth level, activities describe each task. At level six, a sequence of sentences or diagrams, like UML activity diagrams, describe each activity.

At its top level, SCOR has five core management processes:
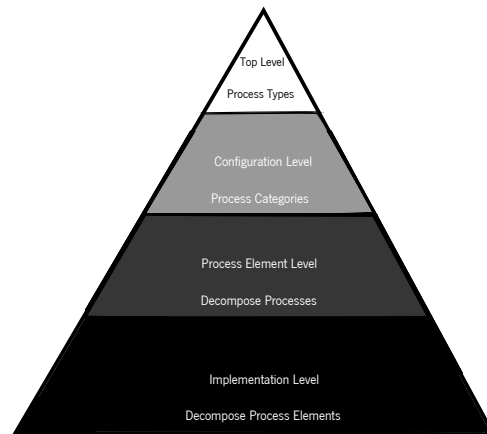
Figure 2.7.: Levels of process details in SCOR.

- Plan, to create a course of actions based on demand and supply and to meet sourcing, production, and delivery requirements;

- Source, to support demand by procurement of goods and services;

- Make, to transform materials until they reach the final product stage according to demand;

- Deliver, to provide to customers finished products according demand;

- Return, to give back materials to suppliers or to receive finish products in post-delivery customer support.

2.3. Information Systems

Traditionally, two perspectives define information systems: one related with their functions, the other related with their structure. According to the functional perspective, an information system is a technologically implemented medium for the purpose of recording, storing, and disseminating linguistic expressions as well as for the supporting of inference making. From a structural perspective, an information system consists of a collection of people, processes, data, models, technology, and partly formalized language, forming a cohesive structure, which serves some organizational purpose or function (Hirschheim et al., 1995). According to the structural perspective, an information system can have the following perspectives:

- people, as the users of the information system. Of course, an information system affects other people than the users, but in an indirect way;

- data, as either the information represented in a suitable form for processing by a computer or in some non-IT representation (e.g. a sheet of paper or some sign to indicate the status of a real world activity);

- processes, as the operations performed by the users either on data or not;

- models, as abstractions of real world "things";

- technology, as a set of systematic techniques enhancing the remaining components of the information system.

Both the functional and the structural perspectives embody the notion that human activities are included in the concept of information systems. This way, an information system has a social context.

At the present, the solutions to handle data, processes, and models, all understood as information systems components, as well as the actions associated with all of these components, are IT-related. The amount of transferred or stored data, the speed and availability of communication means needed in long distance for human interactions turn unfeasible to implement an information system without any IT support, and consequently without any software system to implement those components and rules to mimic the human decisions.

Currently, many organizations define as areas to improvement the pure manual actions contributing to a business process occurring in some information system. As stated by Hammer & Champy (1993), together with the introduction of business processes in organizations comes the opportunity to implement new supporting software systems. This approach converts IT in one of the major BPR supporting factors. Fowler (2003) characterizes information systems by characteristics, such as having persistent and large amounts of data, concurrent accesses with many user interface screens, and the need to integrate with other enterprise applications.

Some embedded IT systems are designed to avoid direct human interaction, as the ones deciding in real-time if the wheels of a car should be blocked or not. Unlike these IT systems, an information system loses its main purpose if constant human interaction is not present. It is pointless to store data, to analyze it, or to distribute it if humans do not do anything with it. As stated by Curtis et al. (1992), process representation becomes a vital issue in redesigning work and allocating responsibilities between humans and computers. Vasconcelos et al. (2001) state that having a unique and standard language to describe different aspects of business is fundamental, in order to create a common ground for discussing both business and their supporting systems (i.e. a formal way to describe business goals, processes, information systems, and the dependencies between them).

Organizations should avoid using multiple languages and models to describe related fields like business and software domains. Delen et al. (2005) present an integrated modeling approach to understand an organization as a whole, achieved by integrating the models of the organization with the supporting ERP. During information systems project time, it is also relevant to consider which development process model is more adequate for a specific project. Since most information systems are software-based, it seems adequate to use software development processes to develop information systems. Nevertheless, the project team should pay special attention to choose a software development process where both real world business models and human interaction are considered, like the RUP (Jacobson et al., 1999).

From all the possible analysis dimensions of information systems, in this thesis, we focus in the dimensions related with software.

2.3.1. Information Systems Concepts

Information systems are artifacts to study in their own right, independently of the characteristics of their users, the organizations in which they are employed, or the technologies used to implement them. Information systems have static and dynamic properties (Wand & Weber, 1990). Other terms for "information systems" include "enterprise application" or, for those with a long memory, "data processing" (Fowler, 2003). Information systems are mediators between business frameworks and information technology and can be characterized by using in-depth system theoretical attributes. For example, the complexity of information systems is as a significant system theoretical attribute. This complexity can be attributed to the fact that information systems work on a business, as well as on a technical level (Thomas, 2006). In the "IS 2010: curriculum guidelines for undergraduate degree programs in information systems" (Topi et al., 2010), issued by Association for Computing Machinery (ACM) and the Association for Information Systems (AIS), are stated some reasons to upgrade from last (2002) version related with major contextual changes regarding information systems and, thus, providing hints on their current shape and environment, namely:

1. complex globally distributed information systems development;

2. web technologies and development. Mature modeling, management, and development platforms for the Web environment have become a core part of IS development;

3. emergence of a new architectural paradigm. Service-oriented architecture, web services, software-as-a-service, and cloud computing are important elements in the new way of organizing the fundamental architecture for software systems that is gradually becoming the dominant paradigm of organizational computing;

4. ERP and packaged software. Information systems and business processes have become closely integrated. Increasingly, core infrastructure applications are based on large-scale enterprise systems so that the focus has shifted from development to configuration and implementation;

5. ubiquitous mobile computing;

6. IT control and infrastructure frameworks. Frameworks and standards such as COBIT, ITIL, and ISO 17799 have become very important sources of guidance for IT/Information Systems practices in organizations through governance models.

Developers should pay special attention to eventual conceptual dissonances between business processes and data, and to the intrinsic complexity of the business logic (Fowler, 2003). For that, information systems should have a sufficient level of abstraction to embody concepts easily understood by humans, like invoice, production plan, or debit note. Nevertheless, this high degree of abstraction must not be an excuse not to embody lower level concepts, like the representation of an electrical signal occurring inside a business process. It is our opinion that information systems should be connected, for instance through a layered approach, into the real world events.

In Figure 2.8, we propose a framework to allow the checking of the generic differences (Delta x,y) that may exist between real world business processes (Px) and the software-implemented business processes (Sy). S1 and P1 models represent the current moment in time (as-is), before the occurrence of some organizational or software change project, and S2 and P2 models represent a desired situation occurring in a future moment in time (to-be).
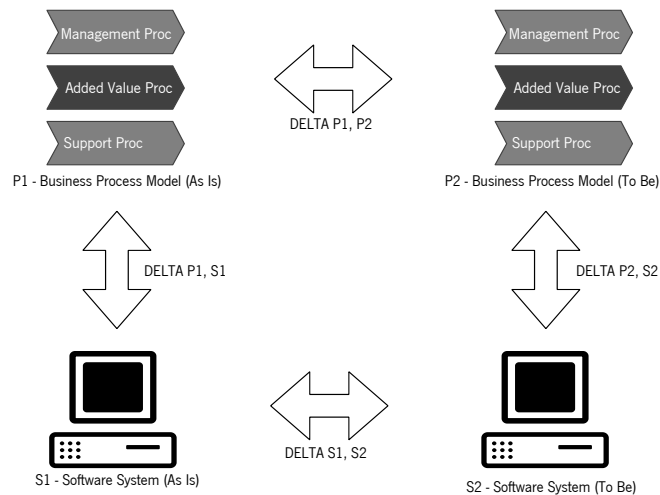
Figure 2.8.: Differences between business processes models and software systems, in as-is and to-be time frames.

Four possible sets of differences (represented by "Delta x,y" in Figure 2.8) may occur between real world and software-implemented business processes:

1. Delta P1, P2: occurs when the current and the future real world business processes have differences. This situation may occur when doing some business process improvement;

2. Delta S1, S2: occurs when the project team changes the software system from the current into the future situation, due to some requirement. Desirably, the transformation of S1 into S2 is supposed to occur simultaneously to the transformation of P1 into P2;

3. Delta P1, S1: relates with the differences between the real world and software-implemented process models at the present time. For this delta, two main causes are possible:

   a) inside the current model of business processes (P1) exists a larger number of processes than those implemented in software (S1), configuring a normal situation in the sense that real world business processes have a broader scope than their respective implementations in software systems. Nevertheless, this delta may also be caused by the software system not being supportive to the real world processes;

   b) inside the current software model (S1) exists a larger number of processes than inside the present business processes model (P1), which may occur due to an inadequately over-dimensioned software solution, but also due to an inappropriate low level of use of the software system by the organization, like when using only some parts of an ERP system;

43

4. Delta P2, S2: in a future time, the software-implemented business processes are different from the real world business processes. Two main causes are possible:

   a) inside the future business processes model (P2) exist more processes than inside the future software model (S2). This may occur in a normal situation, where an organization decides that not all new real world processes (P2) are to be modeled in the new software system (S2), but may also occur because the new software model is not able to provide all the needed business functions, as in the case of a software system without quality;

   b) inside the future software model (S2) exist more business processes than those existing inside the future business processes model (P2), being this a normal situation in the case of the organization knowing that the available software system should support future business expansion. It may also occur because the future software model (S2) is over-dimensioned.

From all the previously presented differences between real world business models and software models, a project team should pay special attention to transformations from a present business model into a new one without the required transformation from the present into the new software model. This situation will lead into an execution of the new business model (P2) based on an outdated software model (P1), causing misunderstandings inside the organization due to the different behaviors in the real world and in the software models, and perhaps creating serious problems with clients or suppliers. In addition, a transformation of the present into the future software model can only be justified if the present software model is inadequate to support the present business model, or if a technical change is undergoing.

Information systems need to embody a representation of the business they are supporting. Dale (2007) sows the roadmap from a major software vendor (SAP) to represent business content. The first period, named Reference Model, supported by the SAP R/3 tool, is retired. In this period, the focus was set on business engineering where graphical process flows describe business processes[1].

In the second period, the Implementation Content, supported by the SAP ERP tool, the focus is set to the implementation, and the notation of the business processes is swim lane diagrams to include the representation of the executing entities.

In the third and last period, the main paradigm is an enterprise service oriented architecture (SOA) based on a process component architecture. In this period, the focus is the SOA governance by model-based design recurring to repositories of enterprise services. In our opinion, such architecture demands sound services representing the parts of business processes and the orchestration mechanisms to compose business processes. This architecture has also the advantages of allowing highly interconnected constellations of components of business processes and the reuse, at runtime, of those components in diverse business processes. We later address the concerns expressed for the last period, together with additional ones, in the Section 6.3 and in the Chapter 7.

---

[1]SAP R/3 uses the EPC notation (Scheer, 2000) to represent business processes.

Table 2.3 represents the three generations of information systems. The information systems from the first generation started to focus on the automation of the activities of one department inside an organization.

The second generation, in our opinion the currently prevailing kind in use, focus on the support at operation time of an entire organization, perhaps spread across some sites.

Table 2.3.: Evolution of the focus of information systems engineering (adapted from (Siau, 2007)).

	First Generation	Second Generation	Third Generation
Scope	Single department	Single organization	Multiple organizations
Number of Sites	Single site	One or few sites	Multiple sites and multiple countries
Organizational Focus	Department level	Seamless integration within an organization	Seamless integration across multiple organizations
Technology Focus	Automate business processes, business processes efficiency	Organization-wide closed loop	Integration of supply-chain, e-commerce and e-market, mobility and ubiquity
Information Systems Engineering Methods and Technologies Focus	Design and development focus (e.g. flowchart)	Support analysis, design, and development (e.g. ER, DFD)	Unification of modeling methods and methodologies (e.g. UML and Unified Process)

The third generation focuses on multiple organizations, linking suppliers, customers, and partner organizations, guaranteeing a ubiquitous presence of the offered services to its users, and allowing processes to run seamlessly across multiple organizations. Siau (2007) provides an overview of the future decades of information system engineering. In this preview, two major scenarios are depicted for the third and subsequent generations of information systems. The first, the Idealistic scenario, depicts major revolutions on the extent of the automation in the analysis, design, implementation, and management of information systems. In addition, analysis and design of business applications can be done by end-users. In the second, the Pragmatic scenario, components, COTS systems, as well as metamodels, MDA, and SOA will become fundamental areas. Domain ontologies will be further developed. It is also pointed that some descendants of the Unified Process will be used.

In our thesis, we use the concepts of the Pragmatic scenario, but not forgetting about automation and the Idealistic scenario, despite its much harder conceptual and technological challenges.

2.3.2. Ontologies and Metamodels

Information modeling is a core vehicle to analyze, design, implement, and deploy information systems (Wand & Weber, 2002). The formal treatment of the semantics in information systems has always been an important and difficult issue, though often implicit or even tacit, in the analysis, design and implementation of such systems (Meersman, 1999).

As described by Uschold (2003), the specification of the semantics is based on questions related with explicitness, formality, and intention of use (machine or human). Based on these three questions, semantics has four distinct categories:

- implicit, like when humans talk about some concept;

- explicit but informal, like when humans write some textual description about some concept;

- formal and explicit for humans, like when some formal language is used to describe a concept but with the purpose of human use only;

- formal and explicit for machines, like when exists the possibility to automatically infer the meaning of some concept based on the already expressed semantics recurring to an inference engine.

According to Pidcock (2003), some forms to represent semantics are controlled vocabularies, taxonomies, thesauri, ontologies, and metamodels. A controlled vocabulary is a list of terms explicitly enumerated, like the glossary artifact in the RUP process. A taxonomy is a collection of controlled vocabulary where the terms are organized in a hierarchical structure, like the taxonomy to classify animals in natural sciences. A thesaurus is also a collection of a controlled vocabulary, but the relations between terms are not restricted to a hierarchy, like in the taxonomies, but can have another type of associations.

From all these representation options, one can derive that controlled vocabulary miss the relation between terms. Taxonomies and thesauri have those relations, either hierarchical or from another kind, but miss grammar rules to explicitly command how terms should be used.

Pidcock (2003) also provides definitions for ontologies and metamodels. A formal ontology is a controlled vocabulary expressed in an ontology representation language. The language should have a grammar for using vocabulary terms to express some meaning within a specified domain of interest. The grammar should also contain formal constraints to control the combination of the terms of the controlled vocabulary. A metamodel is an explicit model of the constructs and rules needed to build specific models, within a domain of interest. For that, a valid metamodel is an ontology, but not all the ontologies are modeled explicitly as metamodels.

The importance of the semantics in information systems comes from the need for an accurate communication (i.e. the exchange of meaning) between different software applications, between software applications and humans, and even between humans.

In the current thesis, the importance of semantics derives from the need to have a continuum of concepts from the business domain into the software domain. Additionally, and since one of the goals (SG7, see Section 1.4) is to have software solutions based on distinct software vendors, it is crucial the definition of proper semantics for each of the distinct and interchangeable software components composing the software solution. For that, as explained in Section 2.2.4, we use BP-RMs to have a common semantics base for the business and the software domains, and to act as a metamodel for a concrete business process model of a given organization.

2.3.2.1. Ontologies

The concepts of ontologies emerged in applied artificial intelligence (AI) as a way to share and to reuse the knowledge on knowledge-based systems. To support the sharing and reusing of formally represented knowledge between distinct AI systems, it is useful to define the common vocabulary representing the shared knowledge. According to Gruber (1993), an ontology is a specification of a representational vocabulary for a shared domain of discourse (definitions of classes, relations, functions, and other objects). According to the OMG (2009b), an ontology defines the common terms and concepts (meaning) used to describe and represent an area of knowledge. An ontology can range in expressiveness from a taxonomy (knowledge with minimal hierarchy or a parent/child structure), to a thesaurus (words and synonyms), to a conceptual model (with more complex knowledge), to a logical theory (with very rich, complex, consistent, and meaningful knowledge).

Developers may create ontologies for different reasons, like:

- to share a common understanding about the structure of information exchange between humans and software systems;
- to enable the reuse of domain knowledge;
- to make explicit domain assumptions;
- to separate the domain knowledge from the operational knowledge;
- to analyze the domain knowledge;

Also according to Gruber (1993), an ontology is an explicit specification of a conceptualization. In this perspective, unambiguous models that represent shared abstract aspects of the real world represent ontologies. These unambiguous models allow that well-defined semantic concepts can be interchanged between humans, between humans and software systems, and between distinct software systems.

Ontologies are descriptions of concepts and relationships that can exist for an agent or a community of agents (Staab & Studer, 2004). An ontology can also be seen as a formal and explicit description of concepts inside a domain of discourse (classes, sometimes also called concepts), properties of each concept describing various features and attributes of the concept (slots, sometimes called roles or properties), and restrictions on slots (facets, sometimes called role restrictions).

An ontology, together with a set of individual instances of classes, constitutes a knowledge base. In reality, there is a fine line where the ontology ends and the knowledge base begins (Noy et al., 2001).

Since ontologies define a common vocabulary for humans and software systems who need to share information in a domain, they may include machine-interpretable definitions of basic concepts in the domain and the relations among them. Ontologies are described recurring to languages. The Ontology Definition Metamodel from OMG (2009b) enumerates several metamodels for languages to define and to use ontologies, like:

- description logics (DL) (Baader et al., 2003), a formalism to represent knowledge;
- common logic (CL) (ISO/IEC, 2007), a first-order logical language intended for information ex-

Table 2.4.: A list of ontologies (adapted from (semanticweb.org, 2012)).

Ontology	Purpose	Language
Dublin Core	Vocabulary to describe documents	RDF
FOAF	Schema to describe persons and their social networks	OWL DL
Basic Geo Vocabulary	Vocabulary for representing latitude, longitude, and altitude	RDF Schema
vCard RDF	Vocabulary that corresponds to the vCard electronic business card profile	RDF
OpenGUID	System for establishing global identity for the semantic web	RDF Schema

change and transmission over an open network;

- resource description framework (RDF) (W3C, 2004), a framework for representing information in the Web;

- web ontology language (OWL) (W3C, 2009), an ontology language for the semantic web (Berners-Lee et al., 2001) (the semantic web is an initiative to allow the exchanging of content on the World Wide Web beyond the boundaries of applications and websites with formally defined meaning[2]). OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents;

- topic maps (TM) (Garshol et al., 2008), a technology for encoding knowledge and connecting this encoded knowledge to relevant information resources.

Additionally, the World Wide Web Consortium[3] defines the SPARQL (W3C, 2008), a query language for RDF.

Table 2.4 lists some currently available ontologies together with the languages used to describe those ontologies. In addition, ontologies for biomedical, engineering, mathematical, or finances are publicly available.

The use of ontologies can boost the search and the interchange of information across distinct software systems. Initiatives like the semantic web, which promotes that the information has an explicit meaning, allow that software can automatically discover, process, and integrate information. For that, it is beneficial that information systems embody ontologies of the domains where they operate. According to Pinto & Martins (2004), the usually accepted stages through which an ontology is built are:

- specification, where the scope and the purpose are identified;

- conceptualization, where a conceptual model is built to be check against the specification of the previous step;

- formalization, where the conceptual description is transformed into a formal model;

- implementation, where the formalized ontology is implemented in a knowledge representation language;

---

[2]http://semanticweb.org/wiki/Main_Page
[3]http://www.w3.org/standards/semanticweb/

- maintenance, where updates and corrections of the implemented ontology occur.

Approaches to use ontologies in information systems, in the case for supply-chain management, were already proposed by Chandra & Tumanyan (2007) as a way to, at runtime, separate the domain knowledge of one component from other components and delivering it upon request.

Most research focuses on the open standards of web services and semantic web but pays less attention to semantics-based enterprise modeling, where automated service discovery and service composition are difficult (Chu & Qian, 2007). Since ontologies are viewed as increasingly important tools for structuring domains of interests, proposals also exist for a reference ontology of business models (Andersson et al., 2006).

In the context of our thesis, ontologies can be used to describe the business content of the business processes for a given BP-RM. This way, software development activities would result in a better software product, due to the improved correctness of the concepts implemented. In addition, a project team could more easily achieve the substitution of a software component implementing a part of a business process by an ontological equivalent implementation, with the possibility to improve the non-functional requirements (e.g. performance).

2.3.2.2. Metamodels

Metamodels are models who make statements about models. A metamodel defines the constructs and their relationships for a given modeling language, as well as the constraints and modeling rules (Stahl & Völter, 2006). Metamodeling is a technique for defining modeling languages. It consists of defining a special model, the metamodel, using a modeling language designed specifically for metamodeling, which defines the language concepts and their relationships (Selic, 2011).

Model-driven engineering (MDE) (Kent, 2002) technologies offer an approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively (Schmidt, 2006). MDE technologies should combine:

- domain-specific modeling languages (DSML), to formalize the application structure, behavior, and requirements within a particular domain, such as a supply-chain. DSMLs are described using metamodels, which express the relations between the concepts in that particular domain;
- transformation engines and generators, which analyze aspects of models and synthesize artifacts, such as source code.

Early in projects, MDE tools deal with requirements expressed by models, they can perform model-checking and impose domain-specific constraints in order to obtain better software systems.

MDE is wider in scope than OMG model-driven architecture (MDA), because MDA does not address some aspects, as the dimensions of the modeling space other than those of the platform independent and the platform specific models. MDE combines process and analysis with architecture (Kent, 2002). The MDA standard from the OMG is just a specific incarnation of the MDE approach (Favre, 2004).

The MDA is a set of OMG standards that enables the specification of models and their transformation into other models and complete systems (Mellor et al., 2003).

The MDA defines an approach to IT system specifications that separates the specification of the system functionality from the specification of the implementation of that functionality on a specific technology platform. To this end, the MDA defines an architecture for models that provides a set of guidelines for structuring specifications expressed as models (Miller et al., 2001).
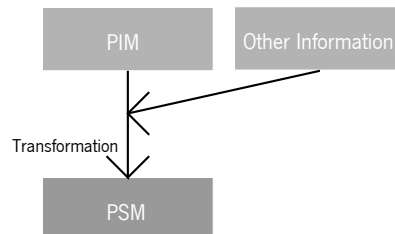
Figure 2.9.: Model transformation from PIM into PSM in MDA (adapted from (Miller et al., 2003)).

MDA has two basic concepts related with platform models (see Figure 2.9):

- platform independent model (PIM), which is a view of a system from a platform independent viewpoint. A PIM exhibits a specified degree of platform independence to be suitable for use with a number of different platforms of similar type;

- platform specific model (PSM), which combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

In MDA (see Figure 2.9), the PIM and other information are combined and transformed in order to produce a PSM. Both the PIM and the PSM have distinct metamodels and transformations can occur between their metamodels or between the platform independent and platform specific models. According to Meservy & Fenstermacher (2005), the MDA adoption, together with the use of UML, can provide a significant effort reduction (10% when capturing business requirements, 60% in coding and construction, and 85% for the final software environment) when executing several phases of a software development process. The trade off is a 20% effort increase during the design and the modeling phases (Sewall, 2003).

Metamodels, like the OMG software & systems process engineering metamodel (SPEM) (OMG, 2008b), are also available to describe software development processes. SPEM is meant to describe a concrete software development process or a family of related software development processes. Efforts, like the Eclipse process framework (EPF) project (**?**), produce customizable software process engineering frameworks. EPF provides a tool, the EPF Composer, that includes the eCore metamodel and that enables process engineers to implement, deploy, and maintain processes. SPEM is based on meta object facility (MOF) (OMG, 2006). MOF provides two metametamodels: the essential MOF (EMOF) and the complete MOF (CMOF). The Eclipse modeling framework (EMF) metametamodel is eCore. eCore is an implementation of EMOF (Staab et al., 2010).

The basic principle of object technology that "everything is an object" has a parallelism in MDE that
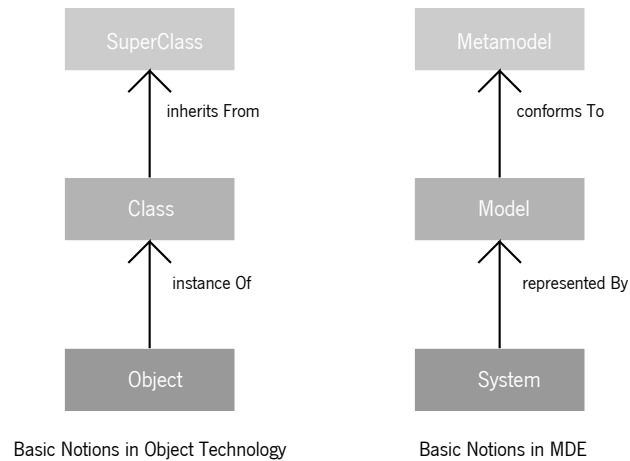
Figure 2.10.: Basic notions in object and MDE technologies (adapted from (Bezivin, 2005)).

"everything is a model" (Bezivin, 2005). In order to avoid confusing the relations between the concepts existing inside both technologies, in Figure 2.10 are also depicted the relations between the object and the MDE basic notions.

In this thesis, we use the EPF Composer tool to describe the proposed development methodology (see Chapter 7 and Appendix BIM Formalization in EPF).

2.3.3. Business Modeling

The business modeling discipline relates with the creation of accurate models to represent a business reality. A business model is a representation of the complex reality of a business organization. According to Bridgeland & Zahavi (2009), there are four business modeling disciplines, namely:

- business process models, to capture how a business performs its step-by-step activities;

- business motivation models, to capture the goals and strategies;

- business organization models, to express who performs the work and who interacts with whom, either internally or externally;

- business rule models, to express both the external constraints (from regulations and laws) and the internal norms.

Good modeling techniques provide support for the separation of concerns principle, for rigorous analysis of designs, and for structuring construction activities (France & Rumpe, 2005). As expressed by B. Selic, cited by France & Rumpe (2005), in contrast to other engineering disciplines, software development activities have the capacity to generate final products (software systems) out of models. The ability to transform, with some degree of automation, a model into a product by using some set of transformation rules and tools is an advantage that should be exploited when developing software products. For that, models could, and should, play a major role on software development activities.

2. Organizational and Information Systems Domains

When developing business software, developers must understand the original business domain where the business requirements are coming from. Thus, the original business domain is the problem domain. Then, for the presented business requirements, software developers must create business software systems, this time in the solution domain. In order to cope with the problem domain, Jackson (2001) proposes the "problem frames" approach. Problem frames promote the decomposition of problems in order to relate requirements, domain properties, and machine specifications. Having decomposed a problem, a way to solving it is through a process of composing solutions to sub-problems (Laney et al., 2004).

To describe some domain, domain-specific languages (DSL) can be used. The DSL concept has existed since the first computer languages were designed. DSLs act as an improved abstraction of the problem to allow a rapid creation and maintenance of an application for a particular domain if the abstraction is sufficiently clear to a domain expert (i.e. someone who is familiar with the domain but not necessarily a software programmer). Afterward, the DSL is translated into some other form more usable by the domain-specific runtime system, for example a configuration file, an interface, or even source code (Sprinkle & Karsai, 2004).

A DSL is a small, usually declarative, language that offers expressive power focused on a particular problem domain. In many cases, DSL programs are translated to calls to a common subroutine library and the DSL can be viewed as a means to hide the details of that library (van Deursen et al., 2000).

According to France & Rumpe (2005), DSLs, being either a programming or a modeling language, have advantages over general-purpose languages, because:

- domain-specific constructs are better suited for communication with users in the domain;
- DSLs have restricted semantic scope, and thus leading to create a semantic representation requires less effort;
- in the case of restricted semantic scope, it is easier to generate implementations from models;
- DSLs increase domain specific reuse of components, leading to improved quality in the resulting systems, and a reduction in the development time.

Some, among many, examples of DSLs are the languages to define the routing and mediation rules in Apache Camel, namely a Java-based Fluent API, Spring or Blueprint XML Configuration files, or Scala-based DSL (Apache, 2009a).

According to Fowler (2003), in the object oriented world, a Domain Model is an object model of the domain that incorporates both behavior and data. Thus, a Domain Model creates a web of interconnected objects, where each object represents some meaningful individual, being either as large as a corporation is or as small as a single line on an order form.

Two styles of Domain Models are possible: simple, very similar to databases and having mostly one domain object for each database table; and rich, embodying inheritance, design patterns, and complex webs of small interconnected objects. Both styles of Domain Models are suitable to be used in software

implementations for parts of business processes (see Section 6.3.2). Many domain specifications are available for use, for example:

- the specification for an ALert MAnagement Service (ALMAS) in combat management systems, from OMG[4] consisting of a standard alerts data model and a model for an alert delivery and life-cycle management service (OMG, 2011a);

- the specification of a set of business objects and related abstractions to support international currency (OMG, 2000).

According to the RUP (IBM Rational, 2006), the purposes of business modeling are:

- to understand current problems in the target organization and identify improvement potentials;

- to assess the impact of organizational change;

- to ensure that customers, users, developers, and other parties have a common understanding of the organization;

- to derive the software system requirements needed to support the target organization;

- to understand how a to-be-deployed software system fits into the organization.

RUP also states the need to have a dynamic view of the processes of the organization, complementing a static view of the structure of the organization. Out of these purposes and needs comes the necessity to express the business modeling effort in more than one language and view. In addition, according RUP, six scenarios for business modeling are available, with their use depending on the context and the needs, and that can promote the use of different notations and techniques to execute them:

- organization chart, when only a simple map of the organization and its processes is needed in order to better understand the requirements for the application that will support the organization;

- domain modeling, when mainly managing and presenting information requirements are present. In such cases, workflows are not considered;

- one business many systems, to use when one business modeling effort is to be used by a large system of by a family of applications sharing the same business model;

- generic business model, when an application is to be used by several organizations. This scenario is later used for Chapters 5 and 6;

- new business, when an organization wants to start a completely new line of business. This would be the case of using the concepts of Section 6.3.2 with the proposed methodology of Section 7.2;

- revamp, to support completely renews on how an organization does its business.

Thus, depending on the business modeling scenario, the surrounding environment of a project, the skills of the developers, or the characteristics of the target organization, then the business modeling effort may be described ranging from informal languages, like written text or organizational charts,

---

[4]http://www.omg.org/spec/index.htm

by semi-formal languages as UML use case diagrams or UML activity diagrams, or even by formal languages, like CPNs. It is a responsibility of the project team to decide the blend of artifacts and languages to express the business modeling efforts.

Ontologies are increasingly important tools for structuring domains of interests (Andersson et al., 2006). Major business modeling techniques are: e³value (Gordijn, 2004), resource-event-agent (REA) (Hruby et al., 2006; McCarthy, 1982), and the business modeling ontology (BMO) (Osterwalder et al., 2002). Whereas e³value is designed for modeling value exchanges within an e-business network of multiple business partners, the REA ontology assumes that, in the presence of money and available prices, all multiparty collaborations may be decomposed into a set of corresponding binary collaborations. BMO focuses on the position of a specific business partner in the e-business network and how he can make profit (Schuster & Motal, 2009), and for that is considered an ontology focused on the internal scope of an organization. Efforts also exist to combine these different ontologies, like in (Andersson et al., 2006) or in (Schuster & Motal, 2009). Because of the structuring role that ontologies provide, they can be used as sound references to create business models, and subsequently software systems. Despite organizations could be modeled by general-purpose languages, like UML (Marshall, 2000), the business modeling ontologies can bring semantics into those general-purpose languages in order to better describe business models.

The importance of business modeling for software development comes from the need to have correct, meaningful, and holistic models for the business. In this thesis, we also intend to use the same business model, with the same business content, expressed by a single language during the complete life-cycle of a development project and also during the productive usage of the resulting software product.

As stated by Smith & Fingar (2002), the technology-planning horizon for big companies is now a synthesis of software engineering and process engineering.

2.3.3.1. Business Processes Modeling Languages

To properly express business modeling efforts, an adequate language, or a set of languages, should be used at both project and productive usage periods. It is advisable that the choice of the most adequate languages for a particular business modeling effort takes into consideration more than one criteria.

It may happen that the most technically advanced language is the less adequate language for a particular project, because it does not have a proper tool support or because it is unreadable by the business experts of the project. Thus, to assure the quality of the software product resulting from a business process implementation project, it is advisable to select a business process language compatible with the organization where the project is developed, as well as with the organization where the business processes will run (Santos et al., 2013).

Several languages are reviewed by Ko et al. (2009), namely by describing their technological characteristics and their strengths and weaknesses. Twelve business process languages are also compared by Recker et al. (2009), according to a representation model proposed by Wand & Weber (1990), in

order to establish differences to their representational capabilities in the information systems domain. The most common approach to compare the modeling capabilities of the business process languages is the set of workflow patterns defined by van Der Aalst et al. (2003), which shows if the representation of a business process workflow is directly supported by a given language.

We propose that a comparison between business process modeling languages does not restrict to the technological dimension but also includes information systems, organizational, and business related dimensions, according to the information systems strategic triangle (Pearlson & Saunders, 2006). Later, we include (see Section 6.2.3) such a comparison between five major business process modeling languages: BPMN (OMG, 2009a), BPEL (Juric et al., 2004), XPDL (Shapiro, 2006), YAWL (Van Der Aalst & Ter Hofstede, 2005), and CPN (Jensen & Kristensen, 2009), together with the justifications to choose BPEL in a particular project.

The origins of process modeling languages are quite diverse, although two dominant approaches exist: one based on graphical models, and the other based on rule specifications (Lu & Sadiq, 2007). In this thesis, we prefer languages that include, or are able to be expressed, by graphical models due to the intention of having non-IT staff, namely business process experts, manipulating them. Among such languages, and despite not having a strict graphical notation, BPEL emerges as a balanced solution due to, among others characteristics (see Section 6.2.3), its extensive tool support (Oracle, 2012), (IBM, 2012), (Microsoft, 2010), (Apache, 2009b), or (Eclipse, 2012), that also provide graphical visualization capabilities of BPEL XML-based notation. BPEL is also seen as the emerging de-facto standard for executable process specification (van der Aalst & Lassen, 2005).

OASIS (2007) defines the web services business process execution language (WS-BPEL, referred as BPEL in this thesis) a language for specifying business process behavior based on web services. In BPEL, processes export and import functionality by using web service interfaces exclusively. Business processes can be described in two ways:

- executable business processes, to model actual behavior of a participant in a business interaction;
- abstract business processes, to model partially specified processes not intended to be executed.

An abstract process may hide some of the required concrete operational details. Abstract processes serve a descriptive role, with more than one possible use case, including observable behavior and process template. BPEL is meant to model the behavior of both executable and abstract processes, and provides a language for their specification. By doing so, it extends the web services interaction model and enables it to support business transactions.

BPEL builds on IBM's web services flow language (WSFL) (Leymann et al., 2001) and Microsoft's web services for business process design (XLANG) (Thatte, 2001) and combines accordingly the features of a block structured language, inherited from XLANG, with those for directed graphs, originating from WSFL (van der Aalst & Lassen, 2005). BPEL is an XML-based language for enabling task sharing across multiple enterprises using a combination of web services. BPEL is based on the XML schema,

simple object access protocol (SOAP), and web services description language (WSDL). BPEL provides organizations with an industry standard for business process orchestration and execution. By using BPEL, organizations can design a business process that integrates a series of discrete services into an end-to-end process flow. This integration reduces process cost and complexity. According to Bradshaw et al. (2005), BPEL enables the definition of how to :

- send XML messages to, and asynchronously receive XML messages from, remote services;

- manipulate XML data structures;

- manage events and exceptions;

- design parallel flows of process execution;

- undo portions of processes when exceptions occur.

The BPEL process model is layered on top of the service model defined by WSDL. At the core of the BPEL process model is the notion of peer-to-peer interaction between services described in WSDL; both the process and its partners are exposed as WSDL services. A business process defines how to coordinate the interactions between a process instance and its partners. In this sense, a BPEL process definition provides and/or uses one or more WSDL services, and provides the description of the behavior and interactions of a process instance relative to its partners and resources through web service interfaces (OASIS, 2007).

Currently, business processes exist that require human interaction. Since BPEL is web service-based, an expansion of BPEL is undergoing to cover human interaction[5]. Humans can be involved in business processes as a special kind of implementation of a BPEL activity. To facilitate this, a new BPEL activity type called people activity is required. From the point of view of the BPEL business process, a people activity is a basic activity not implemented by a piece of software but realized by an action performed by a human being. When a BPEL engine initiates a people activity, it creates work items ("to-dos") for the activity and distributes them to all people eligible to perform it. Perhaps, one of these eligible users decides to work on the activity by claiming it. This provides the user with unique access to the activity. The user can read the input data, perform whatever actions are needed, create the result data of his work as the activity's output data (or a fault), and pass that data back to the BPEL engine ("check-in") or make changes to the data in the underlying document. In addition, the nature of a knowledge-workers interaction dictates that there must be a provision for ad hoc attachments as a necessary part of the process (Kloppmann et al., 2005).

2.3.3.2. Enterprise Architecture

Organizations have business goals and, to better and faster achieve them, they rely on information systems. According to Ross et al. (2006), organizations have foundations for execution, as humans

---

[5]See OASIS WS-BPEL Extension for People (BPEL4People) Technical Committee on https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=bpel4people

Figure 2.11.: The IT-business alignment problem (adapted from (Ross et al., 2006)).

have inborn abilities to successfully perform complex activities without thinking (e.g. breading). A foundation for execution digitizes routine processes within an organization to provide reliability and predictability in processes that must go right. For that, an example of a foundation for execution is the IT infrastructure and the digitized business processes automating the core capabilities of an organization.

To build an effective foundation for execution, organizations must be able to execute the following disciplines (Ross et al., 2006):

- operating model, which is the necessary level of business process integration and standardization for delivering goods and services to customers;

- enterprise architecture (EA), considered the organizing logic for business processes and IT infrastructure, reflecting the integration and standardization requirements of the organization's operating model. EA provides a long-term view of the processes, systems, and technologies of an organization;

- IT engagement model, the system of governance mechanisms that ensure business and IT projects achieve both local and company-wide objectives.

The better the foundation for execution of an organization, the better the performance of that organization, namely in terms of profitability, faster time to market, or lower IT costs (Ross et al., 2006). In an organization with a good foundation for execution, the capacity to adapt and react to a changing business environment improves because IT and business practices to support core functions exist.

Figure 2.11 shows the IT-business alignment problem. This problem is also one of the main drivers to the present thesis and some input to its solution is later presented on Chapter 7, through a methodology to transform business process requirements into business software systems. The IT-business alignment problem is expressed by the existence of gaps between the business strategy and the solution design that supports it, as well as between the solution design and the data, applications, and infrastructures.

On top of the potential intrinsic gaps caused by the existence of the three different models (e.g. gaps caused by inappropriate model transformations), the organizations have to properly acquire capable human resources to deal with these three different models. Despite the effort to create and maintain an adequate foundation for execution, the foundation for execution affects positively the business performance.

In this thesis, the business strategy is discussed in Section 2.2. Data, applications, and infrastructures

are discussed, although with different depths, in Chapter 3. Solution design, understood as EA, is briefly presented in the remaining of this Section.

In (Council, 2001) and (USAgov, 2002), EA is defined as an information asset base, which defines the mission, the information necessary to define the mission, the technologies necessary to perform the mission, and the transitional processes for implementing new technologies in response to the changing mission needs. An EA includes a baseline architecture, a target architecture, and a sequencing plan. EA takes a holistic approach including domains of business and solution architectures but always with the global organization context in mind, being common to include enterprise analysis and planning, and architecture governance. There are disciplines related and overlapping (in the scope of the manipulated objects, not on the perspectives on how they are used) to EA (Temnenco, 2007), like:

- solution architecture, which relates with technical domains that provide solutions to challenges (e.g. information systems development, or software maintenance). It embodies an implementation perspective;

- business architecture: concerned with business terms and the description of how business operates. It is a business perspective not an technical IT one.

Despite being an undergoing effort, the main reference for EA is the enterprise architecture body of knowledge (EABOK) (EABOK, 2004). The EABOK proposes the decomposition of EA in the following seven knowledge areas, further broken down into topics:

1. EA charter and context: including topics related with the definition of EA, legislation and guidance, EA and strategic planning, scope and boundary of EA, historical developments in EA;

2. foundational practices and tools for EA development: EA frameworks (such as TOGAF (Spencer et al., 2004), DoDAF (of Defense (USA), 2010), or Zachman (Zachman, 1987)), reference models, EA development processes, modeling methods, EA modeling and analysis tools;

3. establishing and managing the EA program: EA governance, EA planning, EA tailoring, EA costs, risks, EA configuration management, issues in staffing the EA program, EA life-cycle, maturing the EA program;

4. engineering the EA: engineering issues for views, architectural patterns, component-based architectures, service-oriented architectures, federated architectures, using reference models and reference architectures, issues with legacy systems, COTS issues, flexibility and other properties, sequencing plan transitioning and evolution;

5. using the EA: complying with the agency, transforming the agency;

6. evaluating the EA: EA maturity models, assessment of EA quality and properties, assessment of EA products, assessment of EA development processes, assessment of EA usage processes, assessment of EA resources;

7. lessons learned and practical advices.

According to Hagan (2004), EA early contributors are Finkelstine's developments in Information Engi-

neering in mid-1970s, addressing problems like data redundancy, and Chen's enterprise-relationship diagrams and Yourdon's Structured Analysis work in the 1970s and 1980s on data modeling techniques, system analysis, and design methods.

EA is a comprehensive effort for an organization with a much broader scope than normal software development processes and techniques (which address more specific business needs). EA also addresses questions related with decision-making and prioritization of concurrent software development efforts for the global benefit of the organization. As opposite to software development processes, which due to their project-related nature must have a defined start and a defined end, EA efforts can run throughout the organization life-cycle.

According to the EABOK (2004), an EA framework does the following:

- identifies the types of information needed to portray an EA;

- organizes the types of information into a logical structure;

- describes the relationships among the information types.

An organization should choose an EA framework according to the areas where the organization needs more improvements. It is expected that an EA framework be tailored to the organization using it (Abdallah & Galal-Edeen, 2006). Comparisons of the distinct EA frameworks can be found in (Urbaczewski & Mrdalj, 2006) and in (Schekkerman, 2003). Abdallah & Galal-Edeen (2006) provides a framework to compare and select EA frameworks, as well as the influences between each EA framework.

EA frameworks provide the basis for an organization to reflect its strategic business vision. The planned EA, somewhere in the organization's future, materializes into running business systems. An EA describes how the elements of an organization fit together, like its business processes, the management responsible for them, the IT capabilities and infrastructure, both in the present and in the future.

As stated by Temnenco (2007), a new role inside organizations is emerging: the enterprise architect. Such a professional should be a thought leader, a visionary, and an industry expert, including not only technical skills but also human relationship skills. He should be a bridge between technical IT and business stakeholders.

The enterprise architect receives the business vision from organizations. With the solution architecture and the business architecture disciplines, he must provide to the organization a roadmap, where IT and business architectures are aligned with the vision of the organization, and actions to implement the roadmap.

Organizations should link the efforts in the business and in the software domain areas. If organizations do not make this link explicitly and properly, then they can create wastes in their operations. Such wastes are the creation of business entities and processes without proper IT support, and thus leading to the proliferation of department-maintained worksheet files and silos of information. Other wastes are related to organizations that embody "super" IT systems but without proper use (see also Section 2.3.1).
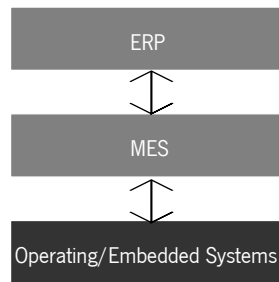
Figure 2.12.: CIM layers.

Brown & Bahrs (2009) point out the need for engineers developing low level systems to be aware and fit their products into the wider and abstract business-driven concerns supported by EA.

2.3.4. Manufacturing Execution Systems

The term MES arose in early 1990's. Initially, it stood for Manufacturing Execution Systems. Some authors, like Scholten (2009), claim that currently it is more appropriate to use the term to designate Manufacturing Enterprise Solutions, mainly because MES functionality coverage has been extended significantly and can now provide a common and single system to support most of the manufacturing execution processes. MES coverage can range from the production order releases to the delivery of finished goods processes (De Ugarte et al., 2009). MES are a particular type of information systems that support operations at plants. Typical tasks covered by MES are the monitoring of the execution of the production, the collection of data from production activities, the calculation of production-related performance indicators, or the preparation of work instructions. MES describes the suite of software products that enables the execution of manufacturing through the integration of planning and control systems (Van Dyk, 2000).

MES are important parts of Computer-Integrated Manufacturing (CIM) efforts. CIM is concerned with the appropriate integration of enterprise operations by means of efficient information exchange within the enterprise, with the help of IT. CIM includes the physical and logical connections of processes by means of data communication technology operating according to specified standards. MES also include the integration of enterprise functions with the enterprise information (Jorysz & Vernadat, 1990).

As depicted in Figure 2.12, the MES layer lays between the ERP level, where more abstract enterprise-wide operations are executed, and the lower level of "Operating and Embedded Systems", where machine controllers, sensors, or actuators are placed (also referred as supervisory control and data acquisition (SCADA) systems). The "Operating and Embedded Systems" layer contains real world objects, like machines in a shop-floor, fork-lifters, machine controllers, and its embedded and controlling software.

MES plays a crucial role in the exchange and transformation of data, collected or transferred to real world objects, with the more abstract level of the ERP systems. The data flowing from MES to ERPs is

bidirectional, like when receiving a production plan and drill it down until production line or machine levels, or, in the other way around, when providing produced quantities and defects detected to the ERP to trigger orders to suppliers or calculate bad quality costs. Similarly, the exchange of data between the MES systems and the operating and embedded systems is also bidirectional.

Harkins et al. (1999) state some reasons to integrate ERP and MES systems, namely:

- the supply-chain visibility;

- the plant decision support;

- better data.

Having timely and accurate data about raw material demands and consumption, production outages, or finished product availability improves the visibility of the supply-chain. If consistent data of shop-floor and business are available for both plant-level and enterprise-level decision support, the soundness of decision-making is improved as well as the data consistency. By having consistent data, its accuracy also improves. If the inventory data kept in the ERP is accurate in relation with the real inventory reported in the MES (e.g. by using barcodes), then proper orders to suppliers can be triggered. Additionally, a proper integration of ERP and MES systems has advantages on the reduction of the daily used resources assigned to activities related with data keeping and on endless meetings and discussions about which data is more accurate.

The most important MES standards are:

- ISA 95, which is an comprehensive and widely used standard for enterprise control systems integration, based on five parts: standard terminology and objects models ISA (2010a); attributes of objects (ISA, 2010b); functions and activities at production level (ISA, 2005); objects models and attributes of manufacturing operations management; and business to manufacturing transactions (ISA, 2007) (the last two parts still are under development). The ISA-95 model also defines relations and data exchanged between its components and external entities;

- manufacturing enterprise solutions association (MESA), which is a model that includes eleven execution activities. It comprises activities like the resource allocation and status, the operations/detail scheduling, the document control, or the dispatching production units (Leibert et al., 1997). According to Scholten (2009), the MESA honeycomb model is clear despite not excelling on the statement of exchange of information between activities;

- VDI guideline, which is considered as a compendium for the deciders, the process owners and users, the involved in selection and introduction processes with their technical emphases, and the providers of software, hardware and integration. The VDI guideline offers a problem-oriented description of MES and its application potentials. It reflects existing concepts of the MESA, completes and updates them and displays the interests of the European manufacturers in a better way. This guideline focuses on the description of the problems and the benefits of MES. The collateral standardization and definition projects (e. g. NE 94, ISA S95, IEC 62264) defer to the aspects of realization in detail (VDI, 2007).

One of the most important generic roles that MES play is closing the bridge between real-time machines and batch-oriented ERP systems. In fact, without MES, ERP decisions taken in the business-domain should always need to be manually transformed into real world objects. For instance, in a factory when the produced quantity reaches the planned figures, some actuators, controlled by a MES, can prevent production lines from receiving new parts, or can block the packaging of products. Without MES, these tasks will depend on the awareness, availability, and capacity of the human resources to intervene in real-time in the shop-floor.

Since its debut, MES have significantly evolved into more powerful and more integrated software applications, as computing technologies have advanced and the diversity of machines and technologies, and consequently the complexity, have increased in the shop-floor. However, MES applications still fall short of adding capabilities for decision-making tools in very dynamic organizations where adaptive execution strategies are required to replenish the supply-chain while dynamically responding to unpredicted change (De Ugarte et al., 2009).

In this thesis, the importance of MES comes from the type of software systems developed and analyzed (later presented in Chapters 5 and 6). In fact, despite supporting distinct business processes, all the four presented software systems are MES.

2.4. Conclusions

The internal structures and the business behavior of organizations are key inputs for the development of business software. Business process-oriented organizations are, currently, a way to focus and to streamline the operations of the business organizations. Thus, the governance of the business processes take a major role on such organizations. By supporting and facilitating business processes governance activities, and by having sound execution frameworks for business processes, organizations can gain competitiveness. For the time being, a proper support for business processes is not feasible without IT.

Additionally, the content of a business processes landscape should not only be derived from the internal knowledge residing inside an organization, and for such purpose, BP-RMs are valuable assets that organizations can use and adapt to soundly create their own business processes landscape.

Due to the actual highly interfacing nature of organizations, it is crucial to decrease misunderstandings and to increase the interoperability between organizations. In this area, ontologies and Domain Models are tools to proceed on those directions.

In our opinion, the business organizations not created from scratch with a mindset oriented towards business processes struggle when changing from the existing more traditional departmental structures. This observation lays in the fact that when an organization is created with a hierarchical and departmental mindsets, leads to the creation of internal nucleus of silos of power attached to the departments.

These departments are the same whom should drive the change into business process-oriented structures, and, at the end, will have to give away its organizational power. In large organizations, even when clear top management dictates guidelines to the remaining organization to commit to changes, internal saboteurs can delay and prevent the transformation. Thus, when structuring an organization relying on business processes is paramount to have a proper recruitment and human resources policies to deploy, check, and change improper mindsets.

Inside large and mid-size corporations, usually, the type of software systems used to support the execution of the business processes are ERPs, mainly to implement the management processes (e.g. business planning) and the support processes (e.g. human resources). Nevertheless, managers should also pay attention to business processes that require additional software support. One of such cases is when MES systems are required to interface between high-level ERP and very low-level embedded software in manufacturing organizations.

It is a mistake to consider that an ERP, or any identically high-level information systems (e.g. CRM), can always be a complete information system solution to all processes of an organization. There are decomposed business processes that, at a certain level of detail, may have requirements related with real-time responsiveness, which ERPs do not normally address.

BPM is a set of guidelines, tools, and techniques to improve the performance of business processes, despite of their categorization as operational, supportive, or direction setting. In a more holistic perspective, BPM is a way of creating an integrated view of a whole organization. Lee & Dale (1998) argue how BPM can be made to work, and, by being a tool for organizational design which needs to be understood, if it should only be understood by a few within the organization. Our perspective is that BPM should be understood as a way to have an integrated management of the whole organization. BPM management efforts relate also with several decomposition levels of business processes, and, for that, a noteworthy number of staff members intervene in BPM management, not only a few, but surely not all staff. Business processes can be decomposed into several abstraction levels, which at the top levels are more related with business strategy and, for that, not all staff is able to influence and change them. Nevertheless, the lower operational levels can be significantly influenced and improved by all members of the organization.

The execution of instances of business processes leads to the creation of data and information. Approaches based on the application of business intelligence techniques, such as data mining and data warehousing, to business processes (Grigori et al., 2004) can foster the self-awareness of an organization and can provide the facts to continuously improve business processes. When companies get to the point of developing intelligent business processes, the value-chain will consist of loosely coupled business processes, and the role of humans will be to strategize and to develop and manage relationships with other human players in this value-chain (Snabe et al., 2008).

Currently, quality and excellence standards related with business processes are widely available and widely used (e.g. ISO 9001 or EFQM excellence model). Tools and mature knowledge to support the

governance of business processes, as well as software development and runtime environments, are also available.

The knowledge on how to incorporate BP-RMs, and on how to tight high-level business processes descriptions residing within quality manuals and guidelines, with business processes occurring in the business reality still needs significant research efforts. Promising technologies, such as the internet of things (IoT) (Gershenfeld et al., 2004), can help on having accurate and real-time information about the execution of business processes. Such information could be used, in a feedback loop, to effectively check how business processes are running in the reality.

3. Software Systems Domain

This chapter presents the concepts related with business software and with software engineering. We present meta-models to build models of software development processes and we discuss the main types of those models. Then, we discuss some subjects associated with software engineering, namely software frameworks and software patterns. The chapter ends with the presentation of topics concerned with the development of business software and with the associated technologies.

3.1. Introduction

Software is the fuel of the world's new economy (Booch, 2000). Software systems play a major role in our lives. Software systems have a profound impact on how humans perform physical tasks (e.g. communicating with other human beings), on the speed we execute tasks (e.g. performing mathematical calculations), on when and where we perform tasks (e.g. working from home), or even on the protection of our lifes (e.g. breaking-aid systems in our cars).

The early definitions characterize the software by expressing its differences to the hardware: "To each user of a computer, the total computing facility provided for his use, other than the hardware, is the software. This is a function of the user, and it is also a function of time; and it should be" (Galler, 1962). Currently, the Association for Computing Machinery computing classification system (CCS) (ACM, 2012) refers software as "the stuff that actually persists after the people and the development tools are gone. Software, by the way, is more than just code - it includes the design, the specification, the rationale, the test suites, etc.". In this latter definition, software may also include tangible assets, like the documentation. According to Pressman & Ince (1992) "software is a logical rather than a physical system element and has characteristics that make a differentiation to hardware, namely: software

is developed or engineered, it is not manufactured in the classical sense; software does not "wear out"; and, although the industry is moving toward component-based assembly, most software continues to be custom built". Currently, the foreseen approach of developing software based on the assembly of components is a common practice, mainly driven by the complexity of the software products. Today, it is unimaginable to create a software system by fully coding all of its components. Software developers make use of frameworks like the virtual machines engines provided by the execution environments of a programming language (e.g. Java virtual machine or .net framework), use low level primitives provided by the operating system, or use software frameworks provided by third-parties. The approach to base the software development on components is aligned with early perspectives on how software engineering could be carried (McIlroy et al., 1968). In the definition of ISO/IEC/IEEE (2010), the software is:

1. all or part of the programs, procedures, rules, and associated documentation of an information processing system;

2. computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system;

3. a program or set of programs used to run a computer.

Due to the intangible nature of most parts of a software product, the software-related activities, like the development, maintenance, or management, have particularities that should be addressed at both project time and runtime. For instance, in opposition to other engineering disciplines, it is easy to create a usable software product out of a set of instructions. Also, and since the laws of natural sciences do not apply to most parts of software, project stakeholders may expect that a software product can do anything and can be instantly developed. To avoid such gaps in the perceptions of the stakeholders, software professionals should consider the particularities of software-related activities in the software development processes, in the maintenance processes, and in the activities occurring in the technical areas related with the product development (e.g. design, architecture).

Dennis (1975) defines the basic software categories: system software and application software. System software is the software that runs the hardware of the computer system (e.g. embedded software, operating systems), and application software is the software that end users perform non-computer related tasks (e.g. databases, games, business software).

In this thesis, we only address the application software category, and more precisely, the business software systems for process-oriented organizations.

Efforts from non-profit consortia, like the OASIS business centric methodology (OASIS, 2006), are focusing on the proper implementation of business processes with web services to provide ubiquitous connections among information systems and the people who use them. Major vendors of business software have changed their software architectures and have included business content in their suites to better support client organizations. In the case of SAP (Dale, 2007), the SAP R/3 reference model, now deprecated, relied on a business engineering approach where graphical process flows based on

event-driven process chains (EPC) (Scheer, 2000) were used to depict scenarios, processes, and functions. Within this approach, it is worthwhile noticing that the detailed business process components are not available to external developers. SAP R/3 provided only the interfaces to the entire transactions. The subsequent SAP approach was the use of the SAP ERP implementation content, focused on the implementation, recurring to a simpler notation by using swim-lane activity diagrams, and exposing both transactions and customization objects. Currently, SAP focus on enterprise SOA process components architecture, allowing SOA governance, with process components available for the composition of business processes by using software-implemented services. This latter approach is also a way to increase the reuse of software, because developers can use process components in more than one business process.

Advices on how to organize a supply-chain for software development, like the one proposed by the software factories approach (Greenfield & Short, 2004), extend the concept of using components to create software products. The software factories approach suggests the use of components, probably developed by distinct organizations, by performing mass customization activities. Software factories are defined as "a configuration of languages, patterns, and tools that can be used to rapidly and cheaply produce an open-ended set of unique variants of an archetypical product, not restricted to automating software development within an individual organization, but promoting the formation of supply-chains and also software mass customization" (Greenfield & Short, 2004). Software factories induce the development of software components prior to project time. During project time, developers customize and merge the components in order to create a software product. With this approach, the development and testing of single components do not influence the project time. In Section 6.3.2, we extend the concept of component by assigning business semantics to it recurring to BP-RMs, and, in Chapter 7, we also explicitly accommodate the use of components developed previous to project time in the proposed methodology for business software development for process-oriented organizations.

3.2. Business Software

According to Pressman & Ince (1992), business software is the largest single software application area. Partial business software systems (e.g., payroll, accounts receivable/payable, inventory) have evolved into management information systems (MIS) that access one or more large databases containing business information. Applications in the area of business software restructure existing data in a way that facilitates business operations or management decision making. In addition to conventional data processing applications, business software applications also encompass interactive computing (e.g. point of sale transaction processing).

Currently, developers build business software systems mainly recurring to third generation programming languages (3GLs), like Java, C++, C#, or Visual Basic. Due to some weaknesses, namely in modeling languages, metadata management, or code generation, Greenfield & Short (2004) state that

3GLs are counterproductive and a paradigm shift for the software development is needed.

The competitive advantages that organizations may have in relation with their contenders are not rooted on the use of IT infrastructures and application productivity software. Despite the heterogeneous level of services and infrastructures available on different countries, inside the same business market the IT infrastructures and standard application productivity software are not the most decisive factors to distinguish between competing organizations. IT assets (e.g. email applications, word processing, worksheets, databases, personal computers, and internet) are of widespread use and, in many cases, without any associated cost. For that, IT infrastructure and application productivity software is a basic utility, like water or electric power.

On the other way, business software can create differentiation between competing organizations. Business software can map and support the execution of more abstract concepts that provide value for the organization, such as a business process. Jost (2006) states that changes on roles of the management of technology inside organizations are occurring (e.g. moving from Chief Information Officer (CIO) into Chief Process Officer (CPO)). The borderline between IT staff and business experts is increasingly blurred, facilitating that business experts use and configure the behavior of business processes and business software.

Organizations share many business concepts among them, such as invoice or warehouse. For that, business software systems should also embody such concepts, expressed as software requirements at development time. Software requirements are the drivers to derive the design and the architecture of the software. Consequently, the quality of the resulting software product is directly related with the quality of the design and the architecture of the software (see Section 3.2.1 for details on software patterns), as well as with the development process used to create the system (see Section 3.3.2 for details on software development processes).

3.2.1. Software Patterns

Patterns are attempts to describe successful solutions to common software problems (Schmidt et al., 1996). Software developers do not invent software patterns. Rather, they discover patterns from experience in building practical systems (Devedzic, 2002). For that, patterns are not innovations. Patterns are pieces of knowledge collected from observation and experience. Patterns are described with the purpose of reuse when someone faces the same problem.

Alexander et al. (1977) started pattern discovery and formalization, in the initial domain of the architecture of buildings. Alexander (1979) states that "the quality of a building can be objective and precise, and patterns could be expressed and verified to achieve a precise measure of such quality". Shalloway & Trott (2002) refer that patterns can also be rooted in cultural anthropology observations, namely in (Benedict, 1946), where it is suggested that, inside the same culture, individuals will agree largely on what is to be considered a good design. Alexander et al. (1977) also state that "Each pattern describes

a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice".

Gamma et al. (1995) started with the use of patterns in the software domain. Currently, software developers apply patterns, among others, to the following software domain areas (Devedzic, 2002):

- design, considered as reusable micro-architectures that contribute to an overall system architecture, by identifying, naming, and abstract common themes in object-oriented design (Gamma et al., 1993);

- architecture, despite overlapping with design patterns (Buschmann et al., 2007), these patterns are described for components and connectors (Shaw & Garlan, 1996). Architectural patterns normally are coarse-grained than design patterns, but the granularity depends on the designer or the architect;

- analysis, to express conceptual business processes, not directly related with software (Fowler, 1997);

- organization and processes (Vlissides et al., 1996), related with the software development process and with the relations between developers and users;

- risk management, to capture strategies of project management by risk reduction (Cockburn, 1998)

In addition to the knowledge of good practices, expressed by patterns, software developers should also explicitly know the bad practices. For that, Brown et al. (1998) propose the concept on anti-patterns in order to express patterns of bad solutions and suggest refactoring techniques to improve the solutions. Thus, an anti-pattern is either a pattern that tells how to go from a problem to a bad solution, or a pattern indicating how to go from a bad solution to a good solution (Devedzic, 2002). The explicit knowledge of anti-patterns prevents a design to incorporate known bad practices.

Design patterns   are common software design answers for common problems, presented during software development. From the initial work in patterns from the Gang of Four (GoF) (Gamma et al., 1995), a finite number of design patterns, initially 23, could be identified as well as the most appropriated conditions to use them. According to the GoF (Gamma et al., 1993), a template to describe a design pattern has the following elements:

- pattern name;

- intent, stating what the pattern does;

- motivation, expressing a scenario where the problem can be applied;

- applicability, stating the situations where the patterns can be applied, including bad designs that the pattern address;

- participants, describing the classes and/or objects participating in the design pattern and their
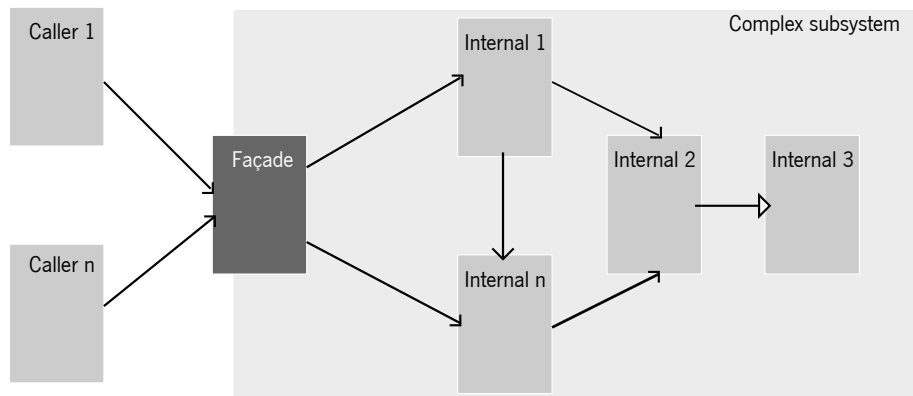
Figure 3.1.: Diagram of the Façade design pattern (adapted from Gamma et al. (1995)).

responsibilities;

- collaborations, describing how the classes and/or objects collaborate;

- diagram, representing graphically the pattern;

- consequences, expressing the results and the trade-offs of using the pattern;

- implementation, stating pitfalls, hints, or techniques to use properly the pattern;

- examples, depicting examples of real systems;

- additional information, to list related design patterns, and possible mixed used with other design patterns;

Shalloway & Trott (2002) state that "Design patterns set a general approach. Whether or not to add functionality depends on the situation at hand. Patterns are blueprints to start a design; they are not carved in stone". Initially, the GoF divided design patterns into three categories:

- creational, related with object creation. Included are the Abstract Factory, Builder, Factory Method, Prototype, and Singleton patterns;

- structural, concerned with the creation of structures of objects. Included are the Adapter, Bridge, Composite, Decorator, Façade, Flyweight, and Proxy patterns;

- behavioral, related with the management of algorithms and communications between objects. Included are the Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, and Visitor patterns.

Figure 3.1 represents the diagram of the Façade design pattern. The purpose of the Façade pattern is to provide a unified interface to callers to a set of interfaces inside some complex subsystem, facilitating the subsystem use by external callers. The subsystem would be more difficult to use if the external callers should know all the interfaces of internal objects. As any other design pattern, the Façade pattern can be combined with other patterns. For instance, when developers need a stateless object, the Façade pattern can incorporate the Singleton pattern to allow only one instance of the class. Among others, we use the Façade pattern in the architecture presented in Section 7.3.3 as the approach to

interconnect tiers.

Architectural Patterns are descriptions of element and relation types, together with a set of constraints on how they may be used. An architectural pattern is a set of constraints on architecture (on the element types and their patterns of interaction). These constraints define a set, or a family, of architectures that satisfies those constraints (Bass et al., 2003). The purpose of architectural patterns is to impose an overall structure for a software system. Shaw (1996) states that such structure:

- is appropriate to the problem the system is solving;

- clarifies the intentions of the designer about the organization of the system;

- provides a paradigm that will help establish and maintain internal consistency;

- allows appropriate checking and analysis;

- preserves the information about itself (for reference in later maintenance activities).

Avgeriou & Zdun (2005) list architectural patterns, coming from different sources, and relate them with the architectural views of a system:

- layered view, related with the decomposition of a system into interacting parts. Patterns: Layers, Indirection Layer;

- data flow view, related with the processing and transformation of streams of data by components. Patterns: Batch Sequential, Pipes and Filters;

- data-centered view, related with the access of a central repository of data by multiple components. Patterns: Shared Repository, Active Repository, Blackboard;

- adaptation view, related with the adaptations occurring in the system during its evolution. Patterns: Microkernel, Reflection, Interceptor;

- language extension view, related with how the system offers an abstraction layer to the computational infrastructure. Patterns: Interpreter, Virtual Machine, Rule-based System;

- user interaction view, related with the runtime structure of user-interaction components. Patterns: Model-View-Controller (MVC), Presentation-Abstraction-Control, C2;

- component interaction view, related with the exchange of messages between components, yet maintaining the autonomy. Patterns: Explicit Invocation, Implicit Invocation, Client-Server, Peer-to-Peer, Publish-Subscribe;

- distribution view, related with the dissemination of components in a networked environment: Broker, Remote Procedure Calls, Message Queuing;

The number of architectural views depends on the software development process in use (e.g. RUP uses the 4+1 view model (Kruchten, 2004)). Due to the complexity, expressed by the number of software components and their connections, the software architecture discipline has emerged as a foundational concept for the successful development of large, complex systems (Clements et al., 2003), and for that, architectural patterns are a fundamental pillar to develop sound architectures.

Patterns of Enterprise Application Architecture are patterns related either with the software architecture, when they represent significant decisions about the parts of the architecture, and with design, when they help to realize the architecture (Fowler, 2003). Fowler lists more than 50 enterprise application architecture patterns, organized in Domain Logic, Data Source Architectural, Object-Relational Behavioral, Object-Relational Structural, Object-Relational Metadata, Web Presentation, Distribution, Offline Concurrency, Session State, and Base patterns. For this thesis, the importance of these patterns comes from the direct relation with business software. Among others, we use the Domain Model enterprise architecture pattern as the domain logic for the later presented concepts (in Section 6.3.2) and Actions (see Section 6.3). The Domain Model pattern encapsulates both data and processes inside the same object model of a domain, meaning that the complete set of objects modeling a business area are inside the same object model.

Enterprise Integration Patterns are patterns to integrate applications and business processes (Hohpe & Woolf, 2002). Enterprise integration patterns address the concern to integrate business software applications by using messaging. Messaging is a technology that enables high-speed, asynchronous, program-to-program communication with reliable delivery as opposed to other integration approaches (e.g. file transfer, shared database, or remote procedure invocation) (Hohpe & Woolf, 2004). In this thesis, we use enterprise integration patterns in Action Centauro 2.0 and in Section 7.3.3 as the basis to reliably interconnect different parts of a proposed software architecture.

3.2.2. Software Frameworks

According to Stahl & Völter (2006), a software framework is anything that can be adapted or extended via systematic extension or configuration mechanisms. A component is a self-contained piece of software with clearly defined interfaces and explicitly declared context dependencies.

The use of software frameworks can improve the quality of a software product by using tested software components, provided by the frameworks, to create new components. Software frameworks can also help reducing the duration of a project because the effort to develop the majority of the software is done before project time. In addition, software frameworks can act as technology enhancers for the software product, because they can provide already developed technological sound solutions (e.g. reliable messaging, hot deployment), probably not considered as features for the software product if they should be developed from scratch.

In the next paragraphs, we present two software frameworks, the Spring (Johnson et al., 2005) and ServiceMix 4 (SMX4) (Foundation, 2011), that help the development of business software and are the software backbones for the Actions of Sections 5.3, 6.2, and 6.3. We also present the Apache Orchestration Director Engine (ODE) (Apache, 2009b) that allows the execution of BPEL, and is a vital component of the proposals expressed in Section 6.3.
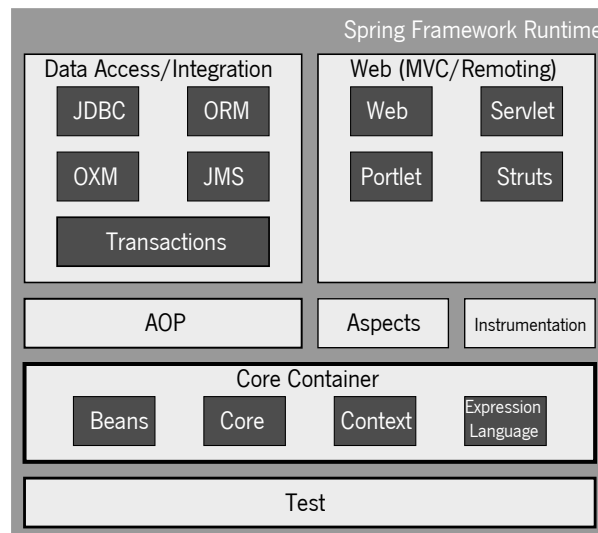
Figure 3.2.: The architecture of the Spring framework (based on (Johnson et al., 2012)).

Inversion of Control   Since Java follows the object-orientation language paradigm, a Java application consists of a set of collaborating objects organized in such a way that requirements are fulfilled. Since classes may have relationships (e.g. Association, Aggregation, Composition (OMG, 2011d)), the objects derived from those classes also have dependencies, namely at runtime. As stated by Johnson et al. (2012), the Java platform misses a way to organize the building blocks into a coherent whole, leaving this task to designers and architects. Design patterns can be used to overcome this limitation, like by using the Factory, Builder, or Service Locator patterns to compose the classes and object instances that are needed to create a software application.

The Spring framework addresses the organization of the building blocks of an application by providing formalized means of composing disparate components into a fully working application ready for use (Johnson et al., 2005).

Figure 3.2 depicts the modules that reside in the Spring framework. Johnson et al. (2012) group the modules in the following components:

- core container, where the implementation of the fundamental parts of the framework resides, like the implementation of the support for the Inversion of Control (IoC);

- data access/integration, where data-related (e.g. Object-Relational Mapping (ORM), or JDBC) and integration-related (e.g. Java Messaging Services (JMS)) technologies are implemented;

- web, including the implementation of the MVC architectural pattern through Spring MVC or Struts;

- aspect oriented programming (AOP) (Kiczales et al., 1997), to decouple the functional code from the code to support design decisions. AOP promotes that design decisions are cross-cutting the application and for that they should not be mixed with code implementing functionality. Among others, cross-cutting concerns may be checking and handling errors, context-sensitive behavior,

monitoring, or logging;

- aspects, to support the AspectJ[1] language (a seamless aspect-oriented extension for the Java, compatible with the Java platform);

- instrumentation, for class instrumentation and support of class-loaders for application servers (e.g. Apache Geronimo[2]).

- test, to support unitary tests (e.g. JUnit) and to provide mock objects to test the code in isolation.

Fowler (2004) proposes a renaming of IoC pattern into Dependency Injection pattern, because he states that IoC is a ground characteristic of each framework and thus too generic. Dependency Injection refers then to have a separate object, an assembler, which populates a field in the listener class with an appropriate implementation for the finder interface, resulting in a dependency diagram. There are three types of Dependency Injection (Fowler, 2004; Johnson et al., 2005): Constructor Injection, Setter Injection, and Interface Injection, depending on where the injection of the implemented object is accomplished.

Despite the presented advantages, the Dependency Injection frameworks have a static wiring of objects. In this kind of frameworks, the wiring between the objects is done once at the start-up of the application and remains active until the application is shutted down (Bartlett, 2009). This characteristic leads into a complex graph that needs to be maintained to describe all the wiring between objects. Additionally, there is no possibility to perform on-the-fly updates to the wiring or to the objects inside the running software system.

Enterprise Service Bus   To overcome the static wiring and missing on-the-fly updates of running software components, developers can use other software frameworks, namely those built on top of the OSGi architecture. The OSGi architecture provides a basis for a general-purpose, secure, and managed Java framework that supports the deployment of extensible and downloadable applications known as bundles (**?**). OSGi architecture provides a mature component system that works in a large number of environments. The OSGi component system is currently used to build highly complex applications like IDEs, application servers, application frameworks, telecom and service solutions, industrial automation, residential gateways, or onboard telematics systems (Alliance, 2012). OSGi-based frameworks overcome the limitations of standard Java JAR file deployment. JAR miss a runtime concept (i.e. JAR are only meaningful at build-time and deploy-time), they do not contain standard metadata to specify their dependencies, they are not versioned, and there is no mechanism to hide information between JAR files (Bartlett, 2009).

In OSGi terminology, the Java modules are renamed to bundles, and its semantics is changed. This transformation is accomplished by adding metadata to each module (i.e. name, version, the list of imports and exports, and optionally, information about the minimum Java version that the bundle

---

[1]See http://www.eclipse.org/aspectj/

[2]See http://geronimo.apache.org/

needs to run on, and miscellaneous human-readable information such as the vendor of the bundle, copyright statement, and contact address). The metadata is placed inside the JAR archive file, in a special file called MANIFEST.MF (Bartlett, 2009).

Papazoglou et al. (2007) consider Enterprise Service Bus (ESB) as a state of the art solution for a capable and manageable integration infrastructure for web services and SOA. ESBs implement a message backbone, and may be based on open-standards designed to enable the implementation, deployment, and management of SOA-based solutions. ESBs are a bus for messaging, service composition, and orchestration for a SOA (Stahl & Völter, 2006). An ESB is a standards-based integration platform that combines messaging, web services, data transformation, and intelligent routing to reliably connect and coordinate the interaction of significant numbers of diverse applications across extended enterprises with transactional integrity (Chappell, 2004).

Vollmer (2011) presents a market research stating that ESBs are in high demand and that they play an increasingly important role in supporting application integration and SOA. Among commercial solutions (e.g. Oracle Service Bus, IBM WebSphere enterprise service bus, or Tibco ActiveMatrix service bus), the ServiceMix (in the FuseSource[3] distribution) is considered as an overall leader, as well as the best open-source ESB.

An example of an OSGi-based software framework is the SMX4 (Foundation, 2011) ESB. The previous version of ServiceMix, the SMX3 (Community, 2008) has a design grounded on the Java business integration (JBI) specification (Ten-Hove & Walker, 2005). SMX3, as a JBI compliant implementation, defines two types of basic components: service engines (SE) and binding components (BC). These two types of components can act as service providers, service consumers, or both. The SEs handle the business and processing logic and the BCs handle the communication with external (to the ESB) software components. Inside SMX3, software components can be added as SEs. SMX4, which on top of the compliance with the OSGi architecture is also compliant with the JBI specification, evolved into dynamic services in an OSGi-based architecture (Alliance, 2009). In SMX4, software components are added into the ESB runtime as OSGi bundles.

Figure 3.3 depicts the architectural layers of SMX4. The Kernel layer is based on Apache Karaf (Apache, 2011) OSGi-based runtime.

Apache Karaf supports both the Apache Felix (Apache Software Foundation, 2012) and the Eclipse Equinox (Eclipse Foundation, 2012) OSGi containers. Apache Karaf also hosts, among others, modules to handle the (hot) deployment of OSGi bundles, security, or logging. In the Technological layer, among others, is provided support for Dependency Injection frameworks such as Spring and Blueprint[4], messaging (JMS), JBI, Java API for XML Web Services (JAX-WS) (Oracle, 2011), or Representational State Transfer (REST) architectural style for JAX (Hadley & Sandoz, 2009).

To handle the exchange of messages, SMX4 can use the Apache Camel (Apache, 2009a) integration

---

[3]FuseSource provides a ServiceMix distribution. See http://fusesource.com/products/enterprise-servicemix/

[4]See http://aries.apache.org/modules/blueprint.html
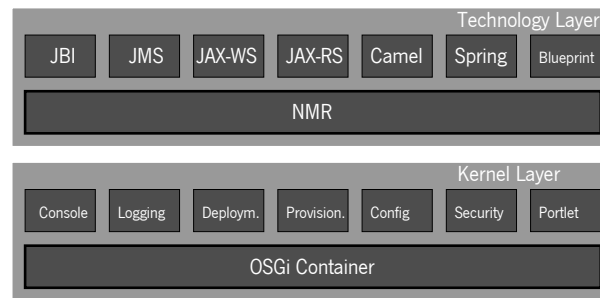
Figure 3.3.: SMX4 ESB architectural layers (based on (Fuse, 2011) and (Dirksen, 2010)).

framework to support routing and mediation rules between messages. Apache Camel provides to developers implementations of Enterprise Integration Patterns. In this thesis, we prefer to use orchestration to materialize the behavior of the business processes, but we use Apache Camel to support the implementation of message routes (see Table 6.10). Orchestration can be seen as combining service calls to create higher-level, more useful composite services, but also often has a definitive "business-level" ring, and in this case is shorthand for implementing business-level processes combining business-specific services across applications and information systems (Louis & Dutoo, 2008).

A product based on an ESB-oriented architecture needs to turn into one based on SOA to help ensure that it successfully delivers business value (Woolf, 2007). Service-oriented computing promotes the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications that span organizations and computing platforms (Papazoglou et al., 2007).

The ESB architecture is only a technological solution, and, in order to get business requirements properly implemented in software, a SOA-like approach is needed to align the business requirements with their respective support in software.

Apache ODE   the Apache Orchestration Director Engine (ODE) (Apache, 2009b) is an engine that can run, among other options, inside SMX4 as a JBI component packaged as OSGi bundle. ODE executes business processes written following the BPEL standard. ODE talks to web services by sending and receiving messages, handling data, and performing error recovery, as described by the BPEL process definition. It supports both long and short living process executions to orchestrate all the services that are part of an application. Figure 3.4 depicts the basic architecture of the Apache ODE, consisting on a BPEL compiler to perform the transformation of BPEL process documents, WSDLs, and schemas into a compiled representation suitable for execution. The BPEL Runtime then executes the compiled process. The BPEL Runtime also provides the implementation of the BPEL constructs, stores the data related with processes instances in a database management system (DBMS) using Data Access Objects (DAO), and handles incoming messages. The Jacob framework (Apache, 2009b) provides an application-level concurrency mechanism (i.e. it does not rely on threads) and a transparent mechanism for interrupting the execution and persisting the execution state. Finally, the Integration
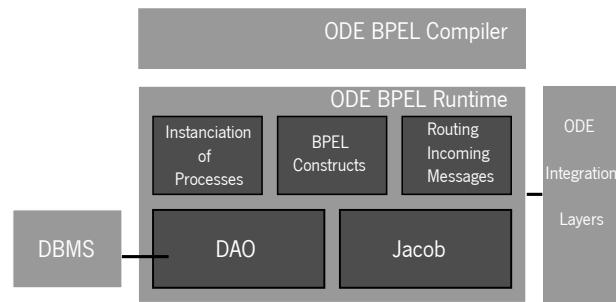
Figure 3.4.: Apache ODE architecture (adapted from (Apache, 2009b)).

Layers are responsible to embed the ODE into an execution environment, like JBI, and thus allowing the ODE to be used inside the SMX4 ESB.

3.3. Software Engineering

An early definition of the software engineering discipline points out the importance of subjective aspects, i.e. art, in the practices. Dennis (1975) states that "Software engineering is the application of principles, skills, and art to the design and construction of programs and systems of programs. It is often asserted that software engineering is largely art and based very little on sound principles". Nevertheless, in the same definition, Dennis (1975) also points that trends are visible and new ideas are developing that promise to increase substantially the role of theory and principle in the design and construction of software systems. Pressman (1988) states that a difference exists between software engineering and mere programming theory, as software engineering involves human activities in producing reliable and usable software. Software engineering also includes large-scale programming activities, management, and quality assurance.

Currently, the software engineering field has a systemic and less art-related approach. For ISO/IEC/IEEE (2010), software engineering is:

1. the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software.

2. the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

The SWEBOK 2004 (Abran & Moore, 2004) follows the IEEE definition. SWEBOK is the main reference for the software engineering discipline, supported by major software companies (e.g. SAP, IBM), by IEEE Computer Society, and by an extensive panel of experts and reviewers.

3. Software Systems Domain

3.3.1. Knowledge Areas

In order to define the scope of the software engineering area, SWEBOK proposes the following knowledge areas:

- software requirements, concerned with the elicitation, analysis, specification, and validation of software requirements;

- software design, viewed as a process to express the software engineering life-cycle activity in which software requirements are analyzed in order to produce a description of the internal structure of the software that will serve as the basis for its construction. Viewed as a product, the software design must describe the software architecture, i.e. how software is decomposed and organized into components, and the interfaces between those components. It must also describe the components at a level of detail that enable their construction;

- software construction, which includes the detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging;

- software testing, performed for evaluating the product quality, and for improving the product, by identifying defects and problems;

- software maintenance, related with the handling of uncovered defects, operating environments changes, and new user requirements, once the software product is in operation. The maintenance phase of the software life-cycle begins following a warranty period or post-implementation support delivery, but maintenance activities may occur much earlier;

- software configuration management, concerned with identification and documentation of the functional and physical characteristics of a configuration item, controlling the changes to those characteristics, recording and reporting change processing and implementation status, and verifying compliance with specified requirements;

- software engineering management, defined as the application of management activities, i.e. planning, coordinating, measuring, monitoring, controlling, and reporting, to ensure that the development and maintenance of software is systematic, disciplined, and quantified;

- software engineering process, explained in two levels. The first level encompasses the technical and managerial activities within the software life-cycle processes that are performed during software acquisition, development, maintenance, and retirement. The second is the meta-level, which is concerned with the definition, implementation, assessment, measurement, management, change, and improvement of the software life-cycle processes;

- software engineering tools and methods are the computer-based tools intended to assist the software life-cycle processes;

- software quality, covers static quality techniques, those which do not require the execution of the software being evaluated (the dynamic techniques are covered in the software testing knowledge area).

Despite being outside of the defined boundaries, other disciplines have interfacing, or even intersecting, knowledge with the software engineering field, namely: computer engineering, computer science, management, mathematics, project management, quality management, software ergonomics, and systems engineering. The business modeling discipline (see Section 2.3.3), interfaces and intersects with SWEBOK disciplines through the software requirements knowledge area. Business models resulting from business modeling activities are key inputs to define properly the requirements for business software.

Currently, SWEBOK is undergoing a revision and public review[5] leading into the SWEBOK v3. Stated reasons for the change are the emergence of new tools, methods, and types of software that nowadays make good practice in the software engineering, or the growth of the underlying body of knowledge. Included in the changes are new knowledge areas:

- computing foundations (IEEE SWEBOK 3, 2012a), concerned with the computer science foundations that support the design and construction of software products. Moreover, it also includes knowledge about the transformation of a software design into a software implementation, the tools used during this process, and the various software development methods;

- software engineering models and methods (IEEE SWEBOK 3, 2012c), to impose structure on software engineering with the goal of making that activity systematic, repeatable, and ultimately more success-oriented;

- mathematical foundations (IEEE SWEBOK 3, 2012b), covers the basic techniques to identify a set of rules for reasoning in the context of the system under study;

- software engineering professional practice (IEEE SWEBOK 3, 2012d), concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner;

- software measurement (IEEE SWEBOK 3, 2008), formerly spread across the other knowledge areas, and relating with measuring the software development product and the software product;

- software security (IEEE SWEBOK 3, 2012e) to cope with traditional issues of inadvertent and unintentional risks and losses, and also with maliciousness, illegitimacy, and concern for confidentiality including protecting privacy and intellectual property.

3.3.2. Software Development Processes

Organizations that develop software must monitor, analyze, and improve continuously their development processes. They must embody into their development processes what they have learned from their operations. They must also continuously look for new processes and practices that they may use to improve their development processes.

According to Curtis et al. (1992), the objectives of modeling a software process are to facilitate the

---

[5]See http://computer.centraldesktop.com/swebokv3review/

human understanding and the communication, to support process improvement and management, to provide automated guidance in performing the development process, and to automate execution support.

Proposals are available to increase the automation degree in model transformations from UML (Rumbaugh et al., 1998) into BPEL (Gardner, 2003), to mechanize BPM models (Hepp et al., 2005), or to use XML to allow automation of dynamic business processes (Chen et al., 2003). These proposals allow improvements on how business software systems support business processes. Nevertheless, a holistic approach to transform business requirements into running software is still missing.

Some methodologies exist to help the development of software systems to support business activities. For tailor-made business software, generic software development processes can include disciplines, like the Business Modeling discipline in the RUP, to help software developers understand and define the business context of their client in a set of deliverables. Some models used in Business Modeling discipline use a notation that applies extensions to the UML in order to design the business processes. The Business Modeling discipline of RUP provides key inputs to requirements and software design activities, namely by supporting the decision of what business processes will be implemented in software, or by deriving some business properties (e.g. the number of employees in the organization) that will influence non-functional requirements. Additionally, the RUP can also be tailored to better support COTS software development projects (Peraire & Pannone, 2005), by having customizations, for instance, for its disciplines, actors, or life-cycle.

To understand properly an organization, developers should create models. Modeling an organization is a task that produces different views of the organization, namely the functional, informational, organizational, and resources views (Vernadat, 1996). These views, in order to provide a holistic comprehension, must be integrated (Delen et al., 2005) and not conflicting. Such a modeling approach implies that the models, starting with generic textual descriptions, should reach a detailed level where developers can properly infer the software requirements. Since the objective of a process is to lead a set of products to a well defined state (Combemale et al., 2006), business process models should be handled during the software development in order to be transformed from paper-based business process models into a software-supported state, and thus executable in a business software system.

3.3.2.1. Metamodels for Software Development Processes

There is no software development process that developers can always use in any type of project without any tailoring and without any addition. Experts on software development processes, together with project leaders, should define the more suitable development process for a given project, because the characteristics of each project differ, among others, in the internal organization of the project team, in the business domain where the software product will be used, or in the technological characteristics of the product. If properly done, the customization of a model of a software development process should rely on already established knowledge about software development processes.
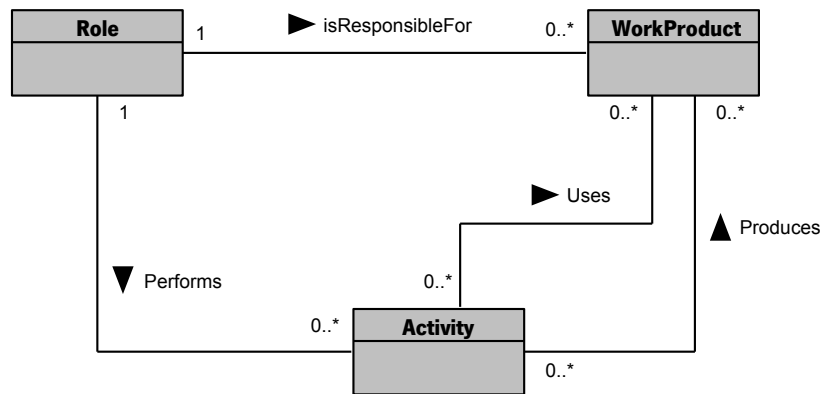
Figure 3.5.: SPEM conceptual model (based on (Combemale et al., 2006)).

As stated by Henderson-Sellers (2002), it is possible to use standard frameworks, defined by metamodels, which can then provide an extensible and customizable process environment such that individual and project-specific processes can be created and configured precisely to those project needs. As pointed by Boehm (1988), the primary functions of a software process model are to determine the order of the phases involved in software development and evolution, and to establish the transition criteria for progressing from one phase to the next. The goal of a methodology is to encourage an approach to solve a particular problem with a set of methods and techniques previously chosen (Ghezzi et al., 1991).

As pointed in Section 2.3.2.2, metamodels are available to describe software development processes, being the most noticeable the Software & Systems Process Engineering Metamodel (SPEM) (OMG, 2008b). As shown in Figure 3.5, the SPEM is built on the notion that a software development process is performed by entities, called Roles, that perform Activities resulting on outputs called Work Products. Activities normally require work products to be used also as inputs.

SPEM is meant to describe a concrete software development process or a family of related software development processes, and has as goals:

- to provide a standardized representation and managed libraries of reusable method content;
- to support the systematic development, management, and growth of development processes;
- to support the deployment of just the method content and process needed, by defining configurations of processes and method content;
- to support the enactment of a process for development projects;

Rumbaugh et al. (1991) define methodology as a series of phases, with associated techniques and notations. Whytock (1993) views a phase as an abstraction in time of a set of aggregated activities. These basic concepts related with software development processes (e.g. technique, phase) need to have explicitly defined relations and, for that, a metamodel, like the SPEM, can also act as an abstract relationship model to derive concrete implementation method libraries, all sharing the same basic concepts and relations. In Figure 3.6, the M1 level process (i.e. a project used in a real project) is
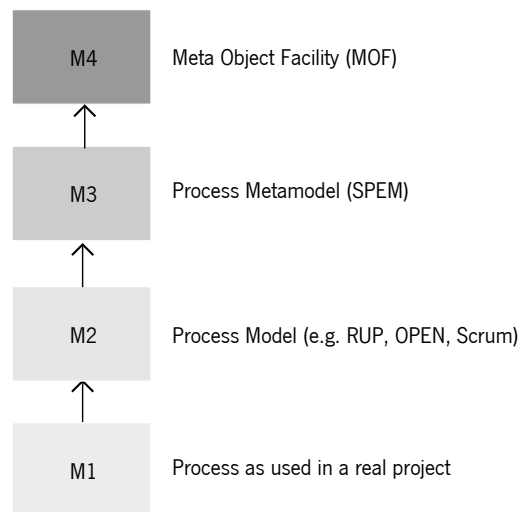
Figure 3.6.: OMG modeling architecture (based on (Acuna & Juristo, 2005)).

an instantiation of a process model, like the RUP or the Scrum, which in turn are instantiations of the SPEM process metamodel, which derives from the top-level Meta Object Facility (MOF). The SPEM metamodel uses UML as the notation and takes an object-oriented approach.

Other frameworks exist to allow the composition of software development process, like the OPEN Process Framework (Graham et al., 1997). The main initial goal of OPEN was to reunite the many different processes and create a framework, including a metamodel, to allow the creation of software development processes (Firesmith & Henderson-Sellers, 2002). In Figure 3.6, SPEM is depicted as a process model. The Eclipse Process Framework (EPF) provides library contents[6] that can be used, recurring to the EPF Composer tool, to create process models (e.g. Scrum, XP, and OpenUP are available for download).

In this thesis, we use the EPF Composer tool to create the model of the methodology presented in the Chapter 7.

3.3.2.2. Traditional Approaches

According to Boehm (1988), the earliest software development process was the Code-and-fix model, where only two steps were considered: first write some code, and then fix the problems in the code. The approach was to do some coding first and to think about the requirements, design, test, and maintenance later. Such kind of approach can lead, among others, into problems related with poorly structured software, missing the customer requirements, or redeveloping code for the same require-ments. Early reports (from 1956) of product failures and time schedule slippages when developing large software systems are reported by Benington (1983) which lead into a move to a stage-wise model approach, like the Waterfall (Royce, 1970). In the Waterfall model, the life-cycle of a software develop-ment process is divided into sequential stages. Each stage contains activities related with one discipline

---

[6]See http://www.eclipse.org/epf/downloads/praclib/praclib_downloads.php

82

of software engineering (e.g. requirements, design, code, or integration). Normally, a process model implementation based on the Waterfall model has some form of assessment to evaluate the transition between consecutive stages. Waterfall-based processes have as main drawbacks the difficulty to cope with changes without causing major impacts on project restrictions, and a focus on fully completed documents at early stages in order to allow the process to proceed to the next stage. If a major business requirement changes during the Waterfall life-cycle, all the fully completed documents should be revised or discarded, and a reset on the project life-cycle may be need, leading to slippages on the time schedule or on the needed resources. The Waterfall-based process model, which was very influential during the 1970s and 1980s, started to have difficulties to cope with constant changes in the project environment, mainly in the business software area, caused by factors external to the project. To better deal with such difficulties, the Evolutionary development model was proposed (McCracken & Jackson, 1982). With the Evolutionary model, the development of a software product is accomplished through the repetition of executions of sequences of stages. Each repetition produces an increment to the product, until the product final status is reached. The Spiral process model (Boehm, 1988) embodies the characteristics of an Evolutionary model. The Spiral life-cycle model addresses many drawbacks of a Waterfall model by providing an incremental development process, in which developers repeatedly evaluate changing project risks to manage unstable requirements and funding (Nuseibeh, 2001).

3.3.2.3. Agile Methods

In 2001, representatives of diverse agile methods, such as Extreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development, Crystal, Feature-Driven Development, and Pragmatic Programming created and subscribed a manifesto that underlies the ideas and concepts supporting all agile methods. The manifesto states principles to reduce, allegedly, bureaucracy and to focus the development teams on the most important deliverable of a software process: the software product.

The manifesto promotes individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan. The twelve principles behind the Agile Manifesto are (Beck et al., 2001):

- the highest priority is to satisfy the customer through early and continuous delivery of valuable software;
- welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage;
- deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
- business people and developers must work together daily throughout the project;
- build projects around motivated individuals. Give them the environment and the support they
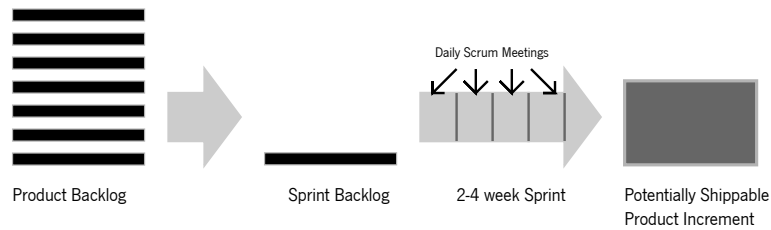
Figure 3.7.: The Scrum life-cycle (based on (Scrum Alliance, 2012)).

need, and trust them to get the job done;

- the most efficient and effective method of conveying information to and within a development team is face-to-face conversation;

- working software is the primary measure of progress;

- agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely;

- continuous attention to technical excellence and good design enhances agility;

- simplicity, the art of maximizing the amount of work not done, is essential;

- the best architectures, requirements, and designs emerge from self-organizing teams;

- at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Despite the fresh and innovative approaches provided, the agile methods are not a solution for all software development projects. The managers of an organization that develops software must pay special attention when project teams state as main reasons to use some agile method that they want to avoid documentation activities or that they will have more "freedom" during development time. These reasons are early symptoms of a project failure.

Agile methods rely heavily on technically skilled and motivated project members, with good communication capacities and tools, and with empowerment and team-awareness, able to take fast and sound decisions. This social environment is difficult to be setup, for instance, in projects with members scattered in different time zones, not speaking the same language, not having trust on each other, or missing the right technical skills.

Scrum (Sutherland, 1993) is a lightweight agile method. Scrum has only three process roles: Product Owner, Scrum Master, and the Team. Scrum is based on the ideas of Takeuchi & Nonaka (1986) where product development should adopt a flexible and holistic strategy. In Scrum, the development team works as a unit to reach a common goal. Since organizations are complex adaptive systems, Scrum requires also complex adaptive behavior.

The Scrum life-cycle is very simple (see Figure 3.7). The process starts with a Product Owner creating a prioritized wish list, called Product Backlog. During the Sprint planning, the Team pulls a small part

of the Product Backlog and creates, this way, a Sprint Backlog. The Team has a certain amount of time (the Sprint) to complete its work, normally between two to four weeks. During Sprint time, the Team meats everyday in a Daily Scrum Meeting, to assess and agree on the next steps inside the Sprint. Along the way, the Scrum Master keeps the team focused on its goal. At the end of the Sprint, the work should be potentially shippable, ready to hand to a customer, or present to a stakeholder. The Sprint ends with a Sprint review and retrospective. As the next Sprint begins, the team chooses another part of the Product Backlog and begins working again. Thus, Scrum is a non-prescriptive method with only very few roles and practices.

3.3.2.4. Unified Process

(UP) (Jacobson et al., 1999) is an iterative and incremental process model. The UP is use case driven, in the sense that uses UML use case models as the functional specification focusing the development in the benefit for each one of its functional users. UP is also architecture-centric, because it completes the functionality view with the form view expressed in the software architecture, embodying the most significant static and dynamic aspects of the software system.

The UP is a process model customizable to many different configurations, each focusing UP concepts on a particular type of a software development or project characteristic. Examples are:

- IBM Rational Unified Process (RUP) (IBM Rational, 2006; Kruchten, 2000) (see Figure 3.8) is a comprehensive set of process models ranging from big business software projects to embedded software projects. Process experts can tailor and extend the RUP to address properly the characteristics of each software development project, by recurring to the IBM Rational Method Composer (RMC) (Kroll, 2005). The fist release of RUP was in 1998 (Jacobson et al., 1999). RUP is the commercial version of UP-based process models;

- Agile Unified Process (AUP) (Ambler, 2006) is a lightweight approach intended for developing business software using agile techniques and concepts, yet still remaining true to the UP;

- OpenUP (EPF, 2012) embraces a pragmatic, agile philosophy that focuses on the collaborative nature of software development. OpenUP is a tools-agnostic, low-ceremony process that can be extended to address a broad variety of project types. OpenUP is a process intended for small and co-located teams;

- Oracle Unified Method (OUM) (Oracle, 2014) focuses on project managers and stakeholders "develop a shared understanding of what is needed, choose an appropriate architecture, and transfer the ownership of the end-product to the stakeholders". OUM has 16 disciplines, extending RUP with Performance Management, Data Aquisition and Conversion, Documentation, Organizational Change Management, Training, Transition, and Operations and Support. OUM also adds an additional phase (Operations) to the RUP.

- Enterprise Unified Process (EUP) (Ambler et al., 2005) extends the RUP by including new phases
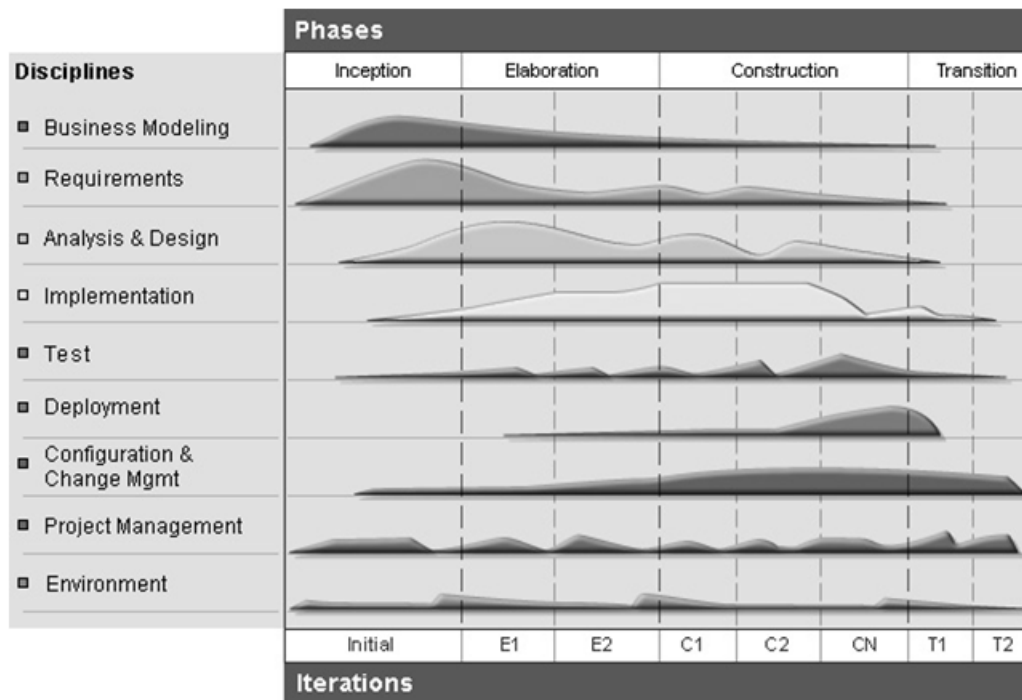
Figure 3.8.: The RUP (IBM Rational, 2006).

(Production and Retirement) and new disciplines (e.g. Operations & Support, Enterprise Archi-
tecture) to better address the complete life-cycle of business software inside an organization.

Figure 3.8 depicts the RUP. RUP has four consecutive phases (Inception, Elaboration, Construction,
and Transition) each ending in a milestone to evaluate the project maturity. The output of the evaluation
is a decision to advance, or not, to the next phase. Each phase has one or more iterations (depending
on the project characteristics). Each iteration produces an executable prototype (only the prototype
resulting from the first iteration is allowed to be non-executable). In RUP, the final software product is
built by the increments generated at the end of each iteration.

RUP has a set of technical disciplines (Business Modeling, Requirements, Analysis and Design, Im-
plementation, Test, and Deployment) that contain the activities necessary to develop technically the
software product. In addition, RUP also includes management disciplines (Configuration and Change
Management, Project Management, and Environment) to organize the non-technical activities.

The rounded shapes of Figure 3.8 express the estimated of effort needed to performe each discipline
during the life-cycle of a standard project (e.g. the Business Modeling discipline requires much effort
in the Inception phase and only residual effort in the Transition phase).

3.3.3. Methodologies to Deploy Business COTS Software Systems

To deploy COTS business software systems into organizations, software vendors or consulting com-
panies develop and use proprietary methodologies. These methodologies focus on tailoring a COTS

system to the needs of a specific client, in most cases, assuming that the current business operations of the client should not be deeply questioned. With this approach to the business software development, during a COTS implementation project the client organization cannot be sure that the software product implements the most appropriate set of business processes. The client organization cannot know if the business processes implemented in the COTS systems follow excellent BP-RMs. In addition, there is no guarantee that the effort needed to upgrade a subset of the implemented business processes is manageable for the organization.

Typically, the supporting tools for a COTS system load, unload, extend, or customize business processes, but only focusing on implementation issues, rather than having a business-oriented perspective. Some platforms, like ARIS (Scheer, 2000), provide a basis for editing the software system that implements the business processes, usually following proprietary BP-RMs. Information systems projects developed with such mindset may fail due to the inappropriate understanding that the implementation is mostly related with IT, where the client, either internal or external, does not participate actively and does not have assigned responsibilities. Thus, to avoid software errors and failures, some best-practices can be used, like providing the adequate resources to the development team, or understood the project as being organizational and not only IT related (Beatty & Williams, 2006).

COTS software vendors propose methodologies, like the Oracle PeopleSoft Enterprise - Rapid Start (Oracle PeopleSoft, 2007), the Microsoft Business Solutions - Navision Rapid Implementation (Microsoft Navision, 2005), or the SAP AcceleratedSAP (Daneva, 2003), that focus on the proper implementation of their own COTS technology, assuming that they will satisfy the clients' needs. Just considering the names of the presented methodologies, one can realize the importance of the time needed to implement a COTS system.

Currently, the major software vendors provide solid solutions to support in business software systems the execution of business processes, usually recurring to COTS systems (e.g. ERP, CRM, and SCM). Normally, the technologies used inside those proprietary COTS systems are hidden from the users, and, in the best cases, standard interface technologies (e.g. web services) are available to interact with them. In addition, when the set of business processes implemented inside the COTS system is not sufficient to support the operation of a company, a significant effort to extend the implemented processes is required, either in new projects or in reengineering contexts. The customization of a COTS system requires a deep knowledge on the specific options it offers. Additionally, proficiency in very specific programming languages (e.g. ABAP in SAP R/3) or in the COTS specific business model and interfaces is usually demanded in order to support the scripting inside the COTS system or to execute a proper external use of interfaces.

Automation efforts, like deriving executable business processes (Weber et al., 2008), are crucial to help achieve a reduced implementation time. An additional approach to reduce project time is to use iterations in order to provide the most needed parts of the solution as soon as possible, evaluated by the client, and complemented with later additions.

A business software project is always an improvement opportunity for an organization, which can be materialized in a set of renewed or newly created business processes.

3.4. Conclusions

Organizations must fulfill the expectations of their stakeholders. Organizations should address the needs of their clients and, at the same time, create value to their shareholders. The implementation of such simple principles has huge impacts on the internal structure of an organization. To address this concern, business processes are a proven solution, and required by many standards, to organize properly the activities focused on the expectation of clients, and thus satisfying stakeholders and shareholders.

To support a process-oriented environment in an organization, the development processes for business software need to address explicitly business processes as the major input for the definition of the software requirements. The maturity of the support of a business software system to the business processes is a major advantage for an organization. Additionally, IT staff and business process experts roles are increasingly merged, thus facilitating that those professionals can use and configure the behavior of business processes and business software. At the end, the capacity of a business software system to easily integrate, manage, embody, and mimic real world business processes is the most important characteristic of a business software system supporting a process-oriented organization.

In the business software domain, it should be possible to create a customized software system recurring to software components developed by different suppliers, yet maintaining the coherence of the complete system. Each piece of software should implement a business function in an adequate manner, and, for that, BP-RMs are of major importance to provide business semantics to software components. The traceability of a business function in a business software system should be assured, as well as the capacity of interchanging parts of the software system without disrupting the software operation or its coherence.

It is of the best interest of an organization to have the best available software components orchestrated in the best possible way, independently of the organizations that developed those components. The ultimate goal of a business software solution is to serve the organization where it will run, and not to have a complete software suite provided by one vendor if the latter cannot prove better than the former.

Software patterns are valuable knowledge resources to allow good designs and architectures. When developing software systems, software engineers should use proven solutions and tailor them to the specific challenges they face by using patterns. In the same manner, to streamline software development activities, software frameworks are important reusable resources, in many cases for free. Software frameworks provide customizable implementations to developers to create the most suitable software solution for an organization.

The way developers conduct software development activities is an important contribution to the final quality, costs, degree of fulfillment of requirements, and used resources of a project. A proper and tailored software development process together with a proper human resources staffing are the most decisive factors to have successful software products.

Currently, methodologies to implement business process have the following problems:

1. methodologies are not holistic, because they do not always start with business content and do not always end up in implemented software;

2. methodologies are not directed to use mixed vendor software components, nor to use OSS;

3. the time to implement a business software system based on non-executable business process model is long;

4. there is a lack in the automation in model transformation activities;

5. it is hard to pick a preferred BP-RM and use it inside an established methodology;

6. many methodologies are proprietary and induce activities caused by the characteristics of a proprietary COTS software;

7. a clear focus on the quality of the business processes content is missing. Instead, we observe a focus on the targeted COTS software.

Due to the presented problems, in this thesis, we propose a holistic methodology to transform business requirements into running software. We also present integrated proposals based on BP-RMs, software frameworks, software components embodying business content, and analysis and proposals of business software systems designed and architected recurring to patterns and having as ultimate goal to provide the best possible solution to support properly the operations of process-oriented organizations.

Part III - Contribution

4. Action Research in Action

> "There is nothing so practical as a good theory."
>
> Kurt Lewin

This chapter presents the Action Research method, and mainly it discusses its usage for the research on business software development. It is also explicated a proposal to evaluate Actions based on the EFQM RADAR assessment, combined with metrics for the resulting software product, and evaluation of the stopping condition for this Action Research study. Finally, we also present an initial Action, the PWage project, which acted as the trigger to use the Action Research approach and as the main motivation to continue the research on how to develop software for process-oriented organizations.

4.1. Introduction

Action Research method was originated in social sciences and developed by Lewin (1947) as a research approach of field theory. Trist (1976) views it as a psychosocial equivalent of Operational Research. Action Research has an iterative nature because it is conducted through several research cycles. As stated by Myers (2008), typically Action Research is an iterative research process that capitalizes on learning by both researchers and subjects within the context of the social system of the subjects.

Action Research is concerned to create organizational change and at the same time to study the process to create it (Baburoglu & Ravn, 1992).

These two characteristics make Action Research studies suitable to be applied in the research inside the domain of busines software development, primarily, because organizations expect changes during software implementation projects (e.g. business improvements), and in second place because projects of business software always incorporate human-related aspects, converting the analysis of the project environment and the analysis of the applied process important areas to derive conclusions and improvements for following projects.

According to Avison et al. (1999), "there is still a lack of detailed guidelines for novice researchers and practitioners to understand and engage in action research studies in terms of design, process, presentation, and criteria for evaluation". For that, in this Chapter we explain how we design, execute, and evaluate the Actions. Later, in Section 8.1, we evaluate the use of the Action Research method for

software development activities.

Since the Action Research is an iterative research method, by exploiting its similarities with Operational Research, we propose to define a stopping condition for the studies. For this PhD project, we define as the stopping condition for the succession of Actions the creation of a software system and the use of a methodology that meets the initial goals of the thesis, stated in Section 1.4.

Later, in Table 5.5 and in Table 6.14 the goals of the thesis are compared with the achievements of each Action to evaluate if the stopping condition is met. The evaluations are "Not Achieved" when targets are not met, "Partially" when some evidence is available, and "OK" when satisfactory and comprehensive evidence is available of meting the targets. The PWage Action is evaluated against the stopping condition at the Conclusion of this Chapter.

To reduce the subjectivity in the evaluation of each Action, we perform the evalutions based on facts. Thus, to check if the the stopping condition is reached, we propose a quantitative evaluation framework (see Section 4.2) later used to evaluate the Actions in our study. The purpose of such framework is to improve the visualization of the Action Research study status, and to relate with the accomplishment of the PhD goals.

According to Baskerville & Myers (2004), the four essential premises of Action Research are:

- all human concepts are defined by their consequences;
- truth is embodied in practical outcome;
- rational though is interspersed with action;
- human action is contextualized socially.

From this four essential premises, a simple procedure on how to conduct Action Research can be derived (see Table 4.1). The aim of this procedure is to guide and sequence the execution of each Action, and to relate each Action with the subsequent one.

Table 4.1.: A procedure to conduct Action Research (based on Baskerville & Myers (2004)).

Step	Description
1	Establish, before hand, the purpose of the action.
2	There must be practical action in problem setting.
3	The theory must be adjusted according to the practical outcome of the action.
4	The reasoning and action must be socially situated.

To illustrate our Action Research study, in Section 4.3 we present a project, Premium Wage, that exposed challenges, and from which were reached conclusions, that lead into the use of an Action Research-like approach. Afterward, Chapter 5 presents the SIIA and the SOL projects, both conceived with the main focus of providing the best possible support to business processes through the usage of adequate software. In these two projects, we analyze the use of a standard software development process, the elicitation of software development techniques and architectures, as well as process im-

provements. Later, Chapter 6 presents the Centauro 1.0 and Centauro 2.0 projects, in which the use of BP-RMs, improved software architectures, and a new methodology to implement adequate software support for business in a reduced time frame are shown. The conclusions of the Centauro 2.0 Action, demonstrate that the stopping condition for the present Action Research study was met, and thus the Action Research study could be concluded. Each one of the five Actions are explicated using the procedure presented in Table 4.1.

Figure 4.1 depicts the chronology of all five Actions together with a rough representation of their effective duration. Further details and measures for each Action are later presented in Section 4.3 and in Chapters 5 and 6.
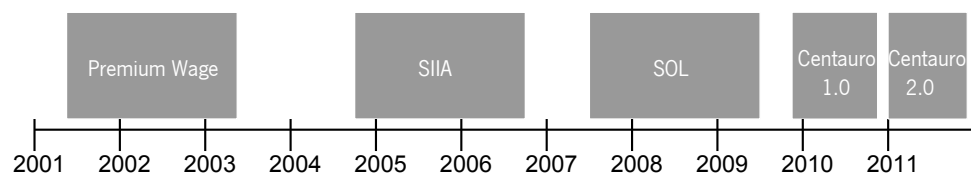
Figure 4.1.: Chronology of Actions.

All five Actions were conducted at Bosch Car Multimedia Portugal (Bosch CMP) (from 1991 to 2008 denominated Blaupunkt Auto-rádio Portugal), by the local information systems group. Bosch Car Multimedia Portugal is a full owned subsidiary of the Bosch Group, it is located at Braga, and accommodates approximately 2000 workers. For each project, a team was setup, composed of both IT and business experts. All team members were Bosch CMP internal workers, in which the author of the thesis is also included. All project activities were performed internally at Bosch CMP without outsourcing.

4.2. Applying Action Research to Business Software Development

To evaluate Action Research studies, Myers (2008) states that, in first place, each study must demonstrate a contribution or a potential contribution to the practice. Even in the case that the action proves to be a failure it can also be considered a contribution. Examples of contributions from Action Research can be a change in a business process or the creation of a software system to support a business process. In second place, each study must contribute to the research (the "theory"). The evidences of positive evaluations for the two claims can be, for the first statement, collected from the changed organization (e.g. in the form of a written letter stating that the action was worthwhile), and for the second statement, from the scientific community (e.g. in the the form of accepted peer-reviewed publications related with the research contributions).

Due to its inherent characteristics of promoting practical demonstrations for research theories, the Action Research studies can also increase the practicality of business related research, frequently

associated with more theoretical approaches. The practicality of research approaches is appreciated by organizations implementing software to support their businesses, mainly due to permanent time constraints in the business environment, like when reducing the time-to-market during the development of new products. In addition, business organizations do not always embody proper human resources to transform more theoretical research outcomes into immediate business value. For that, in the context of research to support organizations (either process oriented or not) by an adequate development of software systems, the Action Research can be a valuable research method since it always creates practical working footprints (e.g. in the form of a running software product).

To maximize the value of the Action Research use, we must assess how we conduct each one of the Actions, and not only putting the focus on produced results by each Action. Such assessment of each Action allows a better planing for the subsequent Action, and thus, allows the Action Research executioner to complete improvement cycles during the conduction of an Action Research study.

To accomplish a sound approach for the assessment of the execution of each Action, we propose the use of an EFQM RADAR-like assessment (EFQM, 2010) (See Figure 4.2). The EFQM excellence model and its RADAR assessments are proven industry models (EFQM, 2011) to guide organizations in their path to organizational excellence. One of the main purposes of the EFQM excellence model is to propitiate an holistic view of each organization. Such holistic view is decomposed into criteria and each one of the criteria is evaluated recurring to RADAR assessments. Due to that fact, RADAR can be used to properly evaluate organizations (or parts of them), processes, and even results included in Action Research studies.
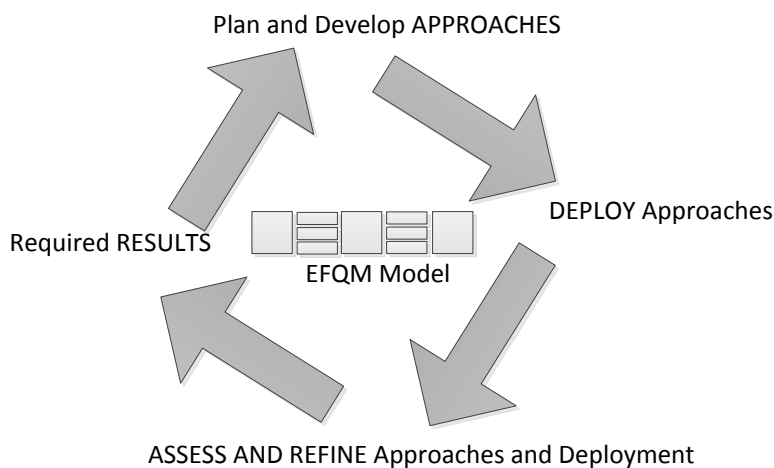
Figure 4.2.: RADAR assessment framework from EFQM 2010 model (adapted from (EFQM, 2011)).

Since organizational improvement is a value present in all organizations, it can be deployed into the execution of the Action Research studies. RADAR assessments are used by organizations to conduct appraisals, either with internal or external assessors, and finally to score their maturity level in the EFQM Excellence Model. RADAR assessments are used to evaluate each one of the nine criteria present inside the EFQM Excellence Model, namely the five Enablers criteria:

1. Leadership

2. Strategy

3. People

4. Partnerships and Resources

5. Processes, Products, and Services

and the four Results criteria:

1. People Results

2. Society Results

3. Customer Results

4. Key Results

When assessing an organization, the RADAR provides a quantifiable feedback, drilled down by each criteria, that allows the assessed organization to setup proper improvement activities for the next evaluation cycle. RADAR assessments can also be used to spot best practices when performing benchmarking activities. Usually, RADAR assessments inside organizations are conducted once per year.

For the current Action Research study, the RADAR assessment was conducted after the end of each Action. To use a RADAR-like approach in an Action Research study, in first place we suggest to scope its applicability only to the procedure presented in Table 4.1, as the Enabler under evaluation. The procedure of Table 4.1 is related with the fifth Enabler criteria of the EFQM model: "Processes, Products, and Services". The procedure is the single focus of assessment because it is the process executed in our Action Research studies and, for that, it is the most important aspect contributing to each Action.

The same assessment approach could easily be extended for other aspects of an Action Research study, like when evaluating the following practices:

- what was the strategy behind the decision to use Action Research;

- how did the leadership (e.g. PhD orientation) deploys the Action Research approach into the PhD student;

- how the Action Research training and learning was carried on;

- how the results of this Action Research study (either good or bad) were transposed into other studies;

- what results are obtained and what is the positioning of that results compared to benchmark organizations using Action Research.

Nevertheless, and since the focus of the PhD project is not to evaluate the usage of Action Research, we only focus the RADAR assessment on the procedure of Table 4.1.

Table 4.2 presents the attributes to score when using a RADAR-like approach for an Enabler criteria are presented. The quantitative values allowing to assess each Enabler attribute of Table 4.2 are shown in Table 4.3.

Table 4.2.: EFQM RADAR scoring attributes for Enablers criteria.

Element	Attributes to Score	Description of Attribute
Approach	Sound	The extent to which the approach has a clear rationale, defined processes and focuses on stakeholder needs.
	Integrated	The extent to which the approach supports policy and strategy, and is linked to other approaches where appropriate.
Deployment	Implemented	The extent to which the approach is implemented.
	Systematic	The extent to which the deployment is carried out in a structured way with the method used to ensure deployment being itself planned and executed soundly.
Assessment and Review	Measurement	The extent to which regular and appropriate measurement of the effectiveness of the approach and deployment takes place.
	Learning	The extent to which learning activities take place to identify best practices and improvement opportunities.
	Improvement	The extent to which the output from measurement and learning is analyzed and used in order to identify, prioritize, plan and implement improvements.

RADAR assessments also include attributes to score the Results criteria. Nevertheless, the capacity to score results is not the most needed feature to evaluate Action Research studies because, RADAR assesses, for instance, how results are linked into strategy, if we have benchmarking data, checking results trends along three years, or verifying goal definition and achievement, which are not the main focus of the current Action Research study. Also, RADAR does not, and can not, include the definition of an adequate set of metrics to use in each kind of organization and product.

Table 4.3.: Proposed maturity levels to score Actions using a RADAR-like approach.

Maturity Level	Description
4	Comprehensive evidence
3	Clear evidence
2	Evidence
1	Some evidence
0	Not implemented

Instead of using RADAR assessment to measure each Action results, we propose to use the values presented in Table 4.4, where the metrics to compare Actions are shown. In order to get a more concrete comparison framework, we propose ten quantifiable metrics, including their dimensions of analysis and units. The aim of these quantifiable metrics is to evaluate the software development process, the product, and the usage of the product, and afterward to allow the comparison of the Actions in our study. To derive the metrics, we use the PMBOK (Institute, 2004) extended with a new set of metrics to widen the quantifiable scope of the analysis. According to the PMBOK, the metrics that allow project comparisons may include:

- the actual project performance against the project management plan. We propose to measure it by the percentage of implemented functional requirements versus the candidate functional

requirements. We extended this metric by using it for the non-functional requirements;

- assessing the project performance. We propose to measure it by the average performance results during quality assessments occurring throughout the project life-cycle;

- cost. We propose to express it by the percentage of the amount of human effort used during the project, expressed in Person/Days, versus the planned human effort;

- schedule information. We propose to evaluate it by the percentage of the real duration, expressed in calendar days, versus the planned duration of the project.

Table 4.4.: Proposed metrics for the comparison of the results of Actions.

Id	Metric Scope	Dimension of Analysis	Metric Name	Metric Formula	Unit
D1	Development Process	Requirements	Functional Requirements	(# implemented FR) / (# candidate FR)	%
D2		Requirements	Non-Functional Requirements	(# implemented Non-FR) / (# candidate Non-FR)	%
D3		Quality	Quality Assessments	average of achievement on quality assessments	%
D4		Cost	Effort	(used effort) / (planned effort)	%
D5		Time	Duration	(real duration) / (planned duration)	%
P1	Product	Complexity	SLOC	# physical source lines of code	SLOC
P2		Complexity	Used Software Frameworks	# used software frameworks	USF
P3		Soundness	Change Requests	# change requests in the first month after the go-live	CR
U1	Usage	Reliability	MTTR	mean time to repair	min
U2		Reliability	MTBF	mean time between failures	min

Additionally, we propose to include the following metrics to measure the results of each Action:

- the complexity of the resulting product should be taken into consideration. For that, we will use two complementary metrics. The first metric is the number of developed physical source lines of code (SLOC), casting off empty and comment lines. Physical SLOC is used instead of Logical SLOC because the first are straightforward to count. This decision is also based on the fact that the development teams were mostly composed of a common set of persons, hierarchically always under the same manager, and subjected to the same development guidelines. The second metric is the number of software frameworks (understood as major software components) used in the software product. The relevance of the second metric is due to its capacity to express the complexity of a software product, when using the customization of software frameworks instead of developed code;

- the soundness of the product after the go-live, expressed by the number of requested changes

during the ramp-up phase. We considered the ramp-up phase as the first month of use. This metric shows the gladness of the requester and the supporting team with the running software system;

• the reliability of the product after the go-live, evaluated by the mean time to repair it and also by the mean time elapsed between consecutive failures, again considering the first month of use and considering failures that stop the complete system or prevent the system to support adequately the business (e.g. a problem requiring a reset on a mobile client is not considered as a failure).

4.3. An Initial Research Trigger: The Premium Wage Project

The Premium Wage (PWage) project, despite being conducted before the time frame of the PhD project, was an important trigger to start the research on how to improve software development processes and on how to check the adequacy of the generated software products for process-oriented organizations. For the purpose of our Action Research study, we consider the PWage project as a pre-Action, since it was not executed explicitly with an Action Research mindset neither was the result of an explicitly planned Action. Nevertheless, its results and lessons learned were seminal for the shaping of subsequent projects.

4.3.1. Purpose of the Action

PWage project was conceived without an Action Research approach. Even so, PWage was one of the case studies of our MSc. Thesis, Software Engineering Oriented to (Business) Processes (Duarte, 2002). At that time, the project was setup with the purpose to include also the analysis of the use of a software development process, RUP (Jacobson et al., 1999), in the development of software for a process-oriented organization and to test it in a non-trivial social environment. Prior to the start of the PWage project, RUP was chosen as the software development process. We took this decision not only because RUP is an iterative and incremental process, but also because it has well defined guidelines, it is configurable and adaptable to different project types, and it also includes a business modeling discipline. All these characteristics of RUP were considered relevant by the software development organization, and thus RUP was elected for the PWage project.

4.3.2. Practical Action

The PWage project was requested to the IT department targeting a major functional requirement: to provide support for the calculation of a monetary prize payed to blue collar workers when the defined criteria for quality, absenteeism, and productivity are reached. Prior to the PWage project, software

existed that was able to retrieved data from quality, absenteeism, and productivity activities, but without the proper integration, detail, and scope. At that time, the quality, absenteeism, and productivity activities related with the payment of monetary prizes were considered to need a closer observation and, probably, some restructuring. The PWage project designed extensions for the quality and the absenteeism activities and completely redesigned the productivity activities.

4.3.2.1. Implemented Business Processes

Prior to the PWage project, the team leaders of each production team already maintained data about products and production lines. They also maintained the data about the team members. The calculation of the productivity of each production line was performed in cooperation by the team leader and by the controller. The line productivity calculation included the business function of maintaining products, production lines, and production teams information. Figure 4.3 presents an UML use case diagram expressing the top level business use cases existing before the PWage project. Prior the PWage project, there was no systematic payment of monetary prizes based on quantifiable production-related criteria. All data was kept on non-communicating databases, either end-user or departmental.
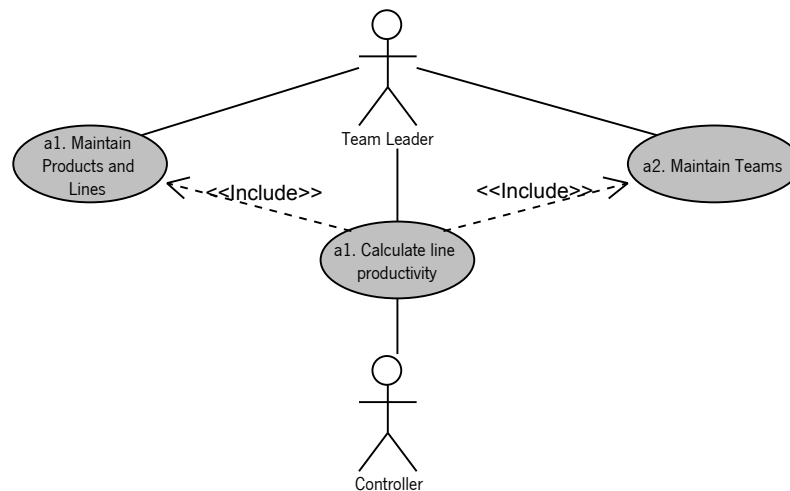
Figure 4.3.: Business use cases for PWage - current (as-is) situation.

The rules to derive the payment of a monetary prize to workers based on quantifiable criteria are derived from a benchmarking activity performed with a Bosch plant located in Malaysia. However, the business process to calculate the resulting values and the payments, as well as the data sources, were locally designed.

To cope with the PWage candidate requirements, a new set of business activities needs to be added to the existing ones. The maintenance of products, production lines, and production teams data, performed by team leaders and controllers remains the same at the more abstract level, but is subject to refinements and new sub-activities are added. Furthermore, a new business use case, extending the calculation of line productivity, is needed because a detailed calculation of the individual perfor-

mances is necessary to accurately award the workers. Moreover, the calculation of the quality and the absenteeism individual performances needs to be carried on by the respective data owners. Both are new business activities. The calculation of the individual monetary prize is performed by the industrial engineer and validated and payed by the wages manager. It is also a new business activity. Figure 4.4 depicts the top level business use cases for the to-be situation after the PWage go-live.
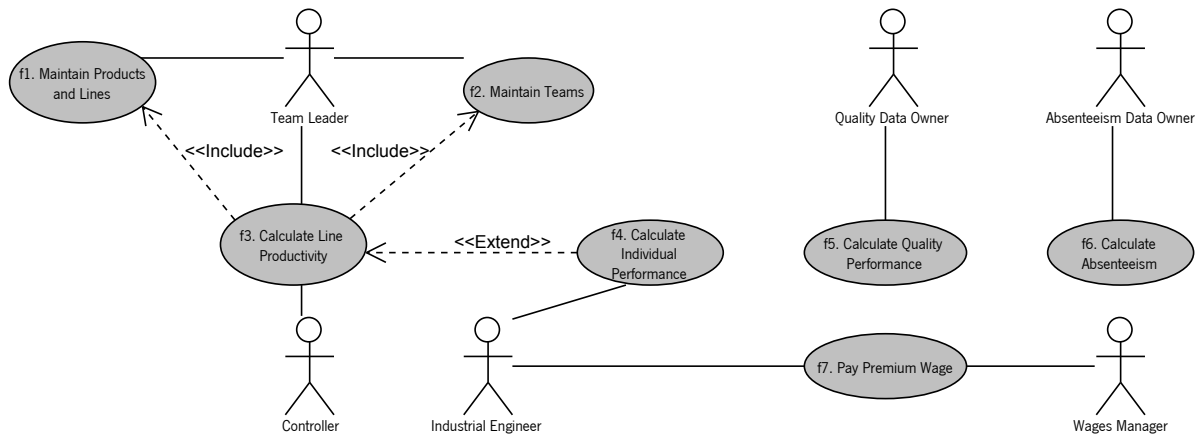
Figure 4.4.: Business use cases for PWage - desired (to-be) situation.

Complementing the functional business requirements, Figure 4.5 depicts an UML activity diagram showing an important part of the PWage mapped business process, related with the calculation of the productivity for each line. This activity diagram is a specification for the business use case "f3. Calculate Line Productivity" of Figure 4.4. The sub-process starts by forking into recording the presences of workers, exporting the master data related with each worker from the human resources system, updating production and production data, exporting produced quantities, and exporting production times. The data exported from the human resources system is afterward used to create the production teams data and then to check if workers exist not allocated to any team. Also, the produced quantities data is checked for errors (e.g. missing booked quantity for products). The exported general production times are detailed to a production line level. All these five threads are joined before the calculation of the line productivity.

In (Duarte, 2002), we provide a more detailed and complete specification for this business process, including the artifacts resulting from the execution of the discipline "Business Modeling" of RUP. These set of artifacts includes the business object models using stereotypes proposed by RUP to extend UML capacity to express business related models.

4.3.2.2. Development Project

PWage project is conceived with RUP as the software development process. PWage project is staffed by two IT and two business headcounts. Among other RUP roles, the author of this thesis acts as project leader, software architect, and programmer. PWage project includes a steering committee
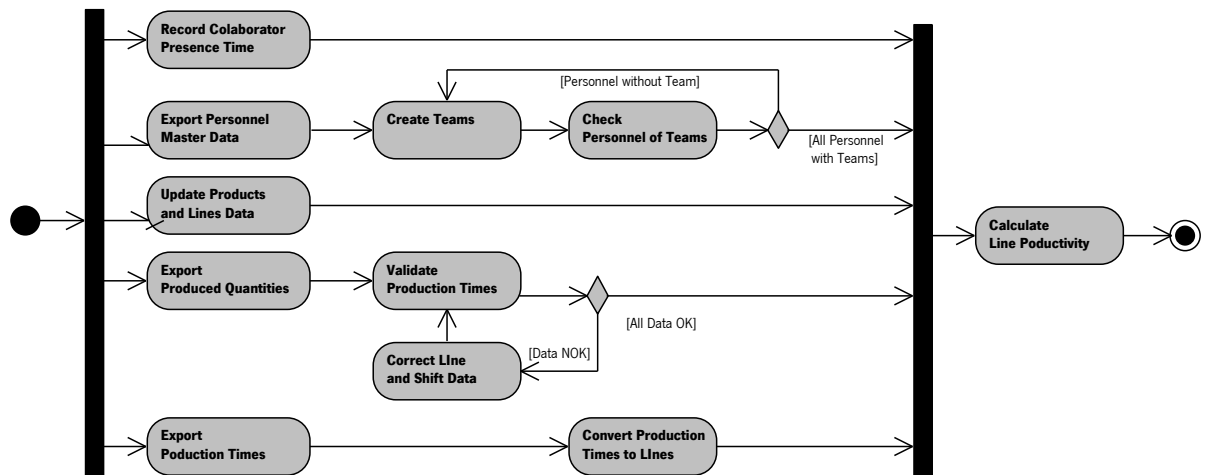
Figure 4.5.: UML activity diagram for PWage line productivity calculation.

responsible to check the project status and to sponsor the project leader when solving potential road-blocking problems. The steering committee includes stakeholders representing all business functions needed to execute the payment of the monetary prize.

The main risk of the PWage projects is a possible missing budget to pay the monetary prizes. The decision to pay a monetary prizes was taken by a former management team, meanwhile replaced, without a rigorous calculation of the average monthly budget needed to pay the prizes. In the case that the problem stated in the risk occurs, the PWage project can be canceled. To monitor the probability of occurrence of the risk, during the course of the project monthly calculations should be made, either manually or using some, meanwhile, developed software prototype. Such calculations should be presented to the steering committee, together with an explanation for the premisses and requirements for the project, in order to get the steering committee commitment to pay the calculated monetary prizes. In the case of PWage project cancellation, the meanwhile developed software interfaces will, nevertheless, bring better data quality to the already existing business functions. The second most important risk is related with not getting an adequate contributing effort from all needed business functions in order to allow the calculation of monetary prizes. For each worker that participates in the already existing business functionality some extension to his normal operations are requested, which do not directly brings business value to each one. The new functions are only adding value for the calculation of the monetary prizes. Since the calculation is also based in the data collected from other systems, it is crucial that all data is properly maintained. The absence of proper data would cause a distortion in the monetary prizes and thus causing social problems between workers and management. The social problems would endanger all the positive motivational impact intended by the management. Also, if a significant number of errors occurs during the calculations for the first months, the PWage software system would never be used again. To measure the risk probability of occurrence, data quality audits in the existing data are setup before the system goes live. To contain the problem, it is also decided to get a communication from the management into the data maintenance related workers stating the

importance of adequately maintaining the data, as well as the nomination of a data quality verifier after the system goes live. If problems in the calculation of the monetary prize, or a low confidence degree in the existent data is perceived by the steering committee then the payment should be suspended. The steering committee decided that it is preferable not paying the amounts with mistakes in the actual month and postpone the payments to the following month, rather than paying amounts in the current month an then correct them in the following month.

4.3.2.3. Resulting Software System

The PWage software system was developed with Microsoft Visual Basic 6.0 (Corporation, 2011) language. It features a three-tier design and architectural pattern. Figure 4.6 depicts the seventeen top level system use cases. This UML use case diagram shows two system boundaries because the resulting software product is composed of both new and reengineered software. Inside the PWage system boundary are the new use cases and inside the ProData system boundary are the reengineered use cases. In the PWage project, the reengineered ProData system is also considered a deliverable. The business use cases depicted in Figure 4.4 are directly mapped into the corresponding system use cases. Additionally, in the use case diagram of Figure 4.6 three external systems (WinVQ, SAP HR, and Production Times) appear as actors due to the necessity to integrate parts of their data.
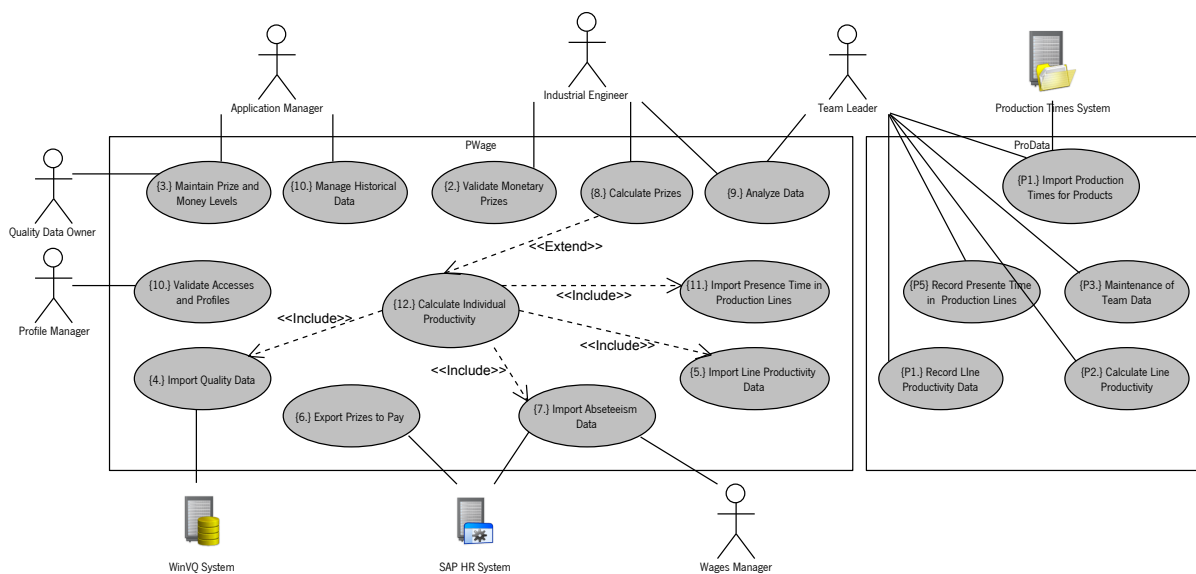
Figure 4.6.: PWage top level use cases.

We used the 4SRS technique (Fernandes & Machado, 2001) to transform the UML system use case model into an UML class model. Instead of empirically infer what design classes should support the functional requirements, 4SRS basically proposes the following sequence of activities:

1. for each system use case, create three objects: Interface, Control, Data;

2. from the full set of objects generated, only the meaningful ones are kept, depending of the

description and of the use case realization;

3. aggregate the similar remaining objects (e.g. different user interface objects that can be merged into a single one to group basic data operations);

4. create associations between the single objects and/or the groups of objects.

Later on, all single objects and groups of objects are transformed into classes using a basic 1:1 transformation. Afterward, a mapping from the classes into a three-tier design is inferred based on the object main characteristics. The objects tagged as Interface, Control, or Data, are respectively distributed into one of the tiers, namely to Presentation, Business, or Data. This technique provides a mechanization that leads to time improvements to reach the final software system design and architecture.

4.3.3. Theory Adjustments

In the next Sub-section, we present the evaluation of the Action Research procedure, and we present the results related with the PWage project and with the resulting software system, along with the lessons learned. On the following Sub-section 4.3.3.2, we present the proposed adjustments derived from the PWage project to improve the way to conduct the software development activities for process-oriented organizations.

4.3.3.1. Metrics and Lessons Learned

Despite the PWage project is setup without a complete Action Research mindset, the data to evaluate the maturity levels of the use of procedure of Table 4.1 is available. As such, Table 4.5 presents the evaluation for that Enabler. The most distinctive parts of this evaluation are the low scoring on Measurement and the top scoring on the Learning and Improvement attributes. The former is mainly related with missing a regular and appropriate measurement system for the effectiveness of the procedure to conduct Action Research. The Learning and Improvement attributes high scores are due to the extension of the learning activities that took place and due to the proposed improvement opportunities, later on detailed in Sub-section 4.3.3.2. The learning and improvement activities are validated by peer-reviewed scientific content, namely by a conference paper, "Using RUP for Process-Oriented Organizations" (Fernandes & Duarte, 2004), and by a journal paper, "A Reference Framework for Process-oriented Software Development Organizations" (Fernandes & Duarte, 2005). These works propose a reference model for process oriented software development organizations based on the value-chain (Porter & Millar, 1985), and propose and discuss the use of the disciplines of RUP as instantiations of the business processes inside the reference model. The resulting BP-RM for software development organizations is listed and analyzed in the "RMK Reference Models Catalogs Project" (Loos & Fettke, 2010) as "Reference Model of Fernandes/Duarte" (see Table E.3 in Appendix E for a detailed characterization). To create this BP-RM, we mapped the RUP disciplines into a generic process reference model. The papers also include proposals for the specification of the business processes

contained in the BP-RM where RUP does not provide guidance, like the Human Resources business process. The papers also point out a governance model for the reference model of Fernandes/Duarte.

Table 4.5.: Maturity levels of attributes of the Enabler in the PWage Action.

Element	Attributes to Score	Maturity Level
Approach	Sound	3
Approach	Integrated	2
Deployment	Implemented	2
Deployment	Systematic	3
Assessment and Review	Measurement	1
Assessment and Review	Learning	4
Assessment and Review	Improvement	4

In Table 4.6, the values for the metrics of Results are shown. The client expressed seven top level business use cases (shown in Figure 4.4) transposed into seventeen top level system use cases (depicted in Figure 4.6). All system use cases are implemented, leading to an 100% result for the D1 metric. The stated candidate non-functional requirements are that the full calculation of the individual prizes should takes less than one hour, and that the result of the calculations should be kept during three years. The first non-functional requirements is validated through appropriate performance and load tests. Additionally, the calculation results should be considered classified data. This requirement is guaranteed by the fulfillment of the security guidelines existing within the organization, which is considered sufficient by the stakeholders. It is also requested that the system should receive the attention of a supporting hotline. This fourth non-functional requirement is considered implemented by using the existing hotline service of the IT department. The fifth non-functional requirement stated is that the functions should only be accessed by selected profiles due to its confidential nature, and that role granting for the software system should only be managed by restricted and identified personnel. This last non-functional requirement is implemented by a separate software system, already in use inside Bosch CMP, able to satisfy this requirement. For that, the D2 metric was also evaluated with an 100% result. During the project life-cycle we perform two quality assessment. The initial quality assessment scored 100% and the second quality assessment scored 80%. The latter value is calculated by 53 statements with positive evaluations out of the existing 66 (in the quality assessment checklist). The main negative points of the second quality assessment are related with the fulfillment of costs and budget, and with missing a complete functional testing specification. This second quality assessment corresponded to the assessment before the go-live of PWage system. For that, the D3 metric has a 90% evaluation. The cost metric, D3, suffered a significant negative deviation. The first effort estimation is 40 PD but at the project end 60 PD are used, causing a deviation to the planned figure of 50%. The deviation is mainly related with the constant changes in the concepts and getting their acceptance inside Bosch CMP. The time frame is fully fulfilled, resulting in an 100% result for the D5 metric. The resulting software system has 1132 SLOC (P1 metric), but not considering the reengineering effort of the ProData system due to the actual complexity of counting the changed lines of source code. The

PWage system does not use any software framework (P2 metric). The last three metrics (P3, U1, and U2) are not calculated.

Table 4.6.: Values of the Results in the PWage Action.

#	Dimension of Analysis	Metric Name	Value
D1	Requirements	Functional Requirements	17/17 = 100%
D2	Requirements	Non-Functional Requirements	5/5 = 100%
D3	Quality	Quality Assessments	average (100%, 80% (53 out 66)) = 90%
D4	Cost	Effort	60 PD/40 PD = 150%
D5	Time	Duration	24 days/24 days = 100%
P1	Complexity	Physical Lines of Source Code	1132 SLOC
P2	Complexity	Used Software Frameworks	0 USF
P3	Soundness	Change Requests	not evaluated
U1	Reliability	MTTR	not evaluated
U2	Reliability	MTBF	not evaluated

The results expressed in the last three metrics of Table 4.6 are crucial to fully understand the PWage project. The key point about PWage project is that it was never used in a productive environment. This means that all project reached a positive go-live status, expressed in the pre-release result of 80% in the second quality assessment, but it was never used by end users after go-live, mainly due to the social environment surrounding the project, later explained in Sub-section 4.3.4.

The lessons learned for the PWage project are:

1. in matrix organizations, where departments and business processes co-exist, some problems, external to the clients and to the software development teams, arise just because of the existence of departments. Since departments are among the stakeholders of a project, they usually block the software development, when it can threaten the current business activities running inside those departments or even questioning the existence of some departments;

2. all stakeholders should be aware of the tasks related to the usage of a new software product. In this way, all stakeholders, as suggested by RUP, should know and validate all the requirements;

3. there is a big difference between real world business processes and their models, either visual or text-based. A good understanding of the business processes together with an adequate and updated documentation are essential to build software systems with quality;

4. the requirements to extract data from business processes should be sufficiently explicated;

5. The responsibles to fully state the main functional requirements should be identified and available. The PWage project missed a responsible to express the calculation formulas for the productivity, leading to a definition of the formulas by the project team, later disapproved by the stakeholders. All business process stakeholders must be involved in the software development for the scope of business processes related with their responsibilities. It is also important that all stakeholders are aware of the tasks related to the use of a new software product. This way,

all stakeholders, as suggested by RUP, should know and validate all the requirements.;

6. RUP is able to provide sufficient flexibility to allow the customization, despite empirical, of a software development process to a particular project;

7. the business modeling language used in business software development (e.g. RUP stereotypes for UML) should be able to support easily the automation of model transformations until a software execution state is reached;

8. the use of RUP, even tough in the first project inside an organization, it is not a delaying factor for the project;

9. RUP proved useful to model and map business processes into software applications for a process-oriented organization;

10. visual languages, namely UML diagrams, are a proper mean to communicate with other project stakeholders, even in the case of non-IT personnel;

11. the use of BP-RMs can shorten the duration of the requirements activities;

In a more abstract level of analysis, the main lesson learned is that the external social environment where the project is being executed is of major importance for the project success. The PWage software product, despite not perfect, was able to implement properly all the functional and non-functional requirements. Despite being an adequate product, and also profiting from a good project internal social environment, the PWage system did not reach a full use by the end users because of its external social context.

4.3.3.2. Proposed Adjustments

The main adjustment on how to conduct projects related with the development of software for organizations, either process oriented or not, is the explicit incorporation into the project of stakeholders with sufficient empowerment to conduct all the needed project activities. This adjustment is caused by the necessity of controlling, as much as possible, the external social context of the project. For that, on the next Actions, the external social environment should be explicitly designed and their associated risks should be evaluated, since the early stages of the project.

At the time of the PWage project, we proposed in (Duarte, 2002) and later in (Fernandes & Duarte, 2004,0), the following adjustments on how to develop software for a process-oriented organization:

- the creation of an environment that allows the execution, test, validation, and simulation of business processes based on business process models. Such environment can also acts as a tool to compare the current (i.e. as-is) and the future (i.e. to-be) business process scenarios. Business performance indicators can be used to compare the business processes at duistinct time frames, making the environment a valuable tool for business modeling decisions on future scenarios;

- business process models should be detailed and updated. A detailed model could allow an

organization to compare with other organizations, and thus acting as a basis for business process reengineering. An updated model expresses the current real world business process situation, and thus prevents wrong decisions based on the model. Additionally, one of the main assets for a software development organization is a detailed and updated business model of the target organization because the tasks following business modeling, like programming or system testing, tend to be more easily automated and executed with supporting tools.;

- RUP should embody an auxiliary process to help on its tailoring. Such tailoring should not only be based on empirical decisions. The customization process should consider the size and format of the organization that will use the software product, the size and format of the organization that is developing the software, and some aspects related with the project under consideration. Such a tailoring process would be a valuable asset for software development organizations missing RUP knowledge. Some years later to PWage proposed improvements, IBM released the Rational Method Composer which addresses this concern (Haumer, 2005) proving the validity of such claim;

- template specifications for software development organizations, possibly integrating known best practices (e.g. for CMMI-DEV (Team, 2010)), should be created. This way, the internal structure of the software development organization provides more guarantees for the quality of the software product and to its clients. Along with the internal structure, the used software development processes and the quality of collaborators are fundamental factors to achieve top quality in software products. The template could also act as a tool to enhance the transition of existing organizations into improved internal models.

4.3.4. Social Environment

The PWage project is setup with the expectation that an increase in productivity, quality, and a reduction in the absenteeism can be achieved by Bosch CMP. PWage is considered a tool to motivate blue collar workers because a direct consequence of their performances would be immediately perceived by the workers in the form of a monetary prize. Nevertheless, PWage system was never used.

The PWage project modeled a business process that has a large amount of stakeholders. This amount of stakeholders caused the elicitation of contradicting requirements. The PWage project team had only two business stakeholders, not representing all the important stakeholders. These two project characteristics, together with the deployment of the final decision on what should be the final format for the process of paying a monetary prize to blue collar workers, from the top management into a lower organizational level, caused a social system where all stakeholders are at the same organizational level, thus, causing difficult and contradictory decision making. Clearly, a process mindset is missing and a prevailing departmental mindset is frequently used to state requirements.

The PWage project embodied a steering committee that should have handled problems related with the

social environment of the project. The root cause for the steering committee to fail on this task is the rotation on the management stakeholders and consequently on crucial mindsets on how to conduct the calculation of the monetary prizes payed to the blue collar workers. The PWage project also suffered a significant negative impact by other project with higher priority that was running in parallel inside Bosch CMP. During the last months of PWage project, the implementation of an ERP was conducted in an overlapping time schedule and used common human resources, causing a deviation on the attention of the management stakeholders away from the PWage project.

Several attempts to start PWage system use were later performed, but a new requirement was added to the PWage system: the decision to broaden the scope of the payment of the monetary prize to all workers, despite being blue collar or not. This new scope violates the business requirements stated for the PWage project, making the developed PWage system inadequate to be used in the new business context. Of course that the PWage system could have been adapted for the new business context, but by broaden the scope to all workers results in less demanding requirements for the calculation of the monetary prizes. The individual productivity factor was dropped for the calculation, and the remaining quality and absenteeism factors could be easily manually calculated. The trade-off between using manual calculations, even with more errors and monthly time consumed, and the usage of a reengineered PWage system, a hot social system, derived in the decision to use manual calculations.

4.4. Conclusions

Each research study needs to be supported by a defined approach and method. It is important that the research method is adequate to the problem being dealt. The selection of a proper research method, among all the available, is currently an exercise of empiricism, despite available tips on the adaptability of each research method to the different generic types of problems. The extra complexity comes from the specific characteristics of each research project.

In this chapter, we explain because the Action Research approach is adequate for the development of business software. We propose a framework to help evaluating Action Research studies based on a proven assessment model, the EFQM RADAR. Our proposal uses an EFQM RADAR-like approach to evaluate how the Action is being executed and a proposed set of metrics to evaluate the results of each Action. Our believe is that proven models can be reused in different contexts. We achieve this claim by using the EFQM RADAR model on Action Research studies. We also consider that quantifiable metrics allow better comparisons of Actions inside the same Action Research study or even between different studies. By using quantifiable indicators, it is easier to derive better conclusions and to manage the improvements for the next Actions based on facts and not only in diffuse perceptions. Nonetheless, to compare Actions we need more than quantifiable indicators. General qualitative analysis is also needed because metrics can't cover all the analysis scope of Actions.

In spite of the fact that we did a post-mortem analysis of the PWage project, the analysis of this Action

was not difficult. For that, the availability of data is essential. Also, this first use of the comparison framework proved that, even a decade after the project realization, some new facts derived from the analysis could be perceived. The Enabler and Results analysis helped to better understand why the project failed to have a proper go-live.

As stated in the Section 4.1, for each Action we must evaluate if we reach the research stopping condition, in the present case the thesis goals. For PWage project, the main thesis goal, "to propose a methodology to generate immediately executable information and software systems for process-oriented organizations, starting with models of business requirements" is not reached, because neither we proposed such methodology (we used RUP), neither the business process are able to be immediately executed in a software system. For the secondary thesis goals, we evaluate SG1, the resulting software products must have quality, as positive since it met functional and non-functional candidate requirements. The SG2 goal, the proposed methodology should be able to be executed in a short project time, is evaluated as negative because we did not propose any methodology. SG3 goal, to specify and to implement a software framework to support the development of software with the methodology, is also negatively assessed because we did not create any software framework. The fourth secondary goal, SG4, the resulting methodologies and software products should be able to be used by business experts without the intervention of software experts, was not achieved because PWage project needed intensive work from IT staff to create business models. The SG5 goal, the proposed methodology should be holistic, ranging from business process models to running software, was not accomplished because we had to create the business process descriptions. Finally, SG6 goal, to promote the composition of software solutions based on free software and originated in different vendors, was also not achieved because we have only used one proprietary software development environment.

5. Building Software to Support Business Processes

"There is nothing so useless as doing efficiently that which should not be done at all."

Peter Drucker

> This chapter presents the SIIA and the SOL Actions. Here, we build MES business software to support the operations of a processed-oriented organization, recurring to standard software development processes, tools, and modeling languages.

5.1. Introduction

Software developers can use general-purpose development processes and tools to create business software systems supporting some process-oriented organization. In addition, no specific modeling language for business processes is necessary to get business process modeled and implemented in a software system.

For a process-oriented organization, we can develop business software systems without explicitly consider the business processes. By exaggeration, we may even demonstrate that the resulting software system is indeed supporting some business process. If the stakeholders of business software development use such kind of argumentation, they are creating a road-blocker for a proper support in business software of the business process landscape of their organization.

If we extend such type of argumentation, we may also argue that we can use almost any random line of code to support some part of a business process, and this fact does not make that randomly created piece of software a high quality, effective, or adequate support for a business process. If such approach is used, the changes in business processes do not guarantee that the corresponding support by business software is done accordingly, because integrated business process and software governance activities are missing.

Thus, the self-knowledge and the ability to govern a set of business processes and their respective support in business software requires, from an organization, a high maturity level on how to deal with its internal structures and activities and how to put them into practice.

It is difficult to move an organization from a maturity level where the existence of business processes can be demonstrated (including during a quality audit) into a maturity level where business processes are understood, lived, and used as key tools to achieve top business performance and efficiency.

The answers to why we need to use business process-aware software development processes, and use business process modeling and execution languages is showed in the Actions of the present Chapter and of the Chapter 6. We intend to demonstrate that by using explicitly and adequately modeled businesses processes in proper languages, combined with the use of appropriate business software frameworks, development tools, and software development good practices (e.g. design patterns), we can surely help to foster both the software development projects and, at the end, the business performance of an organization.

The Actions presented in the current chapter, SIIA and SOL, share common characteristics, such as:

- being developed with RUP;

- being staffed with almost the same IT experts, despite some distinctive roles in the two projects;

- using UML as the modeling language (including the modeling techniques for business processes);

- using some shared technologies (e.g. the .net compact framework as the mobile clients execution environment).

5.2. The SIIA Action

SIIA is the first MES project setup to support explicitly part of a value-added, and very critical, business processes inside Bosch CMP. SIIA has to support, among others, the business processes related with the corrective maintenance and with the production execution, including the respective data collection and calculation of key performance indicators (KPI) for the surface-mount technology (SMT) production and maintenance functional areas. SIIA has an impact on more than 1000 workers, supports the complete set (ca. 24) of SMT production lines, and has more than 300 users.

The Bosch CMP managers of the production and of the production engineering requested to the internal IT department the SIIA project with the expectation that the resulting software system is a major breakthrough in the execution of production, corrective maintenance, quality of shop-floor related data, and allows a correct and reliable calculation of the KPIs. The SIIA software system is also requested to provide online monitoring of the status of the production lines, as well as automated integration with the ERP (SAP).

5.2.1. Purpose of the Action

The main purposes of the SIIA Action are to assess the development of a MES that supports a value-added business process for a process-oriented organization recurring to standard UML and RUP, and only using general purpose programming languages. Additionally, Bosch CMP requested an evaluation of the suitability of RUP to support the IT development team and the adequacy of RUP to MES development.

The starting point for the SIIA Action are the proposals deriving from the PWage project (see Section 4.3), namely those related with UML, RUP, and development of software for process-oriented organizations. The main differences between the staff of the SIIA and the PWage Actions lie on the composition of the development team and on the quantity of involved staff during the development time.

To fulfill its main purposes, the SIIA Action is setup with RUP as the development process, UML as the language for the modeling activities, uses Bosch CMP as the target organization for the software development efforts (a process-oriented organization where production and maintenance are value-added processes), and also uses the .net framework as the development framework.

5.2.2. Practical Action

The SIIA Action uses a software development process tailored from RUP, recurring to the "Classic RUP (for Large Projects)" template (IBM Rational, 2006). Bosch CMP uses the RUP template for Large Projects because it is the first time that uses RUP, and because plans to use RUP as the process model for software development to base future projects. Bosch CMP choose the Large Projects template to obtain a comprehensive know-how of the RUP activities to later allow a better tailoring based on the explicit knowledge of the RUP detailed proposals.

The tailored RUP-based process also embodies the Bosch guidelines for software development. In SIIA, the business processes are modeled recurring to UML use case, activity, and state machine diagrams (e.g. see Figure 5.2) and their implementation is done recurring to a general purpose programming language running on top of the .net framework execution environment. The business processes are hard-coded in .net programming languages (C# and Visual Basic). These constraints are introduced by the IT management to contain the quantity of changes, namely in the development process, variety of modeling languages, and social environment surrounding the project. The justification for the previous decision is the need to minimize the risks associated with the quantity of new technologies that the IT team has to learn. The SIIA project uses the Microsoft Visual Studio tools to support the technical development activities.

5.2.2.1. Implemented Business Processes

Before the SIIA implementation, operators in the SMT shop-floor were inputting production-related data into paper sheets, which clerks after book to SAP and other software tools (e.g. worksheets). Despite being manual or not, all data input activities were, and still are, fundamental to properly feed the ERP system and to allow an accurate reporting for the SMT area. The importance of reporting is due to the need of fast and accurate decision making by the production managers, derived from the need to maximize the utilization of the SMT machines due to their high costs, and also because, when not optimized, the SMT area quickly becomes the production bottleneck.

Before SIIA, the corrective maintenance business process was that, when a malfunction occurred, the machine operator walks to a PC and reports the malfunction using a specific SAP transaction. Meanwhile, the maintenance technicians continuously monitors the maintenance requests in the SAP ERP. When the maintenance technician spots a new request, he/she first memorizes it or prints a description of the reported problem and them walks to the machine where the problem did occur. After arriving, the technician repairs the machine, goes back into the office, describes the implemented corrective actions, and closes the maintenance order in SAP. This corrective maintenance process described has several problems, namely:

- waste activities are performed by skilled operators, e.g. walk into a PC to report a problem, or continuously staring at monitors to discover maintenance requests;

- production operators and maintenance technicians do not have fast and reliable communication channels, like the case when technicians are all working in the shop-floor, it is hard to find one that takes care for a new high priority maintenance request;

- reporting of initial and closing times of maintenance requests may not be always accurate. This inaccurate recorded data is mainly caused by disjointed events in the occurrence and the data recording of malfunctions because operators and technicians tend to write in small pieces of paper, likely later lost, ultimately causing completely distorted reports of productivity or other TPM (Hartmann, 1992) key figures;

- difficulty to instruct all the operators and technicians on how to properly handle the details of maintenance requests in the respective SAP transactions.

Additionally to the corrective maintenance data, the SMT operators need to contribute with data to the production execution business process. For that, the operators record the data on paper forms (e.g. produced quantities, failure quantity, or failure type). Periodically, the clerks comes to the production shop-floor and collects all the forms to later enter that data into SAP (in their offices), to allow, among others, the confirmation of the planned production. Clerks uses the data to create manually reports based on worksheets. Again, problems with data accuracy on such manual process (e.g. paper forms are lost, defect types may not exist) and performing non-productive activities were common.

The initial high-level requirements for the business processes that SIIA supports are:

- corrective maintenance: when a malfunction occurs, the SMT operator picks a barcode scanner, reads a barcode describing the malfunction, and the data is inputted into SAP without any other human intervention. The maintenance department seeks for the maintenance requests in SAP and goes to solve the problem. After the problem is solved, and using a barcode scanner, the reported malfunction is closed in SAP without any other human intervention;

- production execution: with a barcode scanner, the insertion machine operator reads the barcodes in a paper sheet with quantities and part numbers. This data is booked to SAP without any other human intervention. A label to attach to the container is generated by SAP. The operator places the label in the container.

The main non-functional requirements are:

- due to its wide and intensive use, like the support of shifts 24 hours/day * 7 days/week, the support of ca. 300 users, the SIIA system needs to embody user-friendly interfaces and continuous availability of the reporting mechanisms and data, in order to boost the internal competition for productivity between SMT lines;

- because SIIA is a MES supporting an value-added processes, it needs to have a minimal time to repair when failures occur, and the maximum possible time between consecutive failures;

- for the calculation of the TPM KPIs, the data of SAP and SIIA needs to be validated, in order to guarantee the correctness and alighment between both systems;

- operators feed the SIIA system with data collected via mobile barcode scanners;

- operators book and use all data that concerns their jobs in real-time in the exact place where they work;

- data concerning the SMT area is immediately available to everyone inside the organization, and without any intermediate processing to increase the confidence on calculated KPIs;

- development processes, languages, and technologies such as RUP, UML, or the .net programming languages must be used.

During the early stages of the analysis of requirements, the project team proposed to the steering committee that the requirement to use barcode scanners is changed to accommodate the use of mobile devices (PDA) with barcode reading capabilities and robustness adequate for the industrial environment. These changes in the non-functional requirements have a direct benefit because they allow the use of web-based technologies instead of simple barcode readers, in the business processes where mobility is needed. This change caused improvements of the data correctness and a reduction on the efforts related with creating, updating, and printing barcodes to allow data input into SIIA.

The project team elaborated UML diagrams (e.g. use cases, activity, or state machine), later presented to the steering committee for validation. Figure 5.1 shows the top level use cases for the SIIA system. The depicted use cases represent functionalities existing in the business processes modeled by SIIA. Along with human actors, like SMT Operator, Production Manager, Maintenance Technician, and Main-
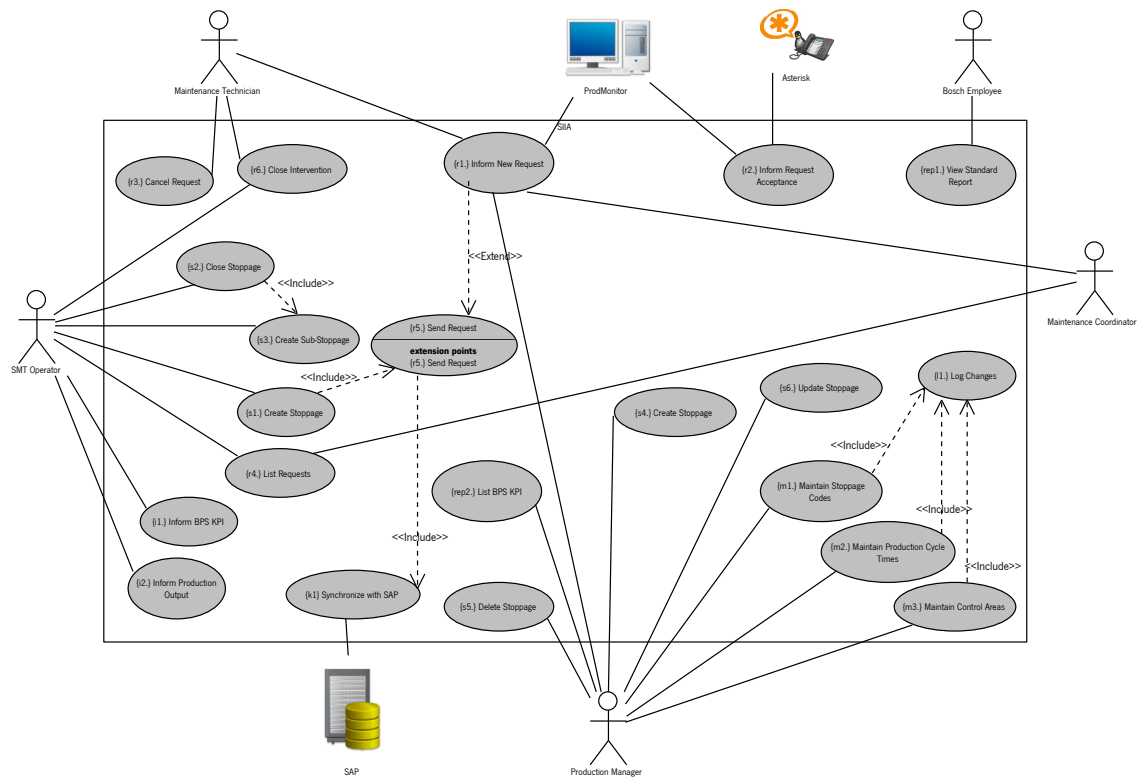
Figure 5.1.: SIIA top level use cases.

tenance Coordinator, external IT systems are also represented to depict the SIIA collaborations with the SAP ERP, with the Asterisk (Spencer, 1999) call-handling systems, and with the ProdMonitor systems. The ProdMonitor systems is an online SMT maintenance monitoring system, developed in parallel with SIIA, to make use of some data collected by SIIA. The ProdMonitor continuously monitors the status of production lines and machines for corrective maintenance purposes (e.g. if lines are stopped or not running at full capacity, or if technicians have accepted maintenance requests). The ProdMonitor allows the production managers to monitor remotely the in-house facilities. Figure 5.2 also shows that any Bosch CMP employee can execute standard reports (e.g. produced quantities, scrap quantities, top problems occurred, efficiency of machines), allowing a complete transparency of the status of production and corrective maintenance business processes.

Figure 5.2 depicts the UML state machine diagram for the business object "Maintenance Request" of the corrective maintenance business process, redesigned with SIIA. In this business process, the maintenance request starts in the client's computing platform, and is then moved into the SIIA server by validating the request and thus becoming an official request. Afterward, the request is informed to technicians and their acceptance is immediately requested online, being both activities done recurring to a phone call, by using the Asterisk system. When the request is in the server, informed, or accepted by the technician, it can still be dropped. The next states do not allow dropping requests, because requests are already closed, and these states are related with reporting activities and not with real world activities.
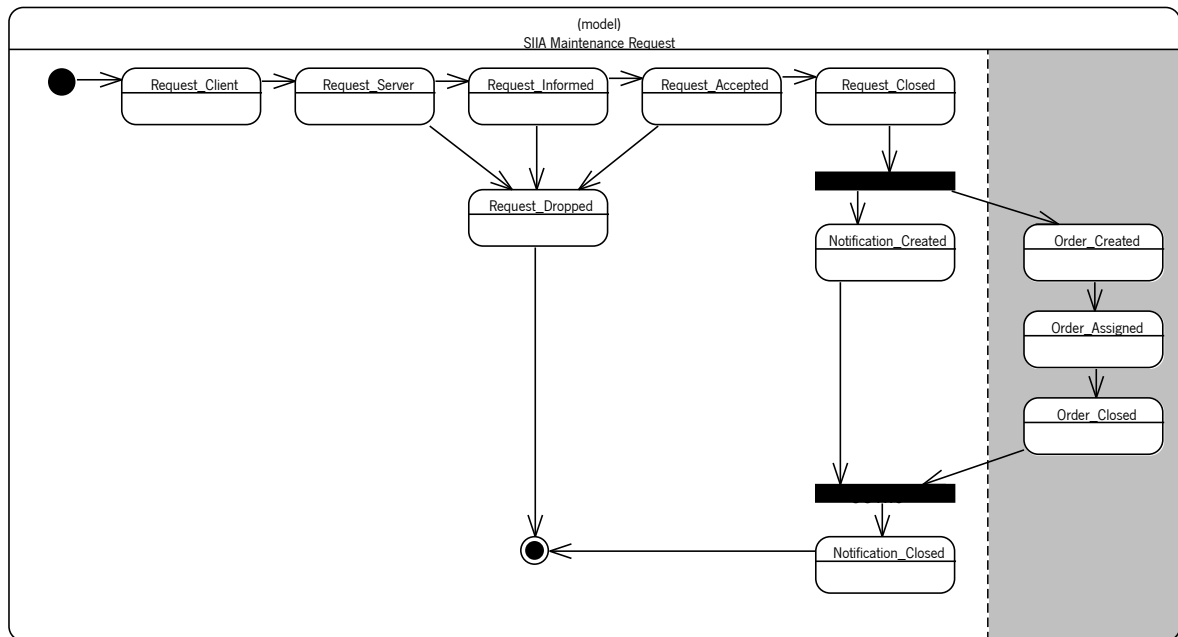
Figure 5.2.: State machine diagram for corrective maintenance requests in SIIA.

After the intervention of the technicians in the line, the request is closed. Then, the integration of the MES (SIIA) with the ERP (SAP) takes place in a forked path. In SIIA, a created notification state exists to exhibit that data input was requested to SAP. In parallel, SIIA incorporates states that detail the processing of the maintenance orders internally in SAP (e.g. Order_Created, Order_Assigned, Order_Closed). When both the notification in SIIA and the maintenance orders in SAP are the fully processed, then SIIA closes the notification. SIIA handles the interface with SAP by triggering the execution, and by supplying the data, to the SAP transactions to process the maintenance orders.

5.2.2.2. Development Project

The business purposes of the SIIA project are the reduction of the paper-based data collection, filled by blue-collar workers in the SMT production, and the centralization, in the SMT production operators, of the data input for malfunctions, produced quantities, errors, and other data to allow an accurate reporting. Bosch CMP, on top of getting a top quality software product, wants to evaluate the SIIA project with the following criteria for success:

- the involvement of all stakeholders during project time: not only the IT staff and one business representative, as it was the common practice in previous projects;

- the execution of formal Quality Assessments (QA) to validate the transitions of phases during project time: QAs are conducted according to the Bosch guidelines for software development processes. Formerly, only the IT team conducted informally the QAs;

- the introduction, in Bosch CMP, of Bosch guidelines for software development activities, including checking the compliance of the software architecture with Bosch standards, validating the

operational concept, or creating internal and external communication plans for the project.

The resulting software system must help Bosch CMP to achieve zero losses in production and thus achieve excellent productivity values. Bosch CMP also wants to visualize data immediately after its creation, to obtain accurate and timely available reports to operators and management, to integrate with SAP top-level processes, to avoid data redundancy in various formats and places, and to be able to accommodate easily future changes and extensions of its business processes and consequently in the SIIA business software.

Expected results from introduction of SIIA are the elimination of manual data inputs, like:

- the produced quantities in SAP (three hours per each eight hour production shift);

- the produced quantities for production reports (eight hours per day) in a local database;

- the produced quantities for TPM cockpit charts (three hours per day) in a spreadsheet.

The customers of SIIA are the SMT and Maintenance departments of Bosch CMP. SIIA must also be capable of later supporting the following, after SMT, production stage: the Final Assembly.

Inside Bosch CMP, SIIA is the first MES project that has a team composed of different profiles of software experts blended with business process experts. All team members report to the same project leaders and steering committee and perform the software development process activities according to standards and best practices, either supplied by RUP or by Bosch guidelines (e.g. QAs for process phase transitions), and supported by a semi-formal and graphical modeling language (UML).

Initially, the SIIA project is setup with two main business purposes: supporting and improving the corrective maintenance processes in the SMT production area, and organizing information and creating a timely and accurate reporting system, available for all stakeholders (e.g. top management, shop-floor workers). Additionally, and due to the need for accurate data, changes are introduced in the SIIA scope to accommodate the requirements related with the production execution business process.

5.2.2.3. Resulting Software System

Figure 5.3 shows the deployment of the components of SIIA and their connections. To cope with the different technologies in the production machines and to support mobility, SIIA has three different user interfaces: simple HTML (for SMT machines based on the Linux operating system), ASPX[1] compatible client for Windows PC platforms, and an ASPX compatible client for mobile clients running Windows.

SIIA totally relies on the .net framework for the BRGWEBAPP server components. The WebSIIA server-side component is the interface with all three types of technologies for user interfaces. All interfacing components inside BRGWEBAPP server recur to the business entities hosted in the SIIA_BusCIs component. The SIIA_iTelefone component is responsible to handle input and output interfaces with the Asterisk BRGVOIP03 server. The SIIA_iBD and Reporting components are responsible to handle the

---

[1]ASPX is a server-side framework designed for dynamic web pages by Microsoft. Any of the .net framework programming languages, like Visual Basic or C#, can be used to create the web pages.
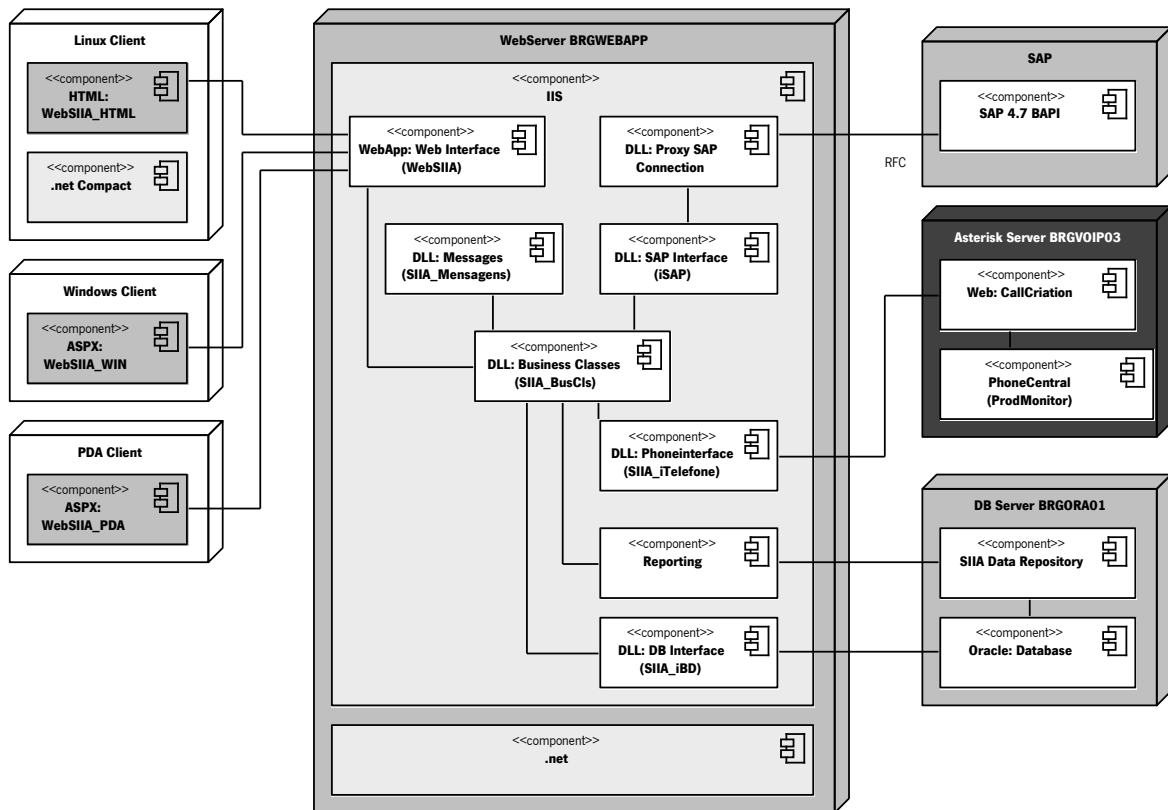
Figure 5.3.: SIIA deployment diagram.

data input and output operations with the Oracle database server BRGORA01. Finally, the iSAP components handles the connection with the SAP server, through a proxy (SAP, 2004) jointly developed by Microsoft and SAP, recurring to Remote Function Calls (RFC). SIIA uses the Application Programming Interface (API) provided by SAP to invoke SAP transactions with data collected from end-user devices (e.g. PDA) realizing this way its MES purposes. The ProdMonitor system is not depicted in the Figure 5.3 because it is completely decoupled from SIIA, and only interfaces with SIIA through the data hosted in the BRGORA01 database server.

5.2.3. Theory Adjustments

The SIIA Action proved that it is possible to develop a MES supporting a value-added business process for a process-oriented organization recurring to standard UML and RUP, and using general purpose programming languages.

Additionally, the Bosch CMP request to evaluate the suitability of RUP to support the IT development team and the adequacy of RUP to MES development during the SIIA Action ended in a positive assessment.

5.2.3.1. Metrics and Lessons Learned

Table 5.1 depicts the maturity levels of the attributes after the SIIA Action, according to the proposals of Table 4.3. Comparing with the respective values of the PWage Action (see Table 4.5), the same results are achieved in:

- the Sound attribute for the Approach element, mainly due to the use of RUP in both Actions;
- the Systematic attribute for the Deployment element, due to the Transition phase of RUP;
- the Learning attribute of the Measurement and Review element, because on both Actions best practices and improvements opportunities are identified and proposed;
- the Improvement attribute for the Assessment and Review element justified by both actions being analyzed and used for planned improvements.

Table 5.1.: Maturity levels of the attributes for the Enabler in the SIIA Action.

Element	Attributes to Score	Maturity Level
Approach	Sound	3
Approach	Integrated	3
Deployment	Implemented	4
Deployment	Systematic	3
Assessment and Review	Measurement	2
Assessment and Review	Learning	4
Assessment and Review	Improvement	4

The remaining attributes scored higher in SIIA than in PWage. The Integrated attribute improved from 2 to 3 mainly because of the activities conducted in the Business Modeling discipline of RUP and due to the steering committee activities to keep project aligned with the policy and strategy of Bosch CMP. The Implemented attribute improved from 2 to 4 (maximum) mainly due to the full extent of the scope of the analysis and due to the implementation of the approach, and, consequently, the resulting software system. Since its first day of use, that all stakeholders view SIIA as a big improvement on the way they perform their activities. The Measurement attribute of the Assessment and Review element is evaluated with 2, because some quantified evidence of this Action is available, which shows that the used approach was effective.

Table 5.2 expresses the Results of SIIA Action. The project team managed to implement all the 21 agreed functional requirements. The steering committee also evaluated further functional requirements to incorporate in a later version of the SIIA system. Out of the 13 non-functional requirements submitted to the team, two of them are not met, namely:

- the MTBF values for the first month of operation;
- QAs, despite using Bosch guidelines, the project team did not execute QAs in their full extent to cover all the transitions between the project phases. Indeed, despite the average score of 90%, the project team only covered with full participation the first and the last, out of the planned

four quality assessments. The remaining two QAs were conducted only with the participation of the IT team, mainly due to different priorities in the tasks of the remaining team members and steering committee members. The values of the effort and duration of the project are also largely exceed, primarily caused by a too optimistic time schedule setup at the beginning and by unclear, conflicting, and incomplete functional requirements stated by business experts.

Table 5.2.: Results of the SIIA Action.

#	Dimension of Analysis	Metric	Value
D1	Requirements	Functional Requirements	21/21 = 100%
D2	Requirements	Non-Functional Requirements	11/13 = 85%
D3	Quality	Quality Assessments	90%
D4	Cost	Effort	498 PD/300 PD = 166%
D5	Time	Duration	420 days/270 days = 156%
P1	Complexity	Physical Lines of Source Code	28518 SLOC
P2	Complexity	Used Software Frameworks	5 USF
P3	Soundness	Change Requests	4 CR
U1	Reliability	MTTR	5 min
U2	Reliability	MTBF	720 min

Bosch CMP triggered four change requests during the first month of use of the SIIA business software system, namely:

- introduce a time stamp expressing the day when the corrective maintenance requests started and the day when they ended. Initially, only the timing was assumed as necessary to calculate the duration of the intervention. This situation was caused by an incomplete analysis of the IT team and by incomplete test cases, and derived on miscalculations when corrective maintenance activities spawned across two, or more, days;

- improve the synchronization of the parallel states of SIIA Notifications and SAP Orders (see Table 5.2). Often, when communication problems occur, the connection between SIIA and SAP is interrupted, causing non-synchronized states between both systems. This situation is caused by not considering, during analysis, the quality of the lower communication layers as an important non-functional requirement that should be explicitly addressed;

- introduce functional requirements that the steering committee considered first to be implemented only in a future SIIA version. Those requirements are related with the editing of master data (e.g. identification of shifts, types of SMT boards, or production cycle times). The root cause for this situation is a priority definition not properly conducted by the steering committee and the availability of the IT team to maintain the related master data directly in the SIIA database;

- introduce improvements in the programming related with cleaning values that remain assigned to some fields in user interface screens. This situation caused improper data input into SIIA and SAP and is caused by programmers and by the lack of full extent test cases.

5. Building Software to Support Business Processes

The complexity of the SIIA system is expressed by 28518 physical lines of source code, and by the use of five software frameworks: .net, .net compact framework for mobile clients, internet information server (IIS), SAP proxy, and Asterisk VoIP server interfacing. The value of five minutes for MTTR is acceptable by SIIA customers, not forgetting that SIIA runs 24 hours/day * 7 days/week. The value of 720 minutes for MTBF is considered not fulfilling the expectations of either the business experts and of the IT team. The former value causes some disturbance on business operations, namely a five minutes (MTTR) delay in the data input, and the latter creates and overhead in the planned capacity for support and maintenance activities. The main cause for the 720 minutes of MTBF is the connection problems between the Microsoft/SAP proxy framework and the SAP ERP.

The lessons learned of the SIIA Action are:

1. standard UML and RUP can be used to develop business critical software for process-oriented organizations. The SIIA Action evaluates positively the development of critical business software recurring to standard UML and RUP, for a process-oriented organization. The quality and business value of the resulting system is unquestionable for business experts. In fact, with the support of SIIA, Bosch CMP is considered a benchmark organization in the SMT area inside Bosch. Several requests to install SIIA in other Bosch plants were made (e.g. Bari (Italy), Penang (Malaysia), Mondeville (France), or Madrid (Spain)). Despite the real values of time and effort are higher than the planed ones, the standard UML proved very useful for the team (regardless of UML being a novelty for the majority of the team). RUP also provided useful guidance for the project. The explanations to miss those targets are more related with the inexperience in the project planning and with unclear requirements from the business experts;

2. UML contributed positively to have concrete discussions on graphical models instead of speculative discussions based on verbal or text models. The SIIA Action showed that visual modeling can be understood by non-experts;

3. hand-carried model transformations in UML require proficiency in the UML language along with business expertise. RUP helps with guidance for the creation of artifacts, but feedback loops with business experts are often required;

4. generic software development processes, like RUP, need to be blended with tailoring tools. Otherwise, the project team can only rely on the expertise and experience of the project leader, or the process expert, for the tailoring of the software development process. In SIIA, the project leader did the RUP tailoring in a manual way, and without guidance for the tailoring activities;

5. the involvement of all stakeholders since the early stages of a software development project is a key factor to achieve the desired quality of the software product and to achieve a smooth start of operations. In addition, during project time, the commitment of all team is higher;

6. for small teams, Bosch guidelines to conduct QAs create an unnecessary overhead in the project team. The time needed to prepare and conduct a QA creates capacity problems and a lack of commitment from all stakeholders;

7. the complete architecture depicting the software landscape of an organization should be considered when developing a MES. Bosch guidelines for software development activities, including the check of the compliance between the SIIA software architecture and the Bosch standards, validating the operational concept, creating internal and external communication plans for the project, are useful to properly create an integrated software product;

8. proprietary software frameworks can stall the pace of a project. At SIIA project time, the SAP proxy developed by SAP and Microsoft had poor documentation and the product was not yet fully mature. The result obtained in MTBF metric expresses both problems. Additionally, the timing to release a newer version of such frameworks is commanded by commercial purposes;

9. the initial planned costs and resources should not be too optimistic, namely when customers do not fully have a significant set of defined requirements. Within project time, it is common that business experts discover new possibilities to implement in the business software, which require time to be discussed and prioritized, and, at the end, cause delays in the project end date. The steering committee members, which are not fully aware of the business process details, still expect, or at least hope, that the project incorporates all changes without impacts on costs, on time, on the amount of implemented requirements, and on the quality of the final product;

10. hard-coded business processes are difficult to maintain and provoke interruptions when corrective or preventive software maintenance activities are to be carried on. Since no life-cycle support for individual software components is available, each time that corrective or preventive maintenance is performed in the SIIA system, no client is able to execute any business process;

11. with the assumptions of the SIIA Action, the reuse of software that supports business processes is difficult. Hard-coded business processes do not consider the evolution of the implemented business process, neither the reuse of components in another business process. If this variability concern is implemented it implies more resources and time for the project that, strictly, may not need them;

12. knowing the complete status of the running software is mandatory to achieve a proper support and maintenance. SIIA incorporates log files and standard messages in all method invocations. The existence of data expressing the status of the software system is crucial to improve MTTR and, after bug corrections, the MTBF metrics.

5.2.3.2. Proposed Adjustments

Based on the lesson learned derived from the SIIA Action, we propose some adjustments for the subsequent Actions. During SIIA project time, and due to the need to improve the maturity of the software development activities in Bosch CMP IT department, we setup an internship to evaluate the gaps of the current practices related with those proposed by CMMi Level 2 (Simoes, 2006). The conclusions helped on improving the project planning and the estimation of resources, by recurring to proven models and by implementing evaluations of software components provided by external suppliers. RUP does

not explicitly implement the evaluation of external suppliers, but CMMi Level 2 advocates performing it.

Due to the difficulties presented when performing UML model transformations, we propose the use and evaluation of some techniques in the following Actions, namely an automated technique to transform use cases diagrams into class diagrams. The technique used is the 4SRS (Fernandes & Machado, 2001).

Due to the stated problems related with software frameworks, the future Actions should evaluate the feasibility of using free and open software frameworks.

The software architecture is created without recurring to any design or architectural patterns. We intend to use such patterns in order to save development time and to incorporate sound solutions for the problems presented to the software architects.

The software implementation of a business process should be detached from compiled source code, to improve both the maintainability and the software reuse. Additionally, the management of the life-cycle of the software components (e.g. the ability to start or to stop a particular component), as well as the ability of a system to accommodate several versions of the same component, should be considered for incorporation into the architecture of future business software systems.

A consensus for the contents of the business processes supported by SIIA was difficult to achieve, because clear definitions are missing and because stakeholders made too many proposals, without a refereeing to decide on priorities, which stalled the project execution. As soon as business processes as defined and agreed, or even reused from benchmarked organizations, the better for software development. For that, we propose to clearly define the business processes just from the beginning of a project (without prejudice of later controlled changes or improvements on the contents of the business processes) or, if possible, to use benchmarked business processes from external organizations.

5.2.4. Social Environment

In the SIIA Action, the author of this thesis acted, following RUP roles, as project leader, software architect, and programmer. Additionally, and following Bosch guidelines, the author of this thesis also acted as a member of the steering committee, responsible for the change management, and is the responsible for the support and operation after go-live.

The SIIA team had no previous experience in working together in a project, either among the IT staff, either with a vast extent of business experts.

None of the team members of SIIA was exclusively dedicated to the project. For that, business experts needed to perform normal operational tasks, and the IT staff needed to provide support and mainte-nance to all software systems of Bosch CMP, together with the participation in other projects. Despite not being the best staffing schema, it is a constraint coming from Bosch CMP internal organization and management evaluated as not critical.

Since SIIA is the first MES covering a business critical area in Bosch CMP, it received much attention from management and blue collar workers during its development. Despite this situation is a positive factor contributing to keep SIIA on top of priorities of daily activities, it also contributes to stall the SIIA development due to an excess of opinions to shape the business processes and contributes to increase the effort to converge into an agreed business process.

SIIA is a breakthrough for all Bosch organization for MES development, due to its ability to interface shop-floor data into SAP. SIIA uses the API from Microsoft and SAP to connect to the business layer of SAP ERP. At the time, Bosch had interfaces based on the user interface layer. Due to this fact, SIIA received resistance from Bosch employees from central departments.

Due to the reliability, availability, and accuracy, the SIIA data is the cornerstone to evaluate the individual performance of blue-collar operators on annual appraisals later deriving on monetary prizes. Blue-collar operators understand and accept the evaluations based on SIIA reports. Before SIIA, blue-collar operators disputed the data used for annual appraisals.

The SIIA system received very positive mentions on formal quality audits and received several requests for implementation in other Bosch locations on benchmark visits. Bosch CMP uses the SIIA system without interruptions since 2006.

5.3. The SOL Action

Bosch CMP started the SOL project triggered by an urgent need to substitute a legacy system supporting the business processes related with internal logistics. The main reasons to setup the SOL project are:

1. maintenance problems, namely by the MTTR and MTBF values below expectations, caused primarily by an outsourced support and maintenance and by the legacy radio frequency technology used to connect mobile clients to servers;

2. interfacing problems with the SAP ERP, caused by a technology based on SAP user interfaces (through the use of macros to simulate human data input) and, for that, hard to maintain when changes in the SAP user interfaces occur;

3. new and updated business processes needed to be incorporated into the mobile solution.

SOL has a direct impact on more than 400 workers and has 115 users from logistics and 25 from production functional areas.

5.3.1. Purpose of the Action

The SOL Action picks the proposals derived from the SIIA Action (stated in Section 5.2.3.2). With the SOL Action, we want to create a new Action environment, namely by:

- introducing improvements in the project planning and estimations, by re-evaluating the RUP

template used (Large Projects), presumably recurring to more than one iteration inside each RUP phase, and by using the Cocomo II as the estimation model (Boehm et al., 2000);

- using the 4SRS automated technique to derive a candidate logical software architecture out of the use case model;

- introducing, explicitly, the use of design or architectural software patterns;

- introducing free software frameworks, like the Spring (see Section 3.2.2) and the Hibernate (Bauer & King, 2006), to, respectively, support the IoC design pattern to better decouple software components, and to support object-relational mapping between an object-oriented programming language (Java) and a relational database;

- improving the flexibility of the reporting mechanisms used in the SIIA project (despite business experts did not claim such improvement). Due to the estimated high quantity of reports in the SOL system, the Eclipse business intelligence and reporting tools (BIRT) (BIRT, 2007) framework is used;

- creating a software environment where business processes should be detached, as much as possible, from compiled source code. The proposed software environment is Java-based and integrates the Spring software frameworks;

- reinforcing timely and clear definitions of the business processes;

- incorporating benchmarked business processes.

Despite not being considered in the proposals derived from the SIIA Action, the IT team decided to introduce a new technology for user interfaces in order to increase user-friendliness. For that, the rich internet applications (RIA) (Fraternali et al., 2010) approach (featuring fat clients in web browsers instead of classical thin clients) is used. The technology supporting RIA is the Flex[2] Cairngorm (Adobe, 2008b) framework. Flex Cairngorm implements the MVC architectural pattern to separate the representation of the information from the interaction with the users.

The SOL Action continues with some practices coming from the SIIA Action due to their positive evaluation, like by:

- using UML and RUP;

- having a steering committee;

- involving the business stakeholders since the beginning of the project;

- validating the software architecture within the complete software landscape of the Bosch organization;

- incorporating the monitoring of the running system to help on maintenance activities (in this project recurring to the Nagios (Nagios, 2013) IT infrastructure and business processes monitoring framework).

---

[2]Flex was initially conceived by Adobe. Since 2011, the Apache Software Foundation is responsible for the development of the framework.

5.3.2. Practical Action

The SOL Action uses the RUP, this time, tailored to the "Classic RUP (for Small Teams)" template (IBM Rational, 2006). The change from the previously used "Classic RUP for (Large Projects)" template is caused by the vast number of deliverables, proposed by the later template, and by the associated tailoring effort related with the low (eight) number of IT elements in the team. In the SOL Action, RUP is not manually tailored, as it was in the SIIA Action, but recurring to the RMC tool (Kroll, 2005).

In the SOL Action, the business processes are modeled with UML use case, activity, and state machine diagrams. The software implementation of the business processes is done recurring to Java and uses Spring as the main software framework. This way, business processes are coded in both Java and XML (due to their Spring nature), and are deployed into an Apache Tomcat runtime[3].

The SOL project has a core team composed of eight software engineers with additional three IT infrastructure experts, and eleven business experts. Additionally, Bosch employees from central departments are members of the steering committee, in order to align Bosch CMP and global Bosch software architectures and concepts for MES development. The Eclipse IDE supports the development environment for the SOL project.

5.3.2.1. Implemented Business Processes

The SOL software system is a business critical application. From Bosch CMP point of view, the SOL-mapped processes are time critical and the SOL data is a key asset for the organization. The implemented business processes should also synchronize perfectly, and avoid overlapping, with the higher-level business processes implemented in the SAP ERP. The business processed mapped in SOL are subject to benchmark activities with other Bosch plants and also checked for compatibility with higher-level business processes definitions existing in Bosch similar plants. The SOL system, including the business software and the mobile clients, is the replacement of a legacy system based on Radio Frequency communication technology. On top of the migration of the wireless communication technology, all the supported business processes are migrated, possibly with adjustments triggered by new business needs, and new business processes are considered. All SOL-mapped business processes are part of the same top-level business process, Fulfillment.

The high-level functional requirements for SOL are related with the support to mobile operations for goods receipt and expedition, material requests, kanbans (Sugimori et al., 1977) with suppliers, confirmation of cargo lists, and with the support to mobility in the warehouse operations mapped in the SAP ERP. These functionalities occur logically in the business processes of internal logistics, and physically in the material warehouses, the near-production buffer warehouses, the final product warehouses, and in the production shop-floor.

---

[3]In fact, it was deployed into an IBM WebSphere application server community edition (a more robust implementation of the Apache Geronimo, which includes Apache Tomcat). SOL only uses the functionalities of Tomcat.

5. Building Software to Support Business Processes

The main non-functional requirements are:

1. improve the values of the MTBF (to higher than one week) and the MTTR (to less than ten minutes);

2. support the end-user mobility in warehouses and internal logistics;

3. Bosch CMP IT department must be the responsible entity for the support and maintenance of SOL (the previous system had this responsibility outsourced);

4. the architecture of SOL software system must be able to accommodate future functionalities;

5. SOL should have a layered design and should incorporate the Spring, Hibernate, BIRT, and Flex frameworks;

6. both, the PC-based client-side and the server-side components must be developed in Java;

7. the client-side mobile clients must be developed in .net (compact framework);

8. SOL must be compatible with the SAP ERP and must integrates with the SAP ERP mapped business processes, including those developed internally by Bosch;

9. SOL must be compatible with the hardware and with the operating systems (Windows Mobile or Windows CE) of the mobile terminals, and with the wi-fi infrastructure;

10. usability is very critical for the acceptance and proper use of the SOL system;

11. performance, namely to have any requested coming from any client serviced and displayed in under three seconds, and to have a mobile client boot time inferior to 30 seconds;

12. SOL must include monitoring.

Figure 5.4 presents the top level use cases for the SOL system. Worth noticing is the association of the external SAP ERP system to almost all use cases. This fact derives from the need to integrate the mobile, and low-level, business processes into the high-level business processes mapped in the SAP ERP (mandatory for Bosch CMP, as required by Bosch).

The external PSP system collaborates with SAP ERP and the Dispatch Operator in the creation of cargo lists to permit only the shipment of valid products. The Near-production Warehouse Operator, the Dispatch Operator, and the Materials Warehouse Operator perform normal warehousing and dispatching functions. The Materials Warehouse Operator performs his tasks inside an open elevator reaching 20 meters height, justifying maximized efforts from the project team to implement properly reliability and user-friend use cases. The Logistics Manager visualizes the KPIs and configures the master data (e.g. heuristics to define the picking routes inside the warehouses).

In SOL, the project team tailored RUP by adding Storyboards for use case realizations, as used in agile methods, in order to better explore and describe how a user interacts with the software system. The ultimate goal of such Storyboards is to validate if the business process flow is accurate and complete.
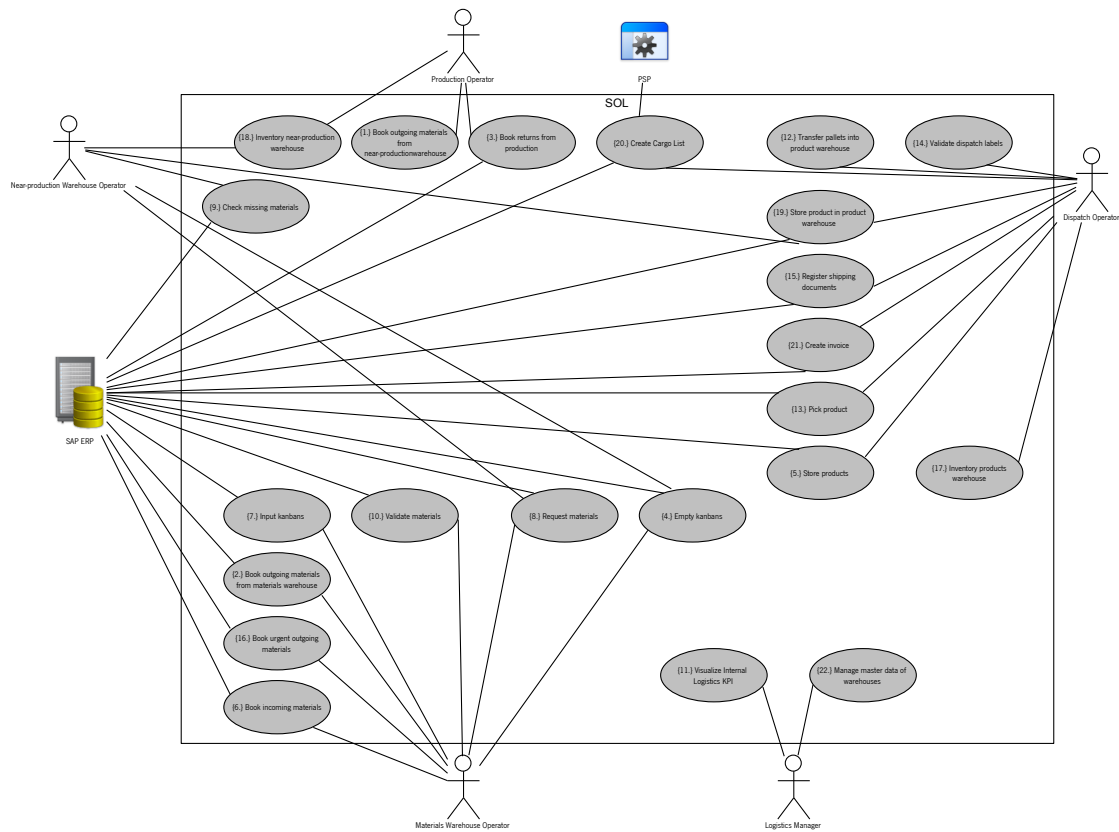
Figure 5.4.: SOL top level use cases.

5.3.2.2. Development Project

The development process used in the SOL project is the RUP. RUP is evaluated positively against agile methods as the preferred software development process for the SOL Action. We base the evaluation on the following facts:

- RUP, as a process model, is able to cope with some useful agile practices, namely by having an on-site customer, work in a 40-hour per week schema, having small releases, or by using pair programming (Pollice, 2002);

- RUP has a better tool support than agile methods;

- the RUP supporting tool defines and explains very well the RUP process. The RUP tool includes templates with proposals for the content of the artifacts;

- The SOL project involves a significant interaction with business representatives, and RUP provides, off-the-shelf, a usable framework for that purpose. Other options to model the software development process are less prescriptive and have less defined artifacts (e.g. XP, Scrum, or other agile method), requiring the project team to create the needed templates, thus stealing time for the development of the software system;

The tailoring of RUP is done recurring to the RMC tool and derived from the template "RUP (for Small Teams)". RMC allows an easy tailoring of RUP, and provides templates that meet both the Bosch CMP

requirements and the characteristics of the project team.

Despite the "RUP (for Small Teams)" template does not incorporate the discipline of Business Modeling, it was considered by the project team fundamental to add it to the project tasks due to the importance, for Bosch CMP, of a very good understanding and implementation of the business requirements. For that, the activities and, thus, the artifacts of the Business Modeling discipline, are considered in the SOL project setup.

For the SOL project, the project team tailored the RUP life-cycle to accommodate two iterations in the Construction phase, in order to better deal with eventual technological problems by making available, as soon as possible, a software system usable in real conditions. Additionally, we planned two iterations in the Transition phase, in order to implement the solution stepwise into the different physical locations.

The initial planning values, resulting from the Cocomo II estimation model, are 22.4 Person/Month for the total effort, and 11.8 months for the duration of the project. According the Cocomo II estimation model, non-nominal cost drivers to express the project characteristics are setup in the following way:

- personnel: high capability of analysts, low application experience, low programmers capability;

- project: mature development tools and moderately integrated, fully collocated team;

- product: very high complexity.

Additionally, the following Cocomo II scale factors have the following setup:

- precedence: largely unprecedented, due to the business processes mapped, the innovator character of the frameworks and development languages used;

- development flexibility: rigorous, related with the very low flexibility in the requirements;

- architecture/risk Resolution: Generally (75%), mainly because architecture is defined and expected to be adequate, based on the non-functional requirements. This judgment is also taking into consideration the use of software patterns in the architecture;

- team Cohesion: Largely Cooperative, based on previous interactions between all the members of the team;

- process maturity: CMM level 1 (upper half), based on the evaluation performed by Simões (2006).

The SOL project has, since its beginning, one IT headcount fully dedicated to define, together with the business experts, all the requested business processes.

5.3.2.3. Resulting Software System

Figure 5.5 shows the deployment of the SOL software components and their connections. The web clients have the Flash player sandbox installed to run the SOL web components. Mobile clients run on top of the .net compact framework. The BRGPROJEE02 server runs the IBM WebSphere application server community edition (WASCE) instance. WASCE hosts specific service components to deal with

Flex services and with the web services (in this case recurring to the Apache CXF). The interfacing service components relate with the components that contain the business logic: near-production warehouse, warehouse, and heuristics components. The warehouse and the near-production warehouse components need to recur to a print service. In the BRGPRJEE02 server, all components sit on top of the Spring framework, supported by the Java Runtime Environment (JRE).

The Nagios server hosts the components for monitoring the IT infrastructure (e.g. if the system is running, if disk space is available, the current memory footprint), together with the components responsible to check if the business processes of SOL are healthy.
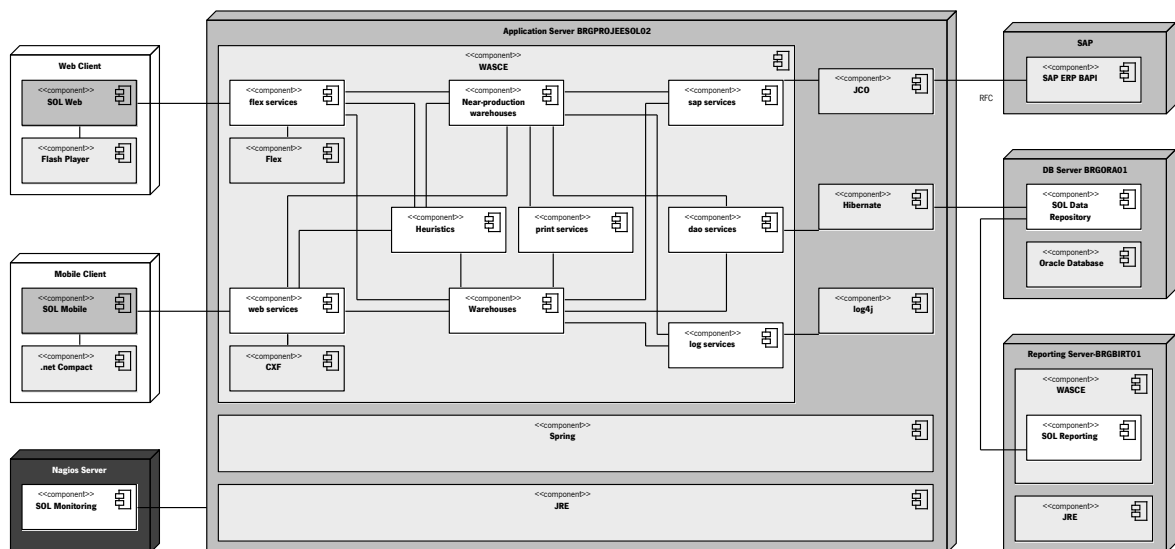
Figure 5.5.: SOL deployment diagram.

The interface of the server BRGPROJEE02 with SAP ERP is done via the SAP services component, recurring to the SAP Java JCO connector. The data access object (DAO) component, recurring to the Hibernate ORM framework, deals with data operations on the SOL data repository hosted in the BRGORA01 Oracle database server. All reporting functionalities are provided by the Eclipse business intelligence and reporting tools (BIRT) in the BRGBIRT01 server, hosting the SOL reporting component embedded in a WASCE instance, and linked to the SOL data repository. The external PSP software system (represented in Figure 5.4) interacts with SOL through the web services component.

5.3.3. Theory Adjustments

The SOL Action proved that free software frameworks are key assets for the development of business software. It also demonstrated that a proper tailoring of RUP, by including some agile practices, is beneficial for a small software development team. The use of software patterns, at either design or architectural levels, creates a more robust software solution able to support business critical processes.

5.3.3.1. Metrics and Lessons Learned

Table 5.3 depicts the maturity levels achieved in the SOL Action, according to the proposals of Table 4.3. The comparison with the SIIA values (see Table 5.1), shows an improvement in the Sound attribute, due to the higher and adequate tailoring effort carried on RUP, and in the Measurement attribute, due to more evidences of a quantifiable measurement of the effectiveness of the approach used in the SOL and its deployment into the project planning and execution.

Table 5.3.: Maturity levels of the attributes of the Enabler in the SOL Action.

Element	Attributes to Score	Maturity Level
Approach	Sound	4
Approach	Integrated	3
Deployment	Implemented	4
Deployment	Systematic	3
Assessment and Review	Measurement	3
Assessment and Review	Learning	4
Assessment and Review	Improvement	4

The remaining attributes stayed in the same level as those of the SIIA Action, either because they are already at the top level (Implemented, Learning, and Improvement) and the project team managed to keep them at the top level, either because improvements could not be implemented, like in the Integrated attribute (due to the improved, but still manual, mapping of the business processes into software), and in the Systematic attribute (because of the considerable effort from project leaders to keep the planned approach for the project to run according to the plan).

Figure 5.4 presents the Results of the SOL Action. The number of functionalities requested was 20. During project time, two additional functional requirements are presented. The project team managed to implement all 22 functional requirements, resulting on an implementation rate of 110%. Also, 100% of the 12 presented non-functional requirements are implemented or agreed (e.g. the responsibility for support and maintenance after go-live).

The average result of the quality assessments is 80%, a lower value compared with SIIA. This result is explained by the use of the "RUP (for Small Teams)" combined with some agile practices, which resulted in less documents fulfilled, and, at the end, less evidences at the quality assessments milestones.

If the two additional functionalities are considered, the Effort result of 105% is positive compared with the foreseen increase of 10% (two more functionalities additional to the initially planned 20). The poor result in the Duration is also explained by the two additional requirements, but mainly by the unavailability either of business experts or physical conditions (e.g. proper quality of service of wi-fi communications, warehouse changes) making inadvisable to terminate the project without all roll-outs in the Transition phase.

Table 5.4.: Results of the SOL Action.

#	Dimension of Analysis	Metric	Value
D1	Requirements	Functional Requirements	22/20 = 110%
D2	Requirements	Non-Functional Requirements	12/12=100%
D3	Quality	Quality Assessments	80%
D4	Cost	Effort	493 PD/470 PD = 105%
D5	Time	Duration	660 days/354 days = 186%
P1	Complexity	Physical Lines of Source Code	16265 SLOC
P2	Complexity	Used Software Frameworks	12 USF
P3	Soundness	Change Requests	0 CR
U1	Reliability	MTTR	3 min
U2	Reliability	MTBF	>43200 min

Included in the 16265 physical lines of source code are 1883 lines of XML configuration files for the software frameworks used (mainly due to the use of the Spring and the Hibernate frameworks). The SLOC result cannot be compared with SIIA because different programming language are used and because of the increased use of software frameworks. The 12 software frameworks used are Flex Cairngorm, Flash Player, Spring, BIRT, Hibernate, JCO, WASCE, .net compact, CXF[4], log4j, Nagios, and JRE[5]. Additionally, at project time, and not accounted in Table 5.4 because it was only used for development, the project team used the JUnit framework (Gamma & Beck, 2006) to write repeatable unitary tests. The result of USF demonstrates that SOL is a complex software system, due to the amount of technologies that the team has to master.

Culminating the efforts at project time for getting suitable business process models, no new change request is presented to the project team during the first month of operation of the SOL system.

The MTBF result is according to the non-functional requirement of being higher than one month, because no failure was detected during the first month of use. The MTTR result is calculated based on tests and not during real operation because no failure was detected.

The lessons learned from the SOL Action are:

1. the template "RUP (for Small Teams)" appended with the "Business Modeling" discipline and some agile practices is better suited for Bosch CMP than "RUP (for Large Projects)", mainly due to the size of the software development team. Nevertheless, deliverables that must presented at quality assessments must be always fulfilled;

2. models of business processes, timely available and incorporating clear definitions, are crucial for the development of business software. To help obtaining clear definitions is advisable to benchmark already existing, and proven, business process models. The SOL project team achieved this purpose by comparing Bosch CMP business processes with other from similar

---

[4]XFire at project time.
[5]Hosting the Java Virtual Machine (JVM) and additional Java class libraries

plants, and by checking the compliance with higher-level process definitions;

3. the mapping of business processes into code is still an issue. Each time the code is re-factored, the UML artifacts need to be carefully checked, or vice-versa;

4. the use of software frameworks increases the quality of the resulting software product. This result can be seen by the lower amount of developed code (and thus having less opportunities to error) and by the results of the Reliability metric. In the case of using free software frameworks, the development cost is also reduced (assuming that the team is proficient in those chosen frameworks);

5. free software frameworks are often poorly documented. When the team is not technologically able to produce quality solutions with that type of documentation, or when training is not feasible, then it is advisable not to use such frameworks;

6. the use of design and architectural patterns (e.g. IoC in Spring, or MVC in Flex) helped reach fast and quality solutions for the presented problems, and, for that, their use is recommended;

7. the decoupling of the implementation of the reporting components from the business process components allows the parallelization of development efforts. The use of a framework such as BIRT allows a flexible solution for the never-ending reporting requests coming from business experts;

8. the use of the 4SRS technique allowed the creation of the first logical software architecture out of the use case model, in just one afternoon, only by two headcounts. For that, the use of such technique in model transformation is an value-added in time and in the quality of the resulting artifact;

9. the use of the RIA approach allowed appealing and usable user interfaces.

5.3.3.2. Proposed Adjustments

Related with lessons learned, the main proposed adjustment is to better align business processes models (in the present case expressed in UML) with their software implementation. For that, executable business process languages should be evaluated in the following Actions.

Additionally, WASCE (by only using the functionalities of Tomcat) and Spring are not yet a complete solution to allow the reuse of parts of business processes already implemented in software, neither to manage the life-cycle of software components. For that, we suggest a modular software architecture (e.g. ESB) based on a dynamic component system (e.g. OSGi).

5.3.4. Social Environment

In the SOL Action, the author of this thesis acted as software architect, process engineer, test manager, technical reviewer, and designer, following the RUP roles.

SOL is requested, primarily, as a substitution of a previous software system that supported the same business processes but had a poor performance on MTTR and MTBF. Since the previous system was externally developed and customized for Bosch CMP, the support and maintenance activities are also assigned to the external company who developed the software. Additionally, mobile communications were based on Radio Frequency terminals, which demonstrated unfit for the business requirements, like when data losses occurred. For that, SOL is an opportunity for the Bosch CMP IT department to, again, demonstrates its ability to create a professional solution and to guarantee the adequate support for Bosch CMP business processes.

As in SIIA, SOL faced roadblocks from Bosch central departments. SOL is a competitor for a solution based on SAP mobile clients, at that time, only able to interface with SAP ERP via transactions. Despite the roadblocks, Bosch CMP steering committee evaluated SOL as providing more business value than the competitor solution. For instance, SOL is connected to another MES in production (PSP, also developed by the Bosch CMP IT department) that together validate the deliveries to customer plants and guarantee that all products for dispatch are formally correct and inside the proper pallets. SAP ERP cannot assure this kind of validation because traceability to each product serial number is needed, as well as real-time data processing, which are not part of SAP ERP abilities.

The SOL project team received from Bosch CMP a "Prémio Equipa+" award. This award recognizes the quality and innovation of the development efforts of a project team.

5.4. Conclusions

Table 5.5 shows the evaluation of the status of SIIA and SOL Actions related with goals of this thesis (see Section 1.4). At the end of the SOL Action, the criteria to stop the Action Research studies are not yet achieved. For that, the Action Research studies need to proceed and, in Chapter 6, more Actions are presented. An improvement is perceived in the definition of a methodology to automate the transition from the models of business processes into running software (MG), namely by using model transformation techniques and software patterns, but not sufficient to stop the Action Research studies.

In SOL, the use of software frameworks, namely Spring, helped on improve the SG3 result, but manual transformations of models are still needed and software reuse at runtime is not yet possible. The Sol Action also demonstrated that the SG5 goal fulfillment improved because the project team reinforced the need to have proper and timely available models of business processes, since the beginning of the project.

Bosch upgrades the content of high-level business processes, mapped in the SAP ERP system, with a frequency higher than once per year. When such upgrades occur, Bosch CMP has three options:

1. to completely stop the production for up to four days (which was the normal situation previous

Table 5.5.: Evaluation of the stopping conditions for Action Research related with SIIA and SOL.

Goal Id	Goal Description	SIIA	SOL
MG	To propose a methodology to generate immediately executable information and software systems for process-oriented organizations, starting with models of business requirements	Not Achieved	Partially
SG1	The resulting software products must have quality	OK	OK
SG2	The proposed methodology should be able to be executed in a short period of time	Not Achieved	Not Achieved
SG3	To specify and to implement a software framework to support software developed with the methodology	Not Achieved	Partially
SG4	The resulting methodologies and software products should be able to be used by business experts without intervention of software experts	Not Achieved	Not Achieved
SG5	The proposed methodology should be holistic, ranging from business processes descriptions to running software	Not Achieved	Partially
SG6	To promote the composition of software solutions based on free software and originated by different vendors	Not Achieved	OK

to SOL);

2. to use a solution based on a full copy of the SAP ERP system (with a downtime of ca. 20 hours and an estimated cost of 600 000⏻);

3. to develop an in-house software solution.

The Bosch CMP decided for the third option, and for that, SOL was reused as the basis for the new software solution. SOL allowed the creation of a sister system called LUA. LUA is a version of SOL without the online interface to SAP ERP, but keeping the same user interfaces that the end users are accustomed. LUA keeps all the SAP ERP associated data internally and later (i.e. when the intervention period on SAP ERP finishes), LUA books the internally kept data, in an sequenced way, to SAP ERP. This way, LUA provides an excellent business value because it prevents the production to be completely shut down for up to four complete days.

LUA also guarantees zero downtime on the production and logistics activities, when the changeover occurs from SOL to LUA and vice-versa.

LUA has a cost near to zero, because only Bosch CMP internal team is involved in their use and setup. Since LUA uses the same user interfaces and behavior of SOL, LUA does not need training actions for operators, which would cost significant time and money for Bosch CMP. Additionally, at development time, the vast majority of the code was reused from SOL.

For the former reasons, SAP ERP, despite being an excellent software product, cannot cover MES areas of intervention. Additional reasons to have MES working together with ERPs are the inability of the latter to cope with real-time events normally occurring in a production shop-floor, and the inability to map the more detailed levels of business processes. For that, classical ERPs, without any extension, cannot be used alone in contexts of process-oriented organizations where low-level business activities occur (e.g.

manufacturing, internal logistics). In such cases, the support of a MES is mandatory.

6. Running Business Process Reference Models in Software Frameworks

> "There is a central quality which is the root
> criterion of life and spirit in a man, a town, a
> building, or a wilderness. This quality is objective
> and precise, but it cannot be named."
>
> Christopher Alexander

This chapter presents the Centauro 1.0 and the Centauro 2.0 Actions. Here, we develop MES business software recurring to ESBs. Centauro 2.0 demonstrates the use of BPEL and a new methodology to transform business process models into running business software systems. Centauro 2.0 also includes proposals for new concepts to allow the transformation of business process models into running software systems.

6.1. Introduction

The business process landscape of a process-oriented organization is the most important input for a proper definition of the functional, and probably the non-functional, requirements of business software.

For an organization, it should be useless to have perfect business process models, but not lived on a daily basis. The business process models should depict the way an organization operates in practice. The models of business processes should not be exclusively used for presentations during non-value-added activities, like management meetings or quality audits.

For the business software development, it is necessary to consider the business processes since the beginning of the software development efforts. Additionally, it is pointless to use effort in a project for collecting processes, sub-processes, and tasks definitions when they are already available in reference models.

When collecting requirements expressed by different business stakeholders inside the same organization, it may occur that conflicting requirements are stated, perhaps causing delays in the project. It can also happen that the low-level requirements of a sub-process are incompatible with the higher abstraction levels of the same process. These conflicting situations are usually seen within organizations with low maturity levels on BPM.

Process-oriented organizations do not need to constantly invent new business processes, neither their content. BP-RMs are available and include the definitions of business processes embodying best-practices and accumulated knowledge. Nevertheless, and despite the good practices contained in the reference models, an organization has always the right to refuse to use some business process, to tailor it, or to add new business processes to its own process landscape.

The software environment supporting the execution of business process models should also contain characteristics to allow a proper BPM, namely to be fast to deal with changes in activities inside the business processes, to allow the existence of different versions of the same business process inside one organization, or to manage the complete life-cycle (e.g. deploy, start, stop, remove) of the business processes. The language used to model the business processes is of major importance. Desirably, such language should embody two main characteristics:

1. be easily understandable by all stakeholders within an organization, including the business experts. This characteristic may induce graphical representations of business processes;

2. be executable. If executable models of business processes are not used, then each time a project team performs a model transformation, expressiveness, time, and capacity may be lost in the translation between the non-executable model and the executable model.

The focus of a process-oriented organization should be on the proper design of its business process, and not on the software implementation or on customization activities, that, from the point of view of the process-oriented organization, are not value-added efforts.

The results of the Actions presented in this Chapter intend to demonstrate that improvements on the degree of achievement of the Action Research goals occurred in relation with those of Chapter 5, namely on the software environment to execute business processes and in the definition of a methodology that promotes and accommodates the use of BP-RMs.

Included in the Centauro 1.0 Action, it is also presented a study to compare several languages to model business processes and to choose an adequate model for the business processes based on the intrinsic characteristics of the language and on the characteristics of a given project team.

No methodology is known to pick business processes contained in BP-RMs, to tailor the business processes, and to create business software. For that, we propose (described in detail in Chapter 7) and use (in the Centauro 2.0 Action) a new methodology.

The main purpose of this Chapter is to demonstrate that it is possible to design the business process landscape of an organization based on best-practice BP-RMs and, then, to reach its implementation in software, using an integrated and holistic methodology.

6.2. The Centauro 1.0 Action

The necessity of Bosch CMP for the Centauro 1.0 system comes from the difficulty of having an inventory with an high degree of accuracy, and from the need to facilitate the movements of incoming and outgoing SMT semi-finished products into a warehouse.

Centauro 1.0 has 20 users and affects ca. 250 workers.

6.2.1. Purpose of the Action

The Centauro 1.0 Action is setup with the following goals:

- to evaluate and define a standard for Bosch CMP for an executable business process language to be used in future MES projects;

- to propose a reusable software framework to allow the execution of business processes. Business processes should be detached from compiled source code. It should also be possible to manage the life-cycle of the software components, while keeping the Centauro 1.0 software system running. For that, the ServiceMix ESB (on their OSGi-related capabilities) and the Spring software frameworks will be evaluated as solutions to support the reuse of software components and to manage the life-cycle of the software components;

- to evaluate a software development process (titled F3 and proposed by Bosch), by checking its adequacy to the development of MES software for Bosch CMP.

The decision to drop RUP as the process model for software development for Bosch CMP is due to the necessity to adhere to the F3 development process, defined by Bosch central IT department in its quality manuals, as the standard for ISO 9001 compliance.

From the previous SOL Action, Centauro 1.0 keeps the intention to use design and architectural patterns, to use UML, to have parallel development of reporting and business processes software components, to use free software frameworks, to use the 4SRS technique, and to use the RIA approach.

6.2.2. Practical Action

The Centauro 1.0 Action uses the F3 development process. F3 is a proprietary development process based on the Waterfall model. F3 was introduced at Bosch CMP in order to guarantee compatibility with standard software development processes for the central Bosch IT division.

In the Centauro 1.0 Action, the business processes are modeled with UML use case, activity, and state machine diagrams. The software implementation of the business processes is done recurring to Java and XML (triggered by their Spring nature) and deployed into a ServiceMix 4 ESB runtime.

The Centauro 1.0 project has a team of six software engineers and four business experts. Additionally to the technical project team, the F3 development process introduces new stakeholders only to obtain

clearances, from local and central departments, to initiate the software development technical efforts. The development of the Centauro 1.0 software system is done recurring to the Eclipse IDE.

6.2.2.1. Implemented Business Processes

The Centauro 1.0 software system maps business processes related with internal logistics movements and with the inventory management, in the SMT production area. These two business processes are part of the top-level business processes, Fulfillment and Finance, respectively.

Each business process implemented in Centauro 1.0 must cope with the business processes implemented in the SAP ERP and must support the mobility of the respective blue-collar users.

The high-level functional requirements for the Centauro 1.0 business processes are related with the booking of (semi-)finished products, at SMT shop-floor, and with the management of a warehouse interfacing the initial SMT production phase with the final assembly production phase, recurring to lean production concepts (Womack et al., 1990) like supermarket, milkrun, and kanban. Additionally, Centauro 1.0 adds the ability to handle a permanent inventory at the supported warehouse.
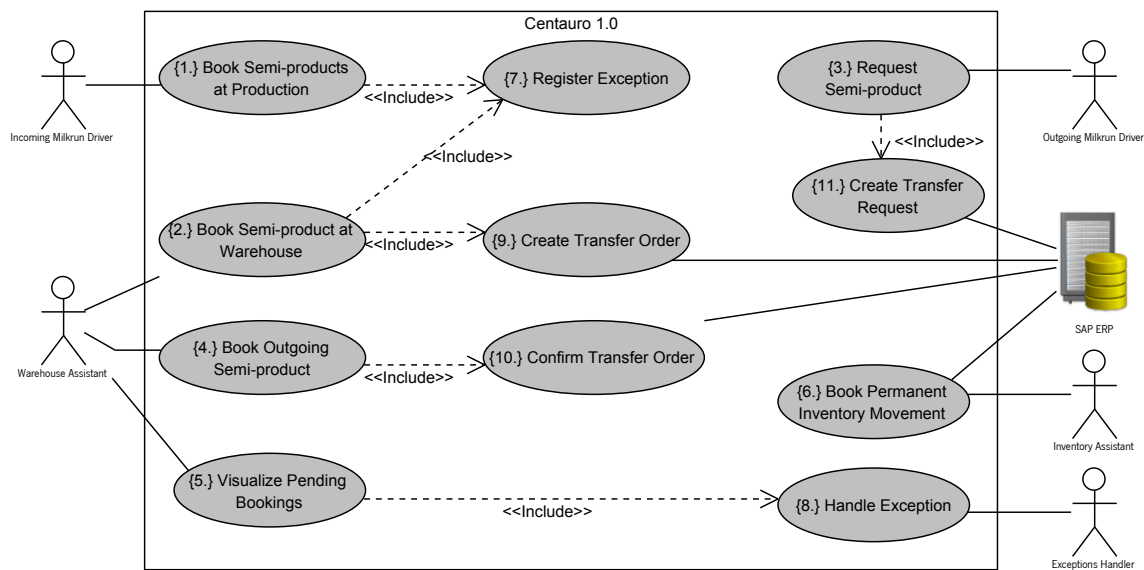
Figure 6.1.: Centauro 1.0 top level use cases.

Figure 6.1 presents the top level functional requirements for Centauro 1.0 software system. The incoming milkrun driver books the semi-finished products, at the production shop-floor. After, these same products arrive at the supermarket warehouse, and the warehouse assistant books again the same products. Then, a transfer order is created in SAP ERP. If any deviation between the two bookings occurs, then Centauro 1.0 registers an exception, later handled by the exceptions handler. For the outgoing semi-products leaving the supermarket warehouse, the warehouse assistant books them and confirms a transfer order existing in SAP ERP. The warehouse assistant can also visualize the pending bookings to be satisfied.

Since Centauro 1.0 maps a supermarket warehouse, the outgoing movements are triggered by the consumption. Each time the outgoing milkrun driver registers consumption at the final assembly production lines during his milkrun route, he is performing a task equivalent to a request of a semi-finished product to replenish the consumed semi-products at the final assembly shop-floor. Additionally, a transfer request is created in SAP ERP.

For inventory purposes, the inventory assistant books the permanent inventory movements in collaboration with SAP ERP.

The main non-functional requirements are:

1. all user interface software components for PC-based use cases must be developed recurring to the RIA approach (by using the Flex framework);

2. support end-user mobility at product request, milkrun, and supermarket warehouse areas;

3. Centauro 1.0 should have an ESB-based software architecture;

4. both, the PC-based client-side and the server-side components, must be developed in Java;

5. the client-side mobile clients must be developed in .net (compact framework);

6. must be compatible with SAP ERP and integrated with the SAP ERP mapped business processes, including those developed internally by Bosch;

7. must be compatible with available mobile terminals hardware, operating systems (Windows Mobile or Windows CE), and wi-fi infrastructure;

8. must include monitoring;

6.2.2.2. Development Project

The project team developed Centauro 1.0 recurring to the F3 software development process. F3 incorporates many activities related with getting the proper approvals to start the software development efforts and to check if the initial plan is feasible. F3 includes very few activities regulating the technical software development efforts. This focus on non-technical content induces, in the project team, a sense of facilitation in the creation of the deliverables related with the technical content.

F3 permits the project team not to create any technical deliverable at all. For that, it is crucial that a software development team using F3 has a very high maturity level. In F3, the technical software development deliverables are restricted to informal documents requesting grant approvals (e.g. exchange of emails to validate the requirements, or to allow the final customer acceptance of the software product) and only three textual documents (Requirements Specification, Test Plan, and Operational Manual).

To avoid missing technical content, the Centauro 1.0 project team decided to use the FURPS+ approach (Eeles, 2001), and to use UML for the functional requirements specification.

Since F3 does not explicitly state what technical development disciplines should be used, for the Centauro 1.0 project, the project team included three technical disciplines (Requirements, Design and
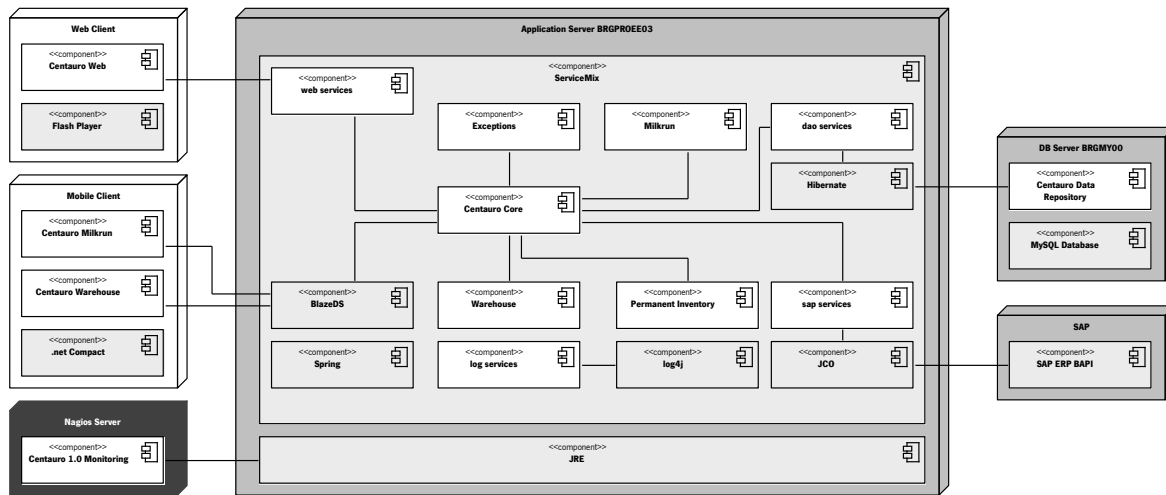
Figure 6.2.: Centauro 1.0 deployment diagram.

Implementation, and Test), and two additional non-technical disciplines (Environment, and Project Management).

Despite F3 induces a waterfall-like approach for its life-cycle, the project team decided to incorporate iterative and incremental characteristics to the project planning. For that, five iterations together with the five corresponding releases of incremental prototypes are planned (core, interfaces for web services, mobile milkrun, mobile warehouse, and backoffice).

6.2.2.3. Resulting Software System

Figure 6.2 presents the deployment of the software components of Centauro 1.0 software system and their connections. The web client machines run the web component on the Flash Player sandbox, connected to the web services server component. The mobile components for milkrun and warehouse, run on top of the .net framework and connect to the BlazeDS (Adobe, 2008a). BlazeDS is a server-based Java remoting and web messaging technology that allows the connection of back-end distributed data and pushes data to, among others, Adobe Flex clients. The application server BRGPROEE03 hosts an instance of the Apache ServiceMix 4 ESB. Inside ServiceMix are the server-side components of the Centauro 1.0 software system, namely the core, web services, exceptions, milkrun, dao service, core, warehouse, log services, permanent inventory, and sap services.

Also, integrated in the ServiceMix instance are the BlazeDS, Spring, log4j, JCO, and Hibernate frameworks. The BRGMY00 MySQL database server hosts the data for the Centauro 1.0 software system. The ServiceMix connects to the SAP ERP through the JCO component. The requirements for Centauro 1.0 do not include the ability to produce standard reports, only the possibility to extract raw data from the database server. The Nagios Server hosts the components for monitoring.

Business Strategy

Organizational Strategy   Information Systems Strategy

Figure 6.3.: The information systems strategic triangle (adapted from (Pearlson & Saunders, 2006)).

6.2.3. Theory Adjustments

One of the main purposes to conduct the Centauro 1.0 Action is to evaluate and define a standard executable model of business processes to use in MES projects, at Bosch CMP. For that, included in the proposal to transform business process models into software-executable models using MDA (Santos et al., 2013), we next present how we have conducted the study to define the standard.

To assure the quality of the software resulting from a business processes implementation project, it is advisable to select a business process language compatible with the organization where the processes will run. We check such compatibility by using several perspectives, later detailed in this section. We include in this section a comparison between five business process modeling languages: BPMN (OMG, 2009a), BPEL (Juric et al., 2004), XPDL (Shapiro, 2006), YAWL (Van Der Aalst & Ter Hofstede, 2005), and CPN (Jensen & Kristensen, 2009). Several languages are reviewed by Ko et al. (2009), namely by describing their technological characteristics and their strengths and weaknesses. Twelve business process languages are compared by Recker et al. (2009), according to a representation model proposed in (Wand & Weber, 1990), to establish differences to their representational capabilities in the information system domain. The most common approach to compare the modeling capabilities of the business process languages is the set of workflow patterns defined by van Der Aalst et al. (2003), which shows if the representation of a business process workflow is directly supported by the language.

The selection process of a business process language for a particular organization should not only be restricted to the comparison of the workflow patterns. An organization should not just be concerned with technological issues.

Thus, based in (Pearlson & Saunders, 2006), we propose that the selection process should be enlarged to incorporate the three strategies of the triangle shown in Figure 6.3, namely information systems, organizational, business.

The triangle relates the business strategy with the information system (IS) strategy and with the organizational strategy. The selection process for the adopted business process language for Bosch CMP, at Centauro 1.0 project time, takes into consideration the information systems strategic triangle. Since IS strategy is related to the definition of the implementation of the business processes in the IS solution, we base our comparison analysis on the workflows that each language supports. Regarding the organizational strategy, we base the comparison on the preferences of the development team. In what

concerns the business strategy, our comparison takes into account a set of aspects related with the alignment of the business process and the business strategy.

Information Systems Strategy   To compare the business process modeling languages by their benefit in the design and execution of the business processes, we conducted a study for the language functionality, using the workflow patterns defined by van Der Aalst et al. (2003). Table 6.1 describes the results of the comparison of the languages.

Table 6.1.: Comparison based on workflow patterns.

#	Workflow Pattern	BPML	BPEL	XPDL	YAWL	CPN
1	Sequence	+	+	+	+	+
2	Parallel Split	+	+	+	+	+
3	Synchronization	+	+	+	+	+
4	Exclusive Choice	+	+	+	+	+
5	Simple Merge	+	+	+	+	+
6	Multi-choice	-	+	+	+	+
7	Synchronizing Merge	-	+	+	+	-
8	Multi-merge	+/-	-	-	+	+
9	Discriminator	-	-	+	+	-
10	Arbitrary Cycles	-	-	+	+	+
11	Implicit Termination	+	+	+	-	-
12	Multiple Instance without Synchronization	+	+	+	+	+
13	Multiple Instances with a priori Design Time Knowledge	+	+	-	+	+
14	Multiple Instances with a priori Runtime Knowledge	-	-	-	+	-
15	Multiple instances without a priori Runtime Knowledge	-	-	-	+	-
16	Deferred Choice	+	+	-	+	+
17	Interleaved Parallel Routing	-	+/-	-	+	+
18	Milestone	-	-	-	+	+
19	Cancel Activity	+	+	-	+	+/-
20	Cancel Case	+	+	-	+	-
	Total +	11	13	11	19	13
	Information System Ranking (untied by +/-)	**4**	**2**	**5**	**1**	**2**

The table includes previous comparisons, which can be seen in (Mendling et al., 2006; van der Aalst, 2003b; Van Der Aalst & Ter Hofstede, 2005; van der Aalst et al., 2002). If the language supports directly the pattern, it is represented in the table by a "+". If the language does not support the pattern, it is represented in the table by a "-".

Organizational strategy   The need for this comparison lies in the fact that the project team members of the software development organization at Bosch CMP will be an important group of users of

Table 6.2.: Results of the surveys submitted to the IT project team.

Survey #	BPML	BPEL	XPDL	YAWL	CPN
1	0.76	0.76	0.76	0.65	0.84
2	0.70	0.80	0.79	0.94	0.99
3	0.64	0.74	0.79	0.83	0.78
4	0.59	0.60	0.79	0.57	0.59
5	0.62	0.72	0.79	0.70	0.76
6	0.67	0.88	0.87	0.90	0.95
7	0.15	0.38	0.35	0.34	0.30
8	0.34	0.53	0.59	0.49	0.48
9	0.78	0.79	0.92	0.69	0.75
Total	5.25	6.20	6.65	6.11	6.44
Organizational Ranking	**5**	**3**	**1**	**4**	**2**

the chosen business process language. On top of the intrinsic characteristics of the language, it is then necessary to conclude which business process language is the best suited for the profile and for the skills of its users. We conducted surveys at the IT department of Bosch CMP to assess the technological skills of the development team. The aim is to establish a language comparison.

In this Action, we base the structure of the survey on a collection of key characteristics of the business process language. We questioned the IT project team on their knowledge and confidence of use on the following areas: workflow modeling, graph-based modeling, XML, Petri nets, pi-calculus, business process modeling languages and notations, SOA, web services, protocols, brokers, ESB, and threading. Additionally, the team was questioned about BPM issues.

This organizational strategic study was helpful to characterize the project team regarding its knowledge on the business process to implement. Thus, the survey has included questions related to the knowledge and confidence of the team on:

- business processes, activities, key performance indicators, and strategic goals;
- BPM-based tools (e.g. BPM systems, EAI, ERP, CRM);
- business quality management (e.g. total quality management, six sigma, balanced score cards).

Table 6.2 presents the results of the survey, related with the business process languages. The answers express the level of knowledge for the presented standards, valued from a minimum knowledge of "1" until a maximum of "5". The presented values are relative results obtained by each of the languages in each of the surveys, resulting then in the sum of the values of the surveys for each language. The results of the surveys are represented in Table 6.2 and express the confidence of the user on using each language (for instance, survey #1 is 76% confident on using BPML, 76% on using BPEL, etc.).

Business Strategy   The third comparison relates with the specific aspects of the business environment, in this case referring to the software development industry. Helkiö et al. (2006) suggest

Table 6.3.: Business relevant aspects considered for comparison.

#	Business Aspect	BPML	BPEL	XPDL	YAWL	CPN
1	Maturity	4	4	4	3	5
2	Usability	4	4	4	3	3
3	Existing Tools	4	5	3	2	5
4	Online Tutorials	5	5	3	3	5
5	Ability to be Translated	5	5	3	4	4
6	Learning Effort	4	4	4	3	3
7	Transformation in OO Code	2	5	3	2	2
8	Implementation Costs	5	5	3	5	4
9	Portability	3	5	3	5	5
10	Interoperability	5	5	4	5	3
11	Security	5	5	3	5	3
12	Efficiency	4	5	2	5	5
13	Data Management	4	4	5	4	5
14	Integration with SAP ERP	5	5	3	4	3
	Total	59	66	47	53	55
	Business Ranking	**2**	**1**	**5**	**4**	**3**

some of these aspects as a basis to compare the tools supporting a language. Other aspects are the generic concepts for using a language in a business process implementation project (e.g. the language maturity, or the implementation costs) with the goal of determining the benefit in using one of these languages in the organization. Table 6.3 represents a set of characteristics, namely: language maturity, usability, available implementation tools, available online tutorials, possibility of translation to other languages, learning effort, ability to transform a business process language into object-oriented code, implementation costs, portability, interoperability, security, efficiency and data management, and the integration of the language with an ERP (e.g. SAP ERP).

For each language, the considered business aspects are scaled from "1" to "5". The value of each business aspect is based on technical specifications and discussion forums. We based the classification on our subjective judgment after analyzing the information for each language.

Final Comparison of Business Process Languages   After the comparisons of the three dimensions of the information system strategic triangle, the final results is collected in Table 6.4, where it is shown the ordered level of suitability obtained by each language. The result of each language is the sum of the values of the three comparisons. Lower values represent better results.

In the Table 6.4, the language with the best overall result in the final ranking is BPEL, mainly because it was considered the most adequate in the business strategy and also because it achieved good classifications in the information systems and in the organization strategies. For the #1 ranking of BPEL contributed significantly the good result obtained from the Bosch CMP software development team. With distinct software development teams, the order of the strategies may vary.

Table 6.4.: Final comparison of the business process languages.

Ranking	BPML	BPEL	XPDL	YAWL	CPN
Information System Strategy	4	2	5	1	2
Organizational Strategy	5	3	1	4	2
Business Strategy	2	1	5	4	3
Total	11	6	11	9	7
Final Ranking	**4**	**1**	**4**	**3**	**2**

6.2.3.1. Metrics and Lessons Learned

Table 6.5 depicts the maturity levels achieved in the Centauro 1.0 Action, according to the proposals of Table 4.3. The comparison with the previous SOL Action (see Table 5.3) shows a significant decrease in the Sound attribute, due to the F3 focus being on documentation and approvals and not on customer needs, and an increase on the Integration attribute, mainly because F3 is a standard in all Bosch organization. The value for the Implemented attribute remains the same, as F3 was executed rigorously, but the Systematic attribute has a decrease due to the missing guidance in F3 for the technical software content, leading into a less structured path of execution of the software development activities.

Table 6.5.: Maturity levels of the attributes of the Enabler in the Centauro 1.0 Action.

Element	Attributes to Score	Maturity Level
Approach	Sound	2
Approach	Integrated	4
Deployment	Implemented	4
Deployment	Systematic	2
Assessment and Review	Measurement	3
Assessment and Review	Learning	4
Assessment and Review	Improvement	4

The remaining Assessment and Review attributes keep the same values because the present Action is subjected to the same measurement, learning, and improvement cycle as the previous Action.

Table 6.6 depicts the Results of Centauro 1.0 Action. The project team managed to implement the 11 requested functional requirements. The same full implementation scenario occurs for the non-functional requirements.

The Centauro 1.0 project does not include any Quality Assessment due to the use of F3, and for that, the D3 dimension of analysis is not applicable.

The results of Effort and Duration present extraordinary values of 100% accuracy, mainly because the team was experienced in the planning (technical and organizational restrictions are well known to the team), and because Centauro 1.0 was a short-time project. The initial go-live date was postponed 11 days by explicit requirement of the customer and, for that, not accounted as a deviation in the duration.

The total amount of 9490 physical lines of source code are divided by 4472 lines of Java, 2319 lines of ActionScript, 1354 lines of XML, 748 lines of JavaScript, 492 lines of MXML, 53 lines of HTML, and 52 lines of CSS. Centauro 1.0 uses 11 software frameworks, namely the .net compact, flash player, BlazeDS, Spring, log4j, Hibernate, JCO, ServiceMix, MySQL, JRE, and Nagios. The notation of ">11" in the Used Software Framework result refers to the bundles used inside ServiceMix (e.g. related to JBI, NMR, OSGi, or CXF).

On the very same day that Centauro 1.0 went live, the customer presented three change requests. The change requests (book semi-product quantities at SMT shop-floor, book urgent outgoing semi-product, and book devolutions to the warehouse) demonstrates how F3 documents and clearances are not expressing the real situation of a project. In the present project, the F3 documents validating the requirements or approving the system to go-live did not induce in the project team the need to perform adequate requirements elicitation or to perform comprehensive functional tests.

Table 6.6.: Results of the Centauro 1.0 Action.

#	Dimension of Analysis	Metric	Value
D1	Requirements	Functional Requirements	11/11 = 100%
D2	Requirements	Non-Functional Requirements	8/8 = 100%
D3	Quality	Quality Assessments	not evaluated
D4	Cost	Effort	28 PD/28 PD = 100%
D5	Time	Duration	16 days/16 days = 100%
P1	Complexity	Physical Lines of Source Code	9490 SLOC
P2	Complexity	Used Software Frameworks	>11 USF
P3	Soundness	Change Requests	3 CR
U1	Reliability	MTTR	3 min
U2	Reliability	MTBF	>30240 min

The Centauro 1.0 business software was in operation for three weeks until the three change requests were implemented, and an updated version went live. For that, the MTBF result expresses that no failure occurred during that period. We calculated the MTTR value based on tests.

The lessons learned of the Centauro 1.0 Action are:

1. F3 follows a waterfall-like process model and does not embody iterative and incremental practices. For that, all the problems associated with a waterfall approach are present in the Centauro 1.0 project, namely the difficulty to deal with changes during the course of the project, or the need to express "perfect" requirements at the beginning of the project. As a consequence, in the very first day of use, major changes to Centauro 1.0 were requested by the business experts;

2. F3 embodies too many bureaucratic aspects, needed inside the Bosch global organization, in order to guarantee the authorizations and clearances for the local software development efforts at Bosch CMP. In F3, the project deliverables related with the technical development of software are simple signed statements (e.g. the proof that users had performed the functional tests is a signed document, without any need to present detailed test plans or to show the test results);

3. F3 has no technical software development content;

4. The software architecture used in Centauro 1.0 system is a base software architecture to run business processes. The business processes models are created recurring to a general purpose modeling language. The use of the ServiceMix-based software architecture performs according to the expectations and provides a base for software reuse. The ESB OSGi-based architecture allows modular software architectures and dynamic components, in the present Action by recurring to the Spring framework using the Dependency Injection software pattern. Nevertheless, the software architecture does not guarantee non-functional requirements, like the performance, availability, or the consistency of the data in all components of the software system;

5. The current software architecture does not allow that software components (implemented as OSGi bundles in ServiceMix) change their connections on the fly. For the time being, the Spring framework, despite allowing dynamic components, forces that the connections between components are described in a XML file that needs to be updated before the loading of the software system;

6. The BPEL is selected as the most adequate business process language to develop MES in Bosch CMP (see details in Paragraph "Final Comparison of Business Process Languages", in Section Theory Adjustments);

Additionally, the evaluation of the use of the free software frameworks, the use of the 4SRS automated technique to derive the candidate logical software architecture out of the use case model, and the use of design or architectural software patterns, expressed positive results and, for that, we will keep them in future projects.

6.2.3.2. Proposed Adjustments

Inside Bosch CMP, the development process engineer should tailor F3 by extending it with iterative and incremental approaches together with the release of prototypes. In such way, project teams using the extensions to F3 are able to conduct properly the testing activities, or to collect the feedback from the business experts since the early stages of the project. Additionally, F3 needs to include technical software development practices, such as modeling the business domain, or capturing and managing the requirements.

To maximize the business knowledge incorporated into a business software system, we propose to design and use a methodology that gets as input a set of BP-RMs and ends with a business software system. Such methodology should follow the business requirements expressed in the BP-RMs, and should allow the tailoring of the business process models by the business experts of the organization where the business software system will run.

The software architecture used in Centauro 1.0, based on ServiceMix with the business processes modeled in UML and implemented recurring mainly to the Spring framework, can be improved to

accommodate non-functional requirements (e.g. end-to-end performance for shop-floor clients). This way, the main concern of the future software development teams will be the functional requirements, probably expressed in business process models, and not if the software system is able to cope with non-functional requirements. For that, in the next Action we expect to develop an improved software architecture able to guarantee non-functional requirements.

Additionally, and to avoid model transformations, we propose that the new methodology and the software architecture are able to accommodate a business process language, in the case of Bosch CMP, the BPEL.

6.2.4. Social Environment

In the Centauro 1.0 Action, the author of this thesis acted as process engineer, business process analyst, software architect, designer, test analyst, and stakeholder[1]. Despite Centauro 1.0 project had a short-time schedule and the resulting software system is small-sized, the mapped business content is of major importance for Bosch CMP, due to the high monetary values involved in a SMT production process. Additionally, the accuracy in the control of the warehouses and in the movements of the SMT semi-finished products is crucial for a proper production planning, for production execution, and for inventory purposes.

The Centauro 1.0 project is the first project in Bosch CMP to use fully the F3 process. For that, extra attention from international management was given on the proper fulfilling of F3 demands.

The technical software development team is requested to fulfill, at the same time, all the bureaucratic demands of F3 and the technical aspects of the development process and of the developed product.

6.3. The Centauro 2.0 Action

Centauro 2.0 is a technical upgrade on the non-functional requirements implemented in Centauro 1.0, containing revised and detailed functional requirements, yet maintaining the same business scope of Centauro 1.0.

The Centauro 2.0 Action has a distinct environment from the previous Actions. The Centauro 2.0 software system did not have a go-live at Bosch CMP.

6.3.1. Purpose of the Action

The Centauro 2.0 Action is setup with the following goals:
- to design and use a methodology able to generate, and immediately execute, information and software systems for process-oriented organizations;

---

[1]The names for the process roles follow a RUP-like syntax and semantics, despite RUP is not used in this Action.

- the methodology to be proposed should be holistic, in the sense that it starts with BP-RMs and ends with a software solution that considers the business requirements expressed in the used BP-RMs;

- to avoid model transformations, we propose that the new methodology is able to use BPEL since its first stages until the software implementation. The software implementation must use a software framework, also to be proposed;

- to propose a software architecture, materialized in a software framework, to allow the business experts to manipulate the business content and the corresponding software implementation of business processes, independently of the software experts. The software framework should provide proper support for business relevant non-functional requirements. In order to incorporate the lessons learned from former Actions, in Centauro 2.0 the software framework should include SMX4 and ODE (to support BPEL).

6.3.2. Practical Action

The Centauro 2.0 Action uses a new methodology, called BIM (see Chapter 7 for a detailed description) to transform BP-RMs into business software systems. In this Action, UML loses importance in favor of BPEL, as the business process modeling language. We use UML activity diagrams to model the first design of the SCOR decompose process elements, despite those could also be initially designed recurring to BPEL. We use BPEL to model the business processes, and as an executable model (recurring to the ODE engine, running inside SMX4). The business scope required by Centauro 2.0 is a subset fully contained in the business scope provided by the SCOR, and, for that, SCOR is the only BP-RM used as input for the first phase of BIM.

BIM uses four new concepts during its life-cycle. The first concept, called Instantaneously Available Organization (IAvO), proposes the creation of templates for detailed business processes and roles, which afterward business entrepreneurs can immediately use. This proposal extends the concept of BP-RM.

The second concept, designated Organizational Aspect (OA), relates with orthogonal characteristics that an organization should exhibit. For instance, an organization may want to implement some quality standard that is cross-cutting to all business processes.

The third one, the Process Framework (PF), is the main deliverable of BIM. The PF, during the BIM life-cycle, passes through a series of states until it reaches an implementation in software.

The fourth concept, the Orchestrated Business Object (OBO), is a software implementation of a business entity together with its associated functionalities and data, compliant with one or more BP-RMs, thus realizing a Domain Model pattern.

Instantaneously Available Organizations   Every day, entrepreneurs create new organizations. Every day, entrepreneurs think they have discovered a killer product or service that will bring a huge advantage over their competitors, either because the forthcoming product is an absolute premier, or because it incorporates new features that will overwhelm the competition. This willing in the business creation process can be truncated by organizational problems. For the entrepreneur, the internal configuration of his own organization, as well as the interfaces with suppliers and customers, can present difficult obstacles, either because the internal configuration is not properly defined or, even worst, because the entrepreneur does not have any idea of what business content should be embodied into his own organization.

The set of common practices shared among organizations operating inside the same business contexts is not negligible. Furthermore, an organization can, and should, perform benchmarking activities of the processes of other organizations, either operating or not inside the same vertical market. For that, it seems plausible that business practices, materialized in business processes, may be shared among different organizations.

For instance, a production plant can look at the core processes of another production plant, but may also look at an air carrier company to check their warehouse handling processes in order to incorporate those benchmarked practices into its own core activities. This means, that the core processes of an organization can embody practices coming from processes of other organizations placed in different vertical markets. Benchmarking activities can be applied to core processes, but they can also be applied, even in a higher scale, for management processes and for support processes because these types of processes are not so specific as the value-added ones.

For those reasons, we propose the concept of Instantaneously Available Organization (IAvO). The main goal behind using the IAvO concept is to have out-of-the-box organizations that entrepreneurs can pick and materialize on a concrete organization. Existing organizations can also use this approach when they want to redesign themselves.

The IAvO concept assumes that entrepreneurs and organizations reach a maturity level high enough that they can perceive the advantages of using standard best practices instead of only imposing their own ideas in the designs of the processes.

On top of the process definitions, an IAvO should also proposes the standard business roles needed to run each process. The information about the standard roles can help entrepreneurs foreseeing the quantity and qualifications of the human capital to be hired.

Despite the business competition among organizations, from an information systems perspective, organizations can share their business processes with the competition. This sharing can occur because the most important differentiations among competitors are based on the characteristics of the products, on core business specific activities (e.g. a shop-floor layout), and mainly because the most decisive factor for the success of an organization is their own human capital.

As an example of collaboration among organizations that share markets, and probably processes,

the SCOR framework was developed and supported by a council (the Supply-chain Council[2]) as a cross-industry standard tool for the supply-chain management, yet incorporating amongst its members organizations that are competitors (e.g. IBM and Oracle). Such BP-RM enables its users to address, improve, and communicate business practices between all the interested parts.

BP-RMs can provide enough variability in order to allow an organization to map properly into its reality what is foreseen in the BP-RM. An example is the different strategies that organizations can choose (Make-to-Stock, Make-to-Order, and Engineer-to-Order Production Execution) in the Make business process of SCOR. Despite details of each organization, a very high degree of generic processes is shared among organizations operating within the same business area, or even between organizations operating in distinct business areas.

Starting with BP-RMs, work can be done to provide business templates that will support an entrepreneur when creating an organization. These business templates, the IAvOs, can provide a complete definition of the internal structure of an organization, as well as a definition of the relations with interfacing organizations.

This way, an IAvO can include:

- a set of business processes that are adequate for a vertical business domain, or, more generically, a set of business processes that only embody generic organizational rules (like country specific financial or labor rules);

- the relations between the described business processes;

- the set of business actors to execute the business process, as well as the associations of business actors with the business processes;

- an estimation formula to derive the quantity of human power needed to staff each business actor (e.g. depending on the external business context, on the internal monetary restrictions to the investment);

- pre-bundled compliance with organizational or quality standards, like EFQM or ISO 9001[3].

To choose adequately an IAvO, the entrepreneur must define the characteristics of the foreseen organization. These characteristics can include the vertical market in which the organization will operate, the size of the organization, or to what standards must the organization adhere. After the entrepreneur picks a proper IAvO, tailoring activities can start, like defining the name and the quantity of warehouses of the organization. This tailoring activities have as premises that the organization contents is derived from business needs and drivers, not the opposite, like when the number of employees, their skills, and their tasks are predetermined.

Of course, the entrepreneur is the owner of the organization he has the power to change what he may

---

[2]http://www.supply-chain.org

[3]

     ◦ http://www.iso.org

consider relevant, but that change will no longer be a guess. Instead, it will be an explicit opinion based on an input from the IAvO.

BIM considers IAvOs in its first phase, where the actors of the development process make the selection of the standard business processes to use in the organization, for later implementation in the subsequent phases of BIM.

IAvOs extend BP-RMs, in the sense that IAvO are more concrete than BP-RMs.

Organizational Aspects   An Organizational Aspect (OA) is a cross-cutting organizational concern.

The concept of OA is inspired on the concept of Aspect, as introduced in AOP (Kiczales et al., 1997). During the first BIM phase, we propose that the business analyst picks standard BP-RMs and/or IAvOs and, by using one or more OAs, he creates an adequate business process model for the organization under consideration.

The business process model incorporates a set of activities semantically compliant with some organizational standard. If some cross-cutting concern within the organization exists, then an OA can model it. The OA concept allow the creation of BP-RMs, or IAvOs, compliant with some organizational standards (e.g. the EFQM excellence model).

OAs can also be used to express traversal concerns inside the organization, like when an organization wants to have traceability for its products or for its shop-floor processes, or wants to adhere to environmental, health, or safety regulations. The OA extension mechanism, similar to that of the AOP's advices, can be a basis to use excellent value-added processes without ignoring the compliance with external business drivers. At the end, the business processes can include all the desired characteristics of the chosen orthogonal quality standards. Of course, conflicts may arise, such as when creating a lean production process while maintaining the traceability data, but they will be more easily tracked and explicitly solved.

BP-RMs with OAs can also provide gains to organizations, when:

1. a new organization is to be created from scratch (e.g. a new subsidiary), assuring the correct transposition from the mother organization of:

   a) the local processes that may be compliant with the central processes;

   b) the standards requested locally and immediately fulfilled (e.g. social responsibility);

2. country-specific customization of processes (e.g. human resources) can be obtained from an IAvO;

3. for an existing organization, during a business process reengineering project, the current processes can be compared with IAvOs that are compliant with some standard. This comparison provides an accurate visualization of the as-is and of the to-be states of the organization, and ensures that the organization can achieve the standard certification because a quantitative measure of the work to be done, until the target situation is achieve, is available.

Process Framework   One of the main problems when creating software is to close the gap between the problem features and the solution features. Normally, the client hands over this activity to the project team. Using some kind of software development process, the project team provides a solution that meets the client's expectations. For that, the project team uses:

- a process, which may includes the use of incremental and iterative builds;
- some activities to fulfill the quality standards required by the development process;
- the guidance to achieve the targeted software architecture;
- the execution of assessments together with stakeholders in order to monitor the project phases;
- the change management.

Despite all best practices, software development processes are far from guarantee a proper implementation of client's expectations for the software product. This fact is not always caused by the bad performance of the software engineers. It may also happen that the client organization lacks a strong and sound knowledge about its own requirements. This lack of knowledge from the organization may endanger the software development effort. In such a way, it is always present the necessity to improve the degree of fulfillment of the expectations of the client, even those not explicitly stated to the project team. In addition, even with a good client and with a good software development team, a software project may fail, among other reasons, due to:

- missing technical skills inside the client organization, like the ability to perform proper functional or regression tests;
- a peak use of the human resources assigned to the project in parallel operational activities. In such case, it can happen that the organization re-staffs the project with less qualified human resources.

The BIM, recurring to the Process Framework (PF) concept, proposes to help closing the gap between the client needs and the respective implementation in software. One of the proposed ideas is to use a set of orchestrated business objects to implement business processes, maintaining the consistency between the models during a software implementation process.

The opportunity to generate automatically a business software system can improve the resulting product quality, by eliminating the errors during manual model transformations that occur in a more conventional software development process. In addition, BIM proposes to use the orchestration of components, which can results in the use of components developed and tested prior to project time.

The PF is a set of models of business processes, and tools that allow the development project stakeholders to manage business processes at different stages of BIM. A PF contains:

1. a set of BP-RMs or IAvOs;
2. a set of allowed activities that can be performed on the BP-RMs;
3. a set of process actors that can perform the activities;
4. a defined state;

5. a business ontology.

The PF is the main artifact manipulated during the BIM process life-cycle. To help handling the PF, visualization tools are desirable. These tools should be able to depict the PF at its different states and should be able to interpret the language of the business process models.

When the PF reaches a software-implemented state, the PF provides a semantically sound software architecture for process-oriented organizations, by maintaining traceable business definitions across all states, during all the phases of BIM. Also and very important, the software-implemented PF can be a basis to freely use software from different sources (either own developed, free, or commercial) in order to provide organizations with the best software components to support their business needs and definitions.

A configured software-implemented PF implements a concrete business architecture.

Orchestrated Business Objects   An Orchestrated Business Object (OBO) is a piece of software that implements one or more business entities and (sub)processes inside some specific business ontology.

OBOs expose the functionality and data of a specific business area to the software-implemented PF in order to allow the orchestration of the components of the business processes.

Each OBO is a black box and it should be interchangeable with other OBO implementing the same business entities and processes.

An OBO exists in a state, defined by its use inside the software-implemented PF, and it must be compliant with the PF. Software engineers should choose from the market a set of OBOs in order to fulfill the requirements of the process-oriented organization, and thus to create its own software behavior based on the best suitable components (OBOs) available.

Prior to project time, a set of OBOs should be available for use when instantiating business processes in the last BIM phase. The concept of creating a project for software development based on previously developed components is similar to the one proposed by software factories (Greenfield & Short, 2004) approach. The OBO concept addresses the need to have different pieces of software orchestrated to implement a specific business process, with the capacity to be interchangeable and without causing any disruption in the behavior of the software system. The quantity of needed OBOs, previously available to a BIM execution, is a clear indicator of the speediness of the BIM execution. The higher the number of available OBOs, the lower the time needed to complete BIM. If some new OBO is necessary during a software development process using the BIM (e.g. due to customizations or due to changes in the business processes), its development should be made during the first phases of BIM.

Therefore, a software development effort for OBOs has as output:

- the set of software components (OBOs) needed and yet not available, in order to fully support all the business processes;

- descriptions on how to use the newly created OBOs, which may include conditions to use them, scope, or protocols for method invocations;

- tools, if needed, to integrate the new business objects into the set of already available business objects.

Software engineers can publish and make OBOs available (e.g. via web services) for their later use inside orchestrations in the software-implemented PF. The interface implementation needed to be compliant with the software-implemented PF can be materialized inside an OBO through the remote invocation of an available API, independently of the used technology (e.g. web services, business API, remote procedure call).

The assembly by orchestration approach used for OBOs functionality and data, as stated in (Greenfield & Short, 2004), allows the assembly of new OBOs without creating dependencies between business objects during the design, compile, and deployment times. This approach is also a form of mediation, as presented in the Mediator pattern (Gamma et al., 1995).

Each OBO is also an implementation of the Domain Model enterprise integration pattern (Fowler, 2003) for a part of a business process.

6.3.2.1. Implemented Business Processes

Next, we present the execution of the discipline Business Requirements from the BIM (see Figure 7.1). Centauro 2.0 functional requirements are initially derived from a BP-RM. Since Bosch CMP is an existing organization and no radical transformation into its internal shape is required, then no IAvO is needed for Centauro 2.0. As BIM proposes, in the Centauro 2.0 project there is not a traditional requirements collection activity, but a statement for the top level requirements, in the present case based on SCOR.

The high-level technique used to derive the functional requirements out of SCOR business processes is presented in Algorithm 6.1. This approach is compliant with the "Steps to Establish SCOR Process Models" (Council, 2008a).

In Step 1, the generic descriptions are collected from the requirements of former Centauro 1.0 Action, expressed in the Figure 6.1.

In the Step 2, we map the business content collected in Step 1 into the SCOR processes. The mapping is next presented in Paragraph "Map Centauro 2.0 Generic Descriptions into SCOR process IDs".

We do not use the swimming lanes proposed in Step 3 because only two organizational sub-entities of Bosch CMP (Production and Logistics) are among the stakeholders and the boundaries and responsibilities of the work are regulated by work instructions in the Bosch CMP quality manuals. In Step 4, the workflows are implemented in BPEL. The descriptions and eventual relevant information, proposed in Step 5 and 6, are shown in Tables 6.8 and 6.9.

6. Running Business Process Reference Models in Software Frameworks

---

Algorithm 6.1 Steps to establish SCOR process models.

---

1. obtain generic descriptions (this is what people describe);

2. map these generic descriptions to SCOR process IDs (normalize);

3. create swimming lanes to reflect organizational boundaries;

4. create workflow with these SCOR processes;

5. add description to workflows to reflect inputs/outputs of the processes;

6. optionally, add other relevant information;

Map Centauro 2.0 Generic Descriptions into SCOR process IDs   From the complete business scope of SCOR, expressed by the SCOR top level process types (Plan, Source, Make, Deliver, Return), Centauro 2.0 is required and targeted by the business experts for Source (S) and Deliver (D) process types.

The second level, or configuration level, for the process type Source (S) includes the following process categories:

- **(S1)** Source Stocked Product: this process may include ordering, receiving, and transferring of raw material items, sub-assemblies, products, or services based on aggregated demand requirements;

- **(S2)** Source Make-to-Order Product: the processes of ordering and receiving product or material that is ordered (and may be configured) only when required by a specific customer order;

- **(S3)** Source Engineer-to-Order Product: the processes of identifying and selecting sources of supply, negotiating, validating, scheduling, ordering and receiving parts, assemblies or specialized products or services that are designed, ordered and/or built based on the requirements or specifications of a specific customer order;

- **(ES)** Enable Source: This is a collection of processes associated with managing and monitoring the Source process data, performance, and relationships. It is a set of enablers for sourcing processes.

The characteristics required by Bosch CMP for Centauro 2.0 fit into the S1 process category specification. For that, only S1 and ES process categories are considered for implementation. ES process category is included because it embodies S1 enablers.

The third level, the process element level, for the S1 process category includes the following decompose processes:

(S1.1) Schedule Product Deliveries;

(S1.2) Receive Product;

(S1.3) Verify Product;

162

(S1.4) Transfer Product;

(S1.5) Authorize Supplier Payment;

In the scope of Centauro 2.0, the decompose process S1.5 is not applicable because the supplier (the SMT production area) is internal to the organization. All the remaining decompose processes for S1, Source Stocked Product, remain applicable.

Also at the third level, the process element level for the ES process category includes the following decompose processes:

(ES.1) Manage Sourcing Business Rules;

(ES.2) Assess Supplier Performance;

(ES.3) Maintain Source Data;

(ES.4) Manage Product Inventory;

(ES.5) Manage Capital Assets;

(ES.6) Manage Incoming Product;

(ES.7) Manage Supplier Network;

(ES.8) Manage Import/Export Requirements;

(ES.9) Manage Supply-chain Source Risk;

(ES.10) Manage Supplier Agreements;

In the scope of Centauro 2.0, the decompose processes ES.5 to ES.10 are not applicable because, again, the supplier is internal to the organization.

For the process type Deliver (D), the second level, or configuration level, includes the following process categories:

- **(D1)** Deliver Stocked Product: The process of delivering product that is sourced or made based on aggregated customer orders, projected orders/demand and inventory re-ordering parameters. The intention of D1 is to have the product available when a customer order arrives, in the case of Centauro 2.0, to avoid the internal customer to stall waiting for sub-assemblies;

- **(D2)** Deliver Make-to-Stock Product: The processes of delivering product that is sourced, configured, manufactured, and/or assembled from standard raw materials, parts, ingredients or sub-assemblies, in response to a specific firm customer order. A reference to the customer order is exchanged with the sourcing or make process, and attached to, or marked on, the product. Products in stock are identifiable by customer order, through labeling and inventory data management;

- **(D3)** Deliver Engineer-to-Order Product: The process of obtaining, responding to, and allocating resources for a customer order that has unique requirements or specifications and delivering a product that is partially or fully designed, redesigned, manufactured, and/or assembled from a bill of materials or recipe that includes one or more custom parts or ingredients;

- **(D4)** Deliver Retail Product: Deliver Retail Products are the processes used to acquire, merchandise, and sell finished goods at a retail store;

- **(ED)** Enable Deliver: The collection of processes associated with managing and monitoring Deliver process data, performance and relationships.

The requirements for Centauro 2.0 fit into the D1 process category specification. For that, only D1 and ED process categories are considered for implementation. ED process category is included because it embodies D1 enablers.

The third level, the process element level, for the D1 process category includes the following decompose processes:

(D1.1) Process Inquiry and Quote;

(D1.2) Receive, Enter and Validate Order;

(D1.3) Reserve Inventory and Determine Delivery Date;

(D1.4) Consolidate Orders;

(D1.5) Build Loads;

(D1.6) Route Shipments;

(D1.7) Select Carriers and Rate Shipments;

(D1.8) Receive Product from Source or Make;

(D1.9) Pick Product;

(D1.10) Pack Product;

(D1.11) Load Vehicle & Generate Shipping Docs;

(D1.12) Ship Product;

(D1.13) Receive and Verify Product by Customer;

(D1.14) Install Product;

(D1.15) Invoice;

In the scope of Centauro 2.0, the decompose processes D1.1, D1.7, D1.13, D1.15 are not applicable because the supplier is internal to the organization. D1.14 is not required. The remaining decompose processes for D1, Deliver Stocked Product, remain applicable.

Also at the third level, the process element level, the ED process category includes the following decompose processes:

(ED.1) Manage Deliver Business Rules;

(ED.2) Assess Delivery Performance;

(ED.3) Manage Deliver Information;

(ED.4) Manage Finished Goods Inventories;

(ED.5) Manage Deliver Capital Assets;

(ED.6) Manage Transportation;

(ED.7) Manage Product Life-cycle;

(ED.8) Manage Import/Export Requirements;

(ED.9) Manage Supply-chain Deliver Risk;

In the scope of Centauro 2.0, the decompose processes ED.8 and ED.9 are not applicable because, again, the supplier is internal to the organization.

Table 6.7 presents the listing of the two process types, their corresponding four process categories, and, finally, the respective 25 decompose process elements constituting the business scope of Centauro 2.0.

Table 6.7.: Listing of decomposed process elements of SCOR for Centauro 2.0.

Process Type	Process Category	Decompose Process
(S) Source	(S1) Source Stocked Product	(S1.1) Schedule Product Deliveries
		(S1.2) Receive Product
		(S1.3) Verify Product
		(S1.4) Transfer Product
	(ES) Enable Source	(ES.1) Manage Sourcing Business Rules
		(ES.2) Assess Supplier Performance
		(ES.3) Maintain Source Data
		(ES.4) Manage Product Inventory
(D) Deliver	(D1) Deliver Stocked Product	(D1.2) Receive, Enter and Validate Order
		(D1.3) Reserve Inventory and Determine Delivery Date
		(D1.4) Consolidate Orders
		(D1.5) Build Loads
		(D1.6) Route Shipments
		(D1.8) Receive Product from Source or Make
		(D1.9) Pick Product
		(D1.10) Pack Product
		(D1.11) Load Vehicle & Generate Shipping Docs
		(D1.12) Ship Product
	(ED) Enable Delivery	(ED.1) Manage Deliver Business Rules
		(ED.2) Assess Delivery Performance
		(ED.3) Manage Deliver Information
		(ED.4) Manage Finished Goods Inventories
		(ED.5) Manage Deliver Capital Assets
		(ED.6) Manage Transportation
		(ED.7) Manage Product Life-cycle

After the selection of the three upper levels of the SCOR business processes (presented in Table 6.7), then in the forth level, the implementation level, organizations can implement supply-chain practices that are unique to their needs. Level 4, and lower, defines specific practices of a given organization in order to achieve competitive advantages, to adapt to changing business conditions, or to adapt to the

Table 6.8.: SCOR metrics applicable to D1.8 tailored for Centauro 2.0 software architecture.

Performance Attribute	SCOR Metric	Tailoring for Centauro 2.0	Centauro 2.0 Architectural Component
Reliability	None Identified	Not used	Not applicable
Responsiveness	Receive Product from Source or Make Cycle Time	Used	Implemented in a reporting system component
Agility	None Identified	Not used	Not applicable
Costs	Cost to Receive Product from Source or Make	Not used. The material supplier is internal to the organization	Not applicable
Asset Management	None Identified	Not used	Not applicable

physical conditions of the location where it operates.

In Centauro 2.0, we propose the use of OBOs. OBOs should be defined at the level 4 to allow the compliance with the upper levels of SCOR and to allow the implementation of unique practices granting business advantages to a particular organization.

SCOR lists all the inputs and outputs of each decompose process element. Those inputs and outputs may come from SCOR (e.g. from other decompose process elements), or may come from external content outside SCOR scope (e.g. from the accounting sub-process of controlling). During the execution of BIM, the information about the inputs and outputs of the decomposed processes elements, present in the SCOR, is the base for the definition of the interfaces of the OBOs software components.

Next, we present only present the details for the mapping of the D1.8 process element out of the 25 process elements presented in Table 6.7. The remaining 24 decompose processes follow the same procedure. Table 6.8 presents the performance attributes foreseen by SCOR for all business processes, independently of their level of abstraction. Table 6.8 presents also the metrics applicable only applicable to the D1.8 decompose process tailored for the Centauro 2.0 software architecture.

SCOR also proposes the particular metrics to use a business process. The abstract metrics proposed by SCOR for the performance attributes, do not materialize in concrete metrics for the reliability, agility, and asset management in the particular case of D1.8, neither exist additional requirements from Bosch CMP. For that, Centauro 2.0 will not use those metrics as requirements, and, consequently, we will not design any architectural component for those performance attributes. Despite SCOR proposes a metric for the cost performance attribute (i.e. the cost to receive product from source or make metric), Bosch CMP decided not to use that metric because the material supplier is internal and there is also a short physical path in the material movements, leading to a not significant cost for Bosch CMP. Consequently, we will also not design any Centauro 2.0 architectural component to support the cost performance attribute for D1.8.

The responsiveness performance attribute, materialized in the receive product from source or make cycle time metric, is implemented in a reporting system component.

The best practices for D1.8 can be found at Table 6.9. In addition to the best practices suggested by SCOR, we added an additional best practice, the 4 Eyes principle. This principle comes from Bosch CMP directives to avoid mistakes in the data collection when it is foreseen to be used in inventories or other business processes where the reliability of data is important. The best practices suggested by SCOR, like the cross-docking, and the downloading of the purchase order and the advanced ship notices for automatic receiving and put-away, are not used in Centauro 2.0. We do not use cross-docking because the warehouse used in Centauro 2.0 is a traditional distribution center with storage. We also do not use the downloading of the purchase order and the advanced ship notices for automatic receiving and put-away because purchase orders are not used in the internal process and also because the ship notices are already inserted in Centauro 2.0 data, when a container arrives at the warehouse.

The automatic identification best practice is guaranteed via the use of PDAs with barcode readers with wi-fi connections, already existing since SIIA (in the SMT production lines), and Centauro 1.0 (in the SMT semi-finished products warehouse). Centauro 2.0 does not need any tailored architectural software component to guarantee the automatic identification best practice because the PDA embedded software handles all the barcode reading and wi-fi needs.

Table 6.9.: SCOR best practices applicable to D1.8 tailored for Centauro 2.0 software architecture.

Best Practice	SCOR Description	Tailoring for Centauro 2.0	Centauro 2.0 Architectural Component
SCOR: Automatic Identification	Bar Coding & Radio Frequency Communications	Used	Implemented via the use of PDA with barcode readers for all movements of materials. Since barcode readers emulate manual data entry, no extra software component is needed
SCOR: Cross-docking	Used in many distribution centers (DC) to increase inventory velocity while maintaining shipping efficiency. In a traditional DC, the receiving process is disjointed from the shipping process and storage acts as an intermediary between the two processes. Cross docking actively links the receiving and shipping processes. In a DC, both cross-docking (no storage) and traditional (with storage) operations might take place	Not used	Not applicable
SCOR: Download Purchase Order & Advanced Ship Notices for Automated Receiving and Put Away	Integration with Procurement Systems & Electronic Commerce with Vendors	Not used	Not applicable
SCOR: Merge-in-Transit	Merge-in-Transit is a practice to combine items from multiple sources into a single customer shipment. This includes items on stock in the distribution center, from which the shipment is sent, items on stock in other distribution centers, items on stock elsewhere (e.g. at a plant or a supplier) as well as make-to-order items. The items to be merged are cross-docked from inbound receipt to outbound shipping. Merging is usually performed in a shipper's distribution center (DC) or in a carrier's terminal	Used. Materials from different production lines are collected in a single shipment into the intermediate storage area via the normal use of milkrun routes. No need for special requirement.	Implemented in Process D1.8 for single source shipments. No additional implementation needed other than identifying the source location at picking time
4 Eyes principle (Not present in SCOR)	To avoid mistakes between physical inventory and information systems inventory data during the movements of materials. For that is advised a cross checking of the data entered at both sending and receiving time	Used. Users sending and receiving materials are not aware of the data inputted by a colleague. Validation is done via software	Implemented via OA

As expressed in Table 6.9, the merge in transit best practice is a requirement. Bosch CMP collects semi-finished products from the SMT production lines via a milkrun cycle, meaning that during each milkrun cycle all semi-finished product types available at any SMT production line are collected and, then, dropped at the warehouse. For that, the milkrun approach already merges in transit any product type, and consequently the Centauro 2.0 project team does not need to develop any special architectural components.

The last suggested best practice from Table 6.9, the 4 Eyes principle, demands that a new software component should be created. In Centauro 2.0, the implementation is done by recurring to an OA.

Creating the BPEL process for the SCOR (D1.8) Receive Product from Source or Make

According to the SCOR, the D1.8 decompose process includes the tasks related with receiving products, verifying products, recording product receipt, determining put-away location, putting away, and recording location. D1.8 may also include quality inspection activities, not applicable in Bosch CMP.

In Figure 6.4, we present an UML activity diagram representing the outline of the tailored D1.8 for Centauro 2.0. The warehouse receives the SMT products in containers. Bosch CMP does not allow a container to transport distinct types of product. After the reception of the products, an action signal triggers the execution of the OA (4 Eyes). In the 4 Eyes OA, the the quantity and type of product inside the received container are read. Afterward, a validation step is performed, recurring to the data about quantity and type of product for the same container, registered in a predecessor business process when dispatching the container from the production line into the warehouse. If the values of quantity and type do not match for the particular container under validation, then a divergence is recorded and the warehouse supervisor is informed. If the values of quantity and type match, then the 4 Eyes OA ends and the flow of actions for D1.8 continues.

After, the recording of the product reception is done, the put-away location is calculated, and the putting away for the warehouse shelf is performed. D1.8 ends with the record of the location where the handled container is stored in the warehouse.

An important topic is the handling of exceptions at the business level, like for instance, when determining what actions must be performed when the warehouse shelves are completely filled. In Centauro 2.0, we opted not to depict the alternative paths at the abstraction level presented in Figure 6.4. Instead we decided to handle them a lower abstraction level, namely in the software components implementing each action. This decision is based in the level of abstraction that an OBO should have for a concrete BP-RM project implementation. We made this architectural decision based on the following criterion:

Criterion 1: "The OBOs supporting a particular BP-RM should be designed at the higher abstraction level where customization of detailed activities can occur."

Following the criterion 1, OBOs can act as drivers between the standard BP-RM activities and the detailed customized activities proprietary of a given organization.

Additionally, we advocate that organizations should be prepared at a physical level to handle business exceptions, like when no more warehouse shelves are available, and should not expect that a software system is always able to solve such unsolved business exceptions at the physical level.

The decision to not handle the business exceptions at the abstraction level presented in Figure 6.4 can lead into process traces with empty outputs in some of their actions. For instance, when the action to determine the put-away location ends with a status where no more containers can be stored
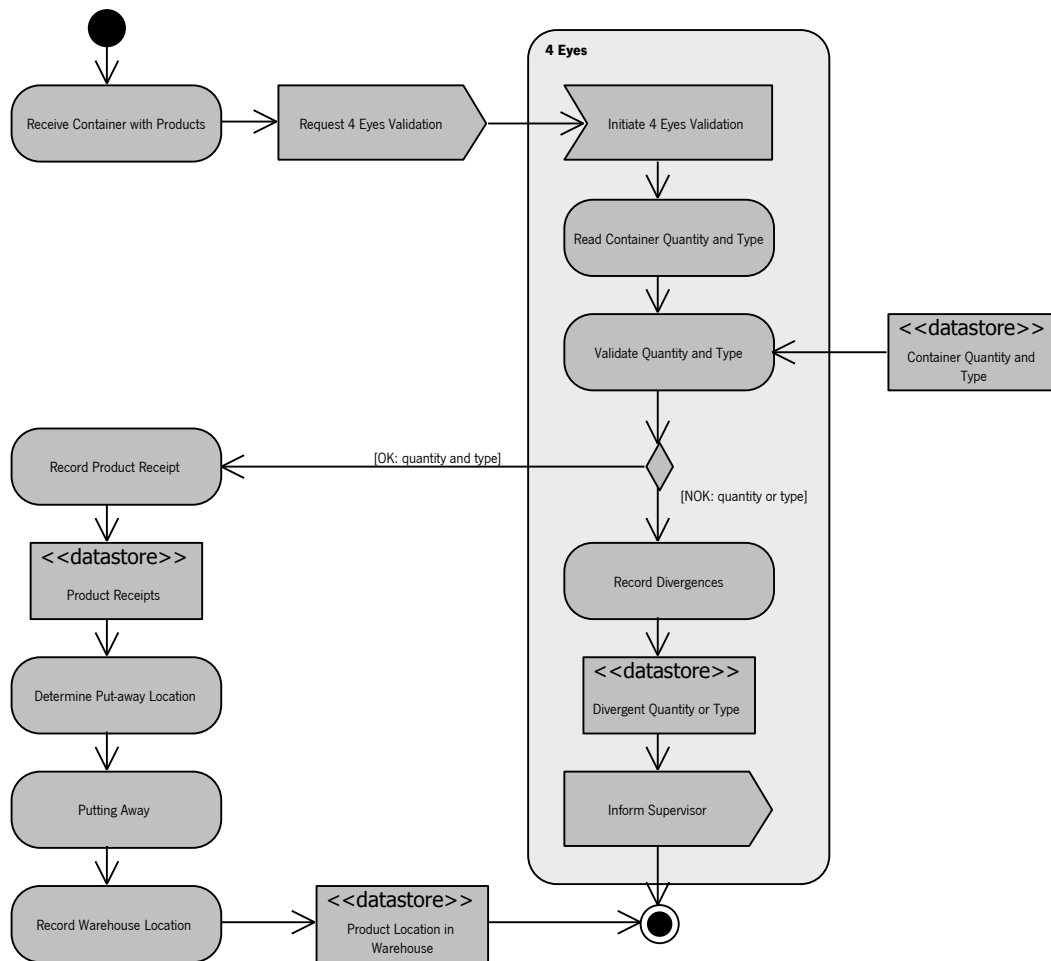
Figure 6.4.: Tailored SCOR D1.8 process element.

at the warehouse shelves, then the subsequent actions would not make sense to execute and the process would be immediately terminated. Nevertheless, we opted to execute always the complete set of actions until the process ends, forcing the execution of the subsequent actions of the action where the business exception occurred. If the business exception occurs, the consequences of such approach are the putting away in a physical location outside the warehouse shelves (e.g. in an input queue used to temporarily place the containers), allowing the subsequent actions to be executed (i.e. putting away the container in the input queue, and record their position in the input queue).

Figure 6.5 depicts a graphical representation of the D1.8, recurring to an abstract process in BPEL. BPEL abstract processes can later be materialized into runnable processes according to the specific characteristics of the running environment. The process of Figure 6.5 is the BPEL implementation of the outline process presented in Figure 6.4.

Not all activities and actions of the outline process are implemented in BPEL, namely the send signal action "Request 4 Eyes Validation", which is implemented with other technology (i.e. a Camel route). All the actions inside the "4 Eyes" activity are contained in a single OSGi bundle triggered by the Camel route when intercepting the flow between "Receive Container with Products" and "Record Product
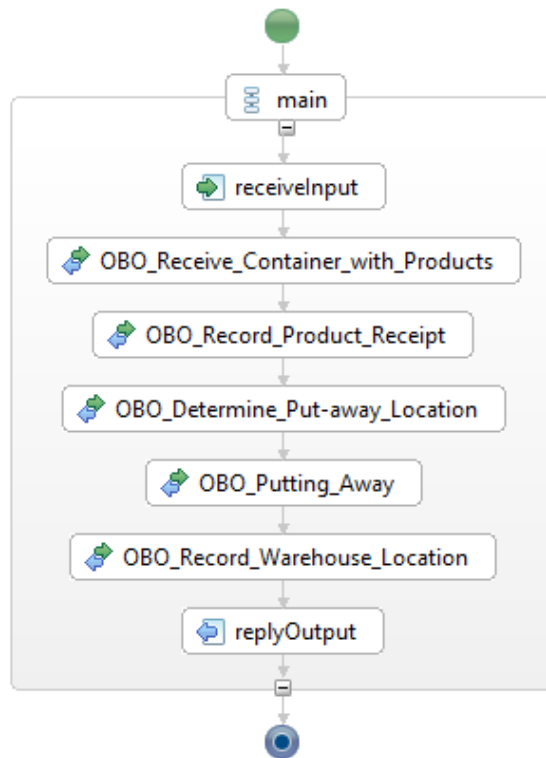
Receipt" actions.

Figure 6.5.: D1.8 modeled in BPEL.

In (Santos et al., 2013), we propose a transformation of business process models into software-executable models using the MDA, recurring to BPEL processes. With this transformational approach, the BPEL abstract process is the platform-independent model (PIM), and the BPEL executable process is the platform-specific model (PSM).

The BPEL process of Figure 6.5 is a synchronous process, starting by receiving the input via a service address and terminating by replying the output, with its caller waiting for process completion. D1.8 could also have an asynchronous implementation, but, according to Bosch CMP production concepts, the clarity and standard work procedures are favored to the throughput of an activity. The depicted process has a simple sequential flow with invoke actions to trigger each of the OBOs (from OBO_Receive_Container_with_Products until OBO_Record_Warehouse_Location).

6.3.2.2. Development Project

The methodology proposed and used in Centauro 2.0 is BIM, later detailed described in Chapter 7.

6.3.2.3. Resulting Software System

Figure 6.6 presents the Seppo. Seppo is an implementation for the PF concept. Seppo is a high-level software architecture based on SMX4, ODE, and BPEL, that overcomes the need for model transformations of the business processes in the PF, during the BIM life-cycle. Seppo allows OBOs and OAs to be implemented recurring to OSGi bundles. OBOs orchestrations can be implemented with BPEL orchestrations deployed into the ODE. The Seppo software architecture is conceived based on the following main non-functional requirements:

1. consistency

    a) goal: always consistent;

    b) solution: nested transactions, Idempotent consumer;

2. end to end performance for real-time clients

    a) goal: less than "n" seconds to receive a message feedback (the "n" value depends of the functional requirement);

    b) solution: asynchronism; content-based routing;

3. availability:

    a) goal: 100% available;

    b) solution: sequential reliable queues for messages from real-time clients and also to ERP input data, using idempotent consumer, and a local repository for the ERP data;

4. support and maintenance

    a) goal: MTTR inferior to one minute;

    b) solution: monitoring via Advisory Topics, alert rules, boot-up scripts.

Seppo embodies technological decisions and solutions to cope with the expressed non-functional requirements. There are three main servers. One for messaging and web services, Hipotalamo1, another for BPEL executions and communications with the data layer, Matrix1, and a database master server. Each one of these three servers has an own running copy that takes on the respective responsibilities, in the case of a crash on the master, via a shared-IP mechanism. This mechanism for redundancy is derived from the non-functional requirement related with availability.

The Hipotalamo and Matrix-class servers hold the SMX4 running instances, where all the bundles are deployed. In the Hipotalamo-class servers, the message broker Apache ActiveMQ takes on the responsibilities to handle the communication with the real-time PDA barcode readers and with other client applications. In order to fulfill the non-functional requirement for consistency, Apache ActiveMQ together with Apache Camel provide Idempotent Consumers to proper handle duplicate messages and transactions to guarantee consistency. ActiveMQ is also used to immediately reply to clients, and asynchronously routing and processing the messages (based on the contents of the messages).

ActiveMQ, for availability purposes, is also used with a master-slave architecture, inside Hipotalamo, with reliable message queues stored in a duplicated network-attach storage (NAS) persistence scheme.
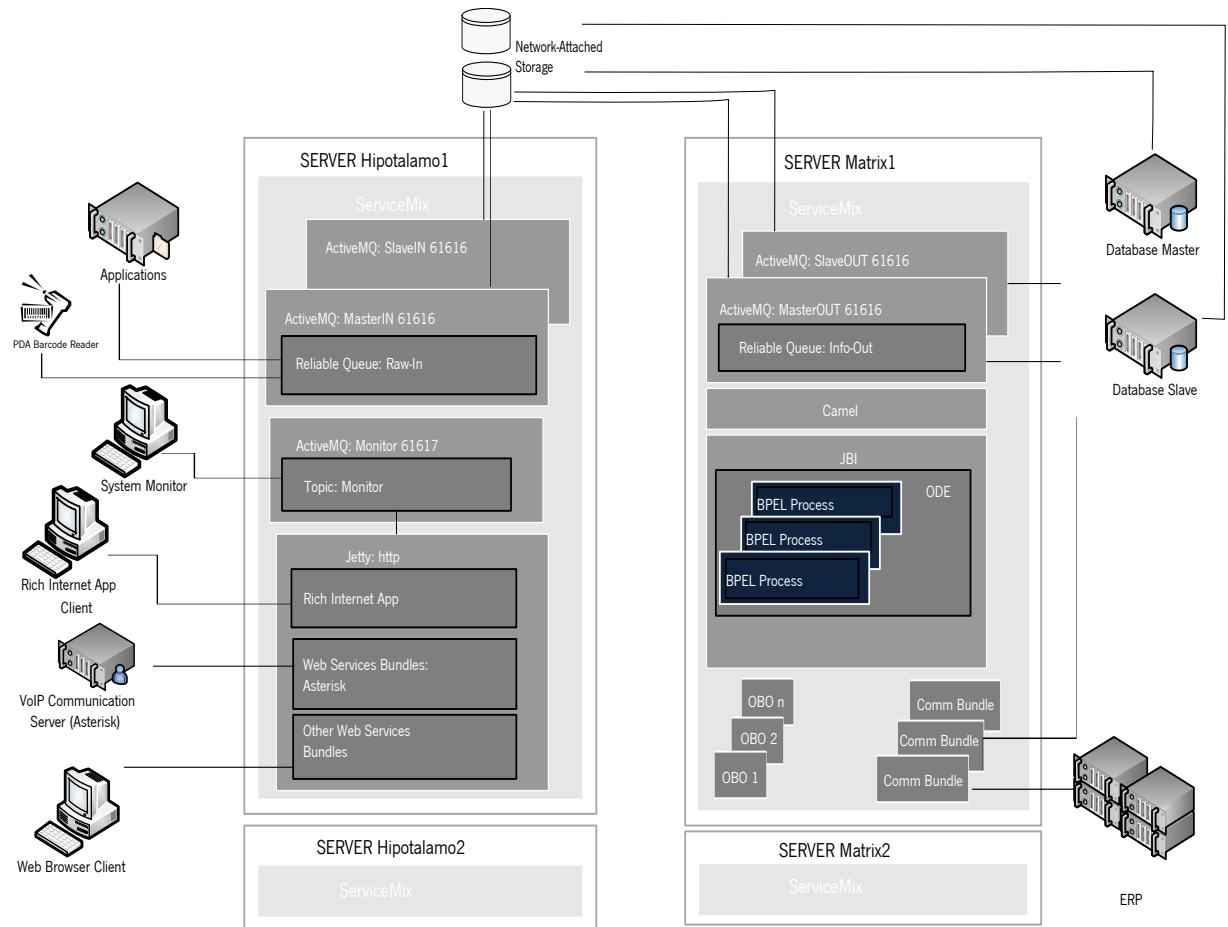

Figure 6.6.: Seppo: a SMX4 and ODE-based software architecture.

The web support can be implemented by recurring to an Eclipse Jetty[4] http server, for the hosting of the server functions of the rich internet applications, to the web services to allow VoIP integration, or to other web services. The monitoring of all the running software is accomplished via the subscription of Advisory Topics in ActiveMQ, alert rules to trigger escalation procedures, and scripts for fast and secure boot-up sequences.

The Matrix-class servers host the reliable message queues to communicate with the database servers. These servers also host Camel and the ODE BPEL container, where the business process orchestrations are deployed. Inside the SMX contained in the Matrix-class server, the OBOs are deployed as OSGi bundles.

Since the architecture of Figure 6.6 is designed for a MES, it is commonly necessary interfaces to an ERP, materialized on specialized communication OSGi bundles.

---

[4]http://www.eclipse.org/jetty/

The storage of the database servers is also placed on the redundant NAS servers due to reliability requirements.

Mapping the D1.8 Activities into the Seppo architecture   The mapping of the activities and actions representing the D1.8 and the OA 4 Eyes principle (see Figure 6.4) is done recurring to the Table 6.10. The D1.8 orchestration is done by recurring to a BPEL process, Process_D1.8, deployed into ODE, inside SMX4, running in the Matrix-class servers.

The action triggering the 4 Eyes OA, Request 4 Eyes Validation, is implemented with a Camel route intercepting the normal flow of D1.8, after the Receive Container with Products action is finished.

Each of the remaining actions in Figure 6.4 are implemented as OSGi bundles, running in SMX4, in the Matrix-class servers. Each implementation represents an OBO. OBOs are not attached to a particular process, and, for that, they are only denominated after the actions they implement, allowing an easier reuse in distinct processes.

Table 6.10.: Mapping the tailored D1.8 into the Seppo components.

Scope	Diagram Element	Action	Implementation Type	Server	Seppo Component	Name of Implemented Component
D1.8	Activity	Receive Product from Source or Make	BPEL process	Matrix	ODE in SMX4	Process_D1.8
	Action	Receive Container with Products	OSGi bundle	Matrix	SMX4	OBO_Receive_Container_with_ Products
		Request 4 Eyes Validation	Camel route	Matrix	SMX4	Route_OA_4_Eyes
		Record Product Receipt	OSGi bundle	Matrix	SMX4	OBO_Record_Product_Receipt
		Determine Put-away Location	OSGi bundle	Matrix	SMX4	OBO_Determine_Put-away_Location
		Putting Away	OSGi bundle	Matrix	SMX4	OBO_Putting_Away
		Record Warehouse Location	OSGi bundle	Matrix	SMX4	OBO_Record_Warehouse_ Location
4 Eyes	Activity	4 Eyes Principle	OSGi bundle	Matrix	SMX4	OA_4_Eyes
	Action	Initiate 4 Eyes Validation	Camel route	Matrix	Camel in SMX4	Route_OA_4_Eyes
		Read Quantity and Type				
		Validate Quantity and Type		(included in OA_4_Eyes)		
		Record Divergences				
		Inform Supervisor				

The OA (4 Eyes), represented by an activity diagram in Figure 6.4, is implemented as an OSGi bundle running also in the SMX4 inside the Matrix-class servers. In this implementation, the Initiate 4 Eyes Validation action is implemented via the Camel route Route_OA_4_Eyes, and all the remaining actions inside the activity OA_4_Eyes are implemented in a single OSGi bundle, to express the oneness of each

OA. Nevertheless, if convenient, the Seppo architecture allows that each one of the actions inside an OA can be implemented in a decoupled way (e.g. for Seppo, we could create detailed OSGi bundles, representing the actions inside the activity 4 Eyes, routed by Camel).

Figure 6.7 depicts the realization of the conceptual Seppo software architecture, presented in Figure 6.6, tailored by the requirements of D1.8 in Centauro 2.0. The shop-floor and mobile clients connect to the Hipotalamo-class servers via the client of the ActiveMQ message broker. We use the .net framework to support local computing on the mobile clients. Due to the importance of the messaging system in Centauro 2.0, the ActiveMQ instances in the Hipotalamo-class servers are monitored via ActiveMQ advisory topics. Additional monitoring for Hipotalamo and Matrix-classes servers, as well as for the NAS file servers and the database servers, is done via the Nagios monitoring framework.

Rich Internet Applications (RIA) connect to the BlazeDS server instance in the Hipotalamo-class servers. From BlazeDS, the requests to the Matrix-hosted functionalities are served via an ActiveMQ client that connects to ActiveMQ instances in the same Hipotalamo server. All the files to hold the input and output messages handled by ActiveMQ instances, in both Hipotalamo and Matrix servers, are hosted in the NAS, along with the data files containing the MySQL database. In the Matrix-class servers, the messages are also handled via ActiveMQ.

The components listed in the Table 6.10 are deployed as:

- a Camel route to intercept and implement the 4 Eyes OA;
- a BPEL process handled by ODE;
- OBOs implemented as OSGi bundles.

The interfacing of Matrix-class servers with the database-class servers is done via the Hibernate component, the interfacing with SAP ERP via the JCO component, and the interfacing with Asterisk via a own developed component, all of them deployed as OSGi bundles.

6.3.3. Theory Adjustments

Centauro 2.0 Action is an initial step in the integration of business requirements together with software requirements in the same execution environment.

Since in the level four of SCOR, and below, each decompose process element is described with classical hierarchical process decomposition, each OBO, at this level representing a Task, can encapsulate a BPEL process to implement the contained activities. Again, this hierarchical process decomposition can also apply to each OBO at level five, representing an Activity, that can be implemented with a BPEL process incorporating even more detailed Activities of the level six of SCOR.
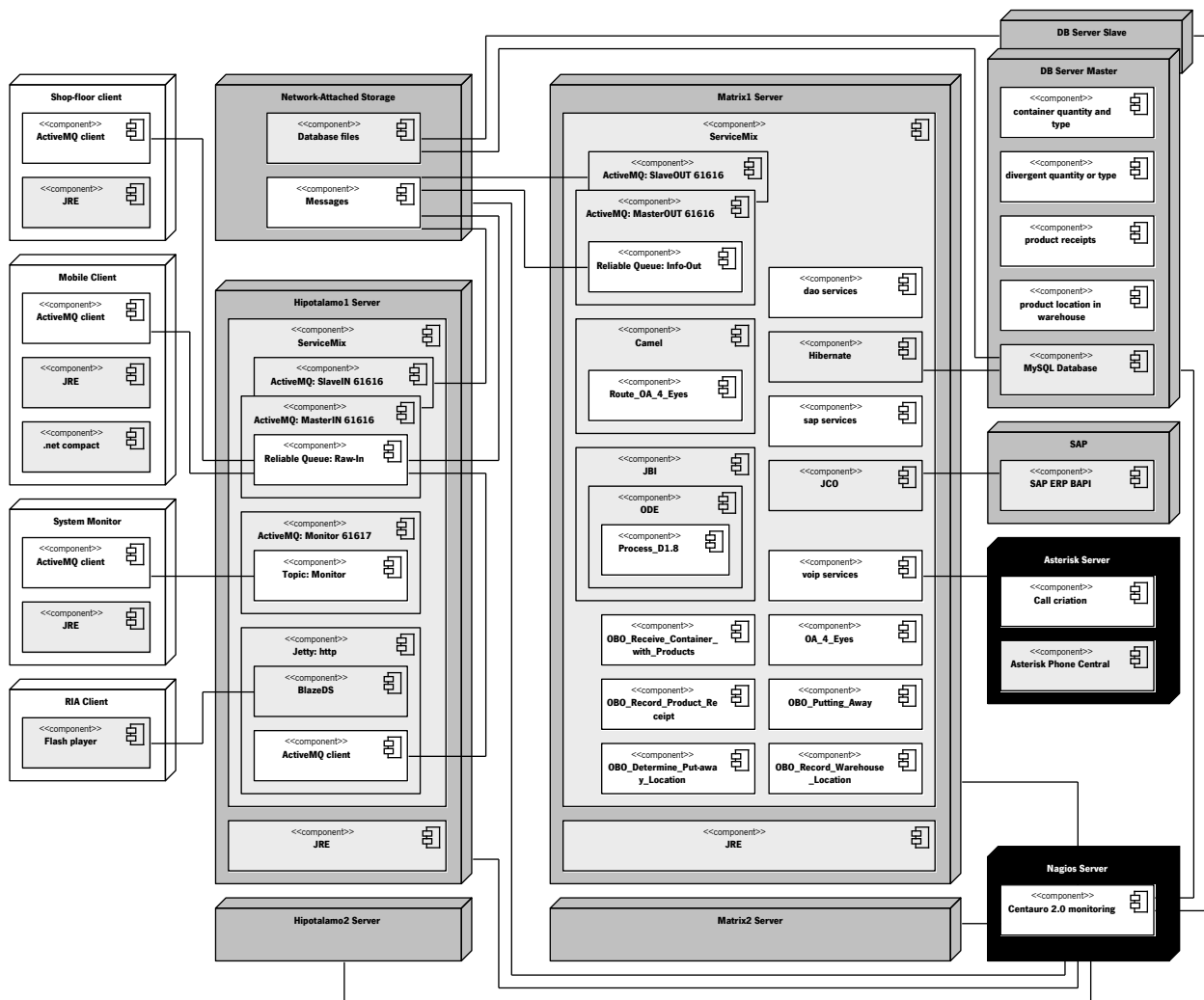
Figure 6.7.: Deployment diagram of Centauro 2.0 in the Seppo architecture for D1.8.

6.3.3.1. Metrics and Lessons Learned

Table 6.11 shows the maturity levels achieved in the Centauro 2.0 according to the proposals of Table 4.3. Comparing with the previous Centauro 1.0 Action (see Table 6.5), the Sound attribute improved, because the BIM together with the techniques supporting technical contents are now available. The maturity level of the Integrated attribute decreased, mainly because Centauro 2.0 recurs to BIM and not to F3. The explicit use of SCOR is accounted positively for integration because SCOR is the basis for Bosch proprietary BP-RM. Previously to Centauro 2.0, Bosch CMP presented to project teams all business requirements without a deep check against SCOR. The Implemented attribute scores a low value, as Centauro 2.0 did not have a go-live in Bosch CMP, despite a prototype was created. The score of the Systematic attribute increased, as BIM defines processes for both business and software contents.

The Assessment and Review attributes keep the same values as the ones from the previous Actions, because Centauro 2.0 is subjected to the same measurement, learning, and improvement cycles.

Table 6.11.: Maturity levels of the attributes of the Enabler in the Centauro 2.0 Action.

Element	Attributes to Score	Maturity Level
Approach	Sound	4
Approach	Integrated	3
Deployment	Implemented	1
Deployment	Systematic	4
Assessment and Review	Measurement	3
Assessment and Review	Learning	4
Assessment and Review	Improvement	4

Table 6.12.: Results of the Centauro 2.0 Action.

#	Dimension of Analysis	Metric	Value
D1	Requirements	Functional Requirements	1/11 = 9%
D2	Requirements	Non-Functional Requirements	8/8 = 100%; 4/4 = 100%
D3	Quality	Quality Assessments	not evaluated
D4	Cost	Effort	not evaluated
D5	Time	Duration	not evaluated
P1	Complexity	Physical Lines of Source Code	not evaluated
P2	Complexity	Used Software Frameworks	>15
P3	Soundness	Change Requests	not evaluated
U1	Reliability	MTTR	not evaluated
U2	Reliability	MTBF	not evaluated

Since Centauro 2.0 did not result in a running software system inside Bosch CMP, the majority of the results showed in Table 6.12 are not evaluated. Exceptions are:

- the functional requirement "D1.8 Receive Product From Source or Make", corresponding to "Book Semi-product at Warehouse" requirement of Centauro 1.0. The remaining 10 functional requirements are not implemented, thus resulting in only 9% of implemented functional requirements;

- the same eight non-functional requirements are implemented together with the four non-functional requirements corresponding to the Seppo framework, resulting in a score of 100%;

- the 15 used software frameworks are, the ActiveMQ, Camel, ODE, Jetty, Asterisk, SAP, .net compact, flash player, BlazeDS, Hibernate, JCO, ServiceMix, MySQL, JRE, and Nagios. The notation of ">15" refers to the bundles used inside ServiceMix (e.g. related to JBI, NMR, OSGi, or CXF), which could also be considered as software frameworks.

Centauro 2.0 proposes new concepts (see Section 6.3.2.2), which are not yet in a final maturity status. For that, in addition to the evaluation scheme used in all previous Actions, expressed for Centauro 2.0 in Tables 6.11 and 6.12, we also evaluate the maturity level of each of the proposed concepts in Table 6.13, recurring also to the scoring scheme proposed in Table 4.3.

6. Running Business Process Reference Models in Software Frameworks

Despite in Centauro 2.0 we did not accomplish to run BIM in its full extent, because at certain point in time Bosch CMP decided to not have a go-live for the resulting software system, there is evidence of BIM usage and, thus, the corresponding evaluation of 2 in the maturity status. For the time being, IAvOs are still in a conceptual phase, since there is no concrete IAvO created. The IAvO concept was not explored in the Centauro 2.0 Action because Bosch CMP is an existing organization and it is not under any reengineering effort. Nevertheless, some evidence exists for the components of future IAvOs, for instance a concrete set of business processes defined in detail deriving from a BP-RM (like D1.8 of SCOR), resulting in a maturity status of 1. The remaining three concepts, OA, PF, OBO, have a clear evidence (3) evaluation on their status expressed by the artifacts generated in the Centauro 2.0, namely Camel routes, Seppo architecture, and OSGi bundles.

Table 6.13.: Maturity status of the proposed concepts after the Centauro 2.0 Action.

Concept	Maturity Status
BIM	2
IAvO	1
OA	3
PF	3
OBO	3

The lessons learned for Centauro 2.0 are:

1. the use of BP-RM is an advantage for the requirements definition because proper, and world-class, business content is immediately available avoiding loops in the collection of requirements, improper or incomplete definitions of business processes, and saving time in the project execution;

2. BIM guides the business tailoring activities, by incorporating and tailoring distinct BP-RM, and thus helping on the creation of a business process landscape suited for a particular organization;

3. BIM creates the support to pick models of business processes and map them into orchestrated software components, allowing easy traceability and reengineering between the business domain and the software domain;

4. business experts can use the graphical representations of BPEL processes for the orchestration of the content of the tailored business processes. Since the main effort to tailor business processes is handled at a lower abstraction level than those provided by BPEL (i.e. in Centauro 2.0, contained in OBOs), the remaining tailoring task of the business processes is the orchestration of OBOs. For that, BPEL graphical representations are understandable by business experts, allowing the same BPEL model to be used during all the life-cycle of BIM;

5. the Seppo architecture provides a framework for process-oriented organizations to implement business processes in software, which also includes support for relevant non-functional requirements;

6. the concept of OBOs allows organizations to develop their own business content, yet maintaining compatibility with higher level business processes;

7. OAs allow organizations to address traversal concerns, easily added or removed, to the execution of their business processes. For instance, in Centauro 2.0, it is only necessary to delete the Camel route Route_OA_4_Eyes to remove the business concern of the 4 Eyes principle for the D1.8 business process;

8. Centauro 2.0 showed that organizations can create implementations of business processes in software in an integrated and understandable way for business experts. Despite technologically feasible, the main roadblocks are linked with the capacity of the organization to understand the benefits of using BP-RMs in a comprehensive way.

6.3.3.2. Proposed Adjustments

Despite the fact that Centauro 2.0 is the last Action of our Action Research study, if someone wishes to prosecute with the Action Research study, the main proposed adjustments are a more detailed specification of BIM and the use of additional BP-RMs other than SCOR.

6.3.4. Social Environment

The project team did not fulfill the project plan for Centauro 2.0 project, as initially expected, mainly because, during the project time the author was invited to move to the quality department of Bosch CMP, embracing tasks not related with the software development. Together with the change of the author, an additional change in the management of the SMT production induced a lower priority for the business processes mapped in Centauro, thus leading to the cancellation of Centauro 2.0 project at Bosch CMP. For those reasons, some of the presented artifacts were finished during project time (e.g. the Seppo software architecture), others were partially completed (e.g. BIM), and others (e.g. the implementation of the software architecture) were finished in a prototyping environment, after the cancellation of the project.

6.4. Conclusions

Table 6.14 shows the evaluation of the status achieved at the end of the Centauro 1.0 and Centauro 2.0 Actions, compared with the goals of the thesis. At the end of Centauro 2.0 Action, the main goal (MG) of the thesis is achieved. The quality of the resulting software (SG1) was not evaluated because Centauro 2.0 software system was not yet used in a productive environment, the time needed to execute BIM (SG2) is only partially evaluated, with positive results, by comparing the implementation time needed to complete the same functional requirement in Centauro 1.0 with the time needed to orchestrate the

OBOs for D1.8. All the remaining secondary goals are achieved.

Table 6.14.: Evaluation of the stopping conditions for Action Research related with the Centauro 1.0 and Centauro 2.0.

Goal Id	Goal Description	Centauro 1.0	Centauro 2.0
MG	To propose a methodology to generate immediately executable information and software systems for process-oriented organizations, starting with models of business requirements	Not Achieved	OK
SG1	The resulting software products must have quality	OK	Not evaluated
SG2	The proposed methodology should be able to be executed in a short period of time	Not Achieved	Partially
SG3	To specify and to implement a software framework to support software developed with the methodology	Partially	OK
SG4	The resulting methodologies and software products should be able to be used by business experts without intervention of software experts	Not Achieved	OK
SG5	The proposed methodology should be holistic, ranging from business processes descriptions to running software	Partially	OK
SG6	To promote the composition of software solutions based on free software and originated by different vendors	OK	OK

7. The Business Implementation Methodology

This chapter presents a methodology that guides the implementation of business processes in software systems. In order to provide guidance during all the life-cycle of the creation of software systems, BIM starts with business processes models and ends up in a running software implementation supporting those business processes.

We also present some technologies that support BIM during project time and operation time, based on OSS frameworks.

7.1. Introduction

The Business Implementation Methodology (BIM) is a guiding methodology to obtain a software system that supports the execution of the business processes existing within an organization. The methodology promotes the use of BP-RMs and intends to reduce the development time. BIM has four distinct phases and several abstraction levels. BIM is applicable both when developing systems from scratch or in reengineering contexts. BIM embodies a special phase to handle the variability of the business processes inside an organization. BIM initiates by tailoring BP-RMs, IAvOs, and OAs, and by considering the characteristics of a specific organization, thus allowing the proper set of business processes to be derived for that organization. Then, a suitable information system can be obtained and its automatable parts can be implemented in a software solution that can run, for instance, on top of OSS frameworks.

The primary concern of BIM is to guide the development of business software, ensuring the adequate support for the business processes of an organization. For that, the main objectives of BIM are:

- to provide a holistic methodology to guide the implementation in software of the business processes of a process-oriented organization, following state of the art BP-RMs;
- to support different business process models and notations;
- to cope with mixed vendor software technologies to support the operation of business processes.

Thoroughly used during all BIM phases is the concept of BP-RM. This concept arose from the idea of sharing best practices among organizations. BP-RMs should represent the business operations and the internal structures of organizations. BP-RMs can be understood as templates from which process models may be developed (Snabe et al., 2008). Consequently, if standard processes are to be used, standard roles to intervene with those processes can also be depicted in advance.

Currently, generic BP-RMs are available for organizations, like the SCOR (Council, 2008a) or the ITIL (OGC, 2008) for horizontal business functions. Other BP-RMs are also available for vertical markets, like the eTOM (Kelly, 2003) for the telecommunications industry or the Fernandes/Duarte reference model (Fernandes & Duarte, 2005) for software development organizations. BP-RMs are standards accepted by organizations on how business processes should be properly designed and used. BP-RMs are usually accompanied with recommendations, training events, books, and certifications, all providing a set of tools for the implementation teams.

Implementing BP-RMs inside organizations is a task that heavily relies on the human skills of the elements of the implementing team, and it is not immune to project environmental conditions, like a low budget for training events.

Currently, after an organization adopts a new set of business processes, comes the need to implement most of the business processes, quickly and with quality, in some supporting business software system.

Considering the problems that methodologies to implement business processes have (stated in the conclusions of Chapter 3), a methodology to develop software systems supporting a set of business processes should deal with the following challenges:

- how to use BP-RMs;
- how to integrate tailored business processes, as required by the client organization;
- how to promote the automation of the techniques used in the methodology;
- how to obtain a quality software solution, in a fast and cost-saving ways.

The BIM intends to address the previous presented challenges, in addition to handle properties of process modeling, like automated execution or process management, as suggested by Curtis et al. (1992). BIM wants to promote the use of state of the art software frameworks.

The opportunity to create a running software system in an automatized way improves the quality of the resulting software product, by eliminating errors during the manual model transformations and by letting the developers focus on the content of adequate business processes and on the software environment, instead of paying attention to the correctness of the transformations of the models. Automation also induces a reduction in the development time.

We propose the concepts of OBO, IAvO, OA, and PF, to allow an easier execution of development projects for business software, but the concepts are not mandatory for a BIM execution. The primary purpose of the concepts is to ease the tasks of the business experts and the software engineers by providing guidance and support to their activities.

BIM can still be used without recurring to any of the four proposed concepts. Nevertheless, without IAvOs, the project team does not make use of a suitable and fully designed organization defined prior to project time. Without OAs, the traversal concerns of an organization will be scattered in the business processes, as well as in their software implementations. Without the PF, all the knowledge about business processes will be spread across distinct and disconnected models. Without OBOs, the client organization misses the opportunity to have semantically sound software components compliant with the business processes, and, after go-live, loses the ability to interchange seamlessly a software component with another semantically compliant component (e.g. due to the low performance of the substituted component).

The BIM aims to provide guidance for the migration from the business processes domain into the software domain.

From the perspective of a business organization, a methodology for the development of business software is more useful if the methodology provides guidance and shows the best practices for the business context, rather than being agnostic on such aspects.

Later, on Section 7.3, we depict some software solutions and technologies that can be used during the operation time of the software-implemented business processes. Also in this situation, BIM is not attached to any particular software architecture. Nevertheless, BIM promotes the use of OSS in order to provide more flexibility to business organizations when choosing their software landscapes, untying organizations from proprietary software solutions, and allowing the composition of a software landscape based on mixed-vendor software solutions.

Desirably, business software solutions should not be a constraint to an organization. This purpose can be achieved if the best business processes can be designed and the best software implementations can be used by each organization, and not only the business processes and the software solutions provided by a particular software vendor.

7.2. The Methodology

The BIM starts with generic BP-RMs and ends up with running software. BIM is not attached to any proprietary software platform neither to any specific notation for business processes or other process artifacts. Nevertheless, for the sake of reducing the project time, BIM encourages the use of executable business process languages and techniques to increase the degree of automation in model transformations (e.g. 4SRS). BIM also provides activities to choose, to use, and to tailor BP-RMs to a specific organization.

The BIM is built on the premise that the quality of a business software system improves if, at the early phases of a project, good models of business processes are designed and presented to the software development team.
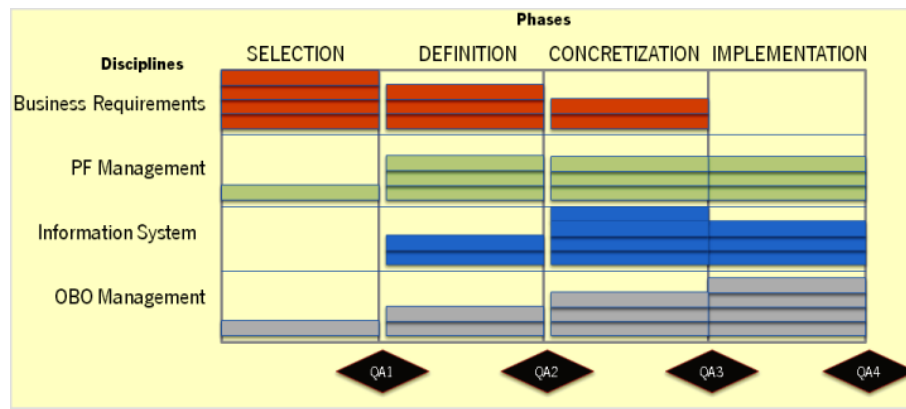
Figure 7.1.: BIM overview.

Figure 7.1 presents the overview of BIM. BIM has four phases: Selection, Definition, Concretization, and Implementation. BIM proposes four disciplines:

- Business Requirements, related with the collection and the governance of the business requirements expressed in BP-RMs, and with the internal business concerns of the client organization;

- PF Management, where all activities concerned with the PF are grouped, including those when the PF is in the software-implemented state;

- Information System, for activities involving the creation and maintenance of both the software-implemented and the manual parts of the business processes. The management of alternative designs for the same business process is also considered, allowing an organization to incorporate variability into the business processes;

- OBO Management, where these software components are developed, customized, and deployed into the running software system.

Figure 7.1 also depicts the estimated effort to perform each discipline, in each phase of BIM. The starting effort for Business Requirements discipline is high and decreases as the project time elapses, until no effort, or only residual, is needed during the Implementation phase. The PF Management effort is low during the initial phase and then it stays stable for the remaining BIM phases, since this discipline is highly related with the PF artifact and this artifact is crossing all the life-cycle of BIM. The depicted effort for the OBO Management discipline represents an ideal situation. If the available quantity of OBOs to implement the defined business processes is low, then the effort related with OBO Management discipline should be higher at the early phases.

7.2.1. Formalization

We formalized BIM recurring to the Eclipse process framework (EPF) (EPF, 2011). Appendix BIM Formalization in EPF includes the detailed views of the formalization.

BIM is composed of four discrete time consecutive phases (Figure 7.2). During the first BIM phase, a Selection of BP-RMs, OAs, and IAvOs is made, followed by the Definition of the chosen business processes to the project. Next, follows the Concretization of the business processes in an information system. At this phase, there is yet no software implementation of the business processes, and the organization may have alternative designs for the same business processes. In the last BIM phase, one, or more, software implementations are created for a subset of business processes concretized in the information system. The need to have different software implementations comes, for instance, from an organization relying on an ERP which has annual upgrades. During those upgrade periods, the organization can benefit from having alternative business process and/or software systems to overcome the inactivity of the ERP.
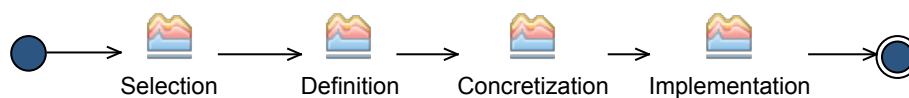
Figure 7.2.: BIM delivery process.

Each BIM phase ends with a quality assessment (QA) milestone. Each QA evaluates the maturity of the project at that time, via a check on the planned deliverables. The QA produces a statement expressing if the project is ready to move into the next phase (or, in the case of the last QA, if the project is ready to terminate). Each BIM phase follows the pattern implemented for the Selection phase (Figure 7.3). Each phase is composed of one or more iterations followed by a QA. While the expected quality and maturity levels of the artifacts, deliverables, and outcomes are not reached, or when some project change or constraint occur, iterations can be run incrementally inside the same BIM phase.
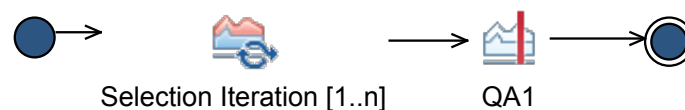
Figure 7.3.: BIM Selection phase.

During project time, the PF passes through the states shown in Figure 7.4. Starting with generic BP-RMs and IAvOs, an Instantiated PF is then obtained. The business processes can be described in BPEL, but also recurring to other languages, like BPMN, UML, EPC, or CPN. The preference for BPEL comes from its intrinsic properties, compared with the properties of the other languages, and mainly with the easiness to read it and at the same time to execute it (Santos et al., 2013).

The Instantiated PF should be compliant with the mission, the vision, and the strategic objectives of the organization, which it refers, as well as with the other business constraints. At this PF state, the description of the process-oriented organization is achieved, being the basis to derive the information system model. Afterward, when the PF is concretized into the information system, the PF is in the Runnable state because its processes, probably not all, may be implemented in software.

7. The Business Implementation Methodology

Table 7.1.: BIM summary.

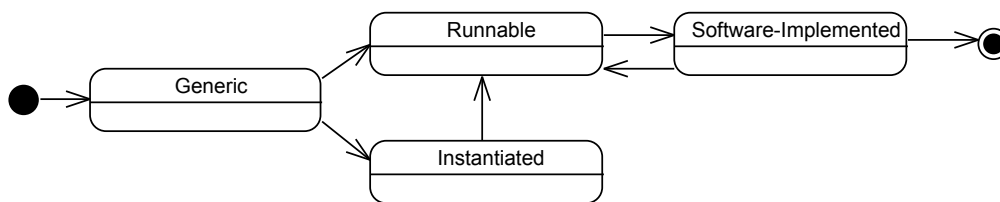| BIM Phase	Metamodel	Domain	PF State
1. Selection	BPEL, BPMN, EPC, CPN, ...	Business Requirements	Generic
2. Definition	BPEL, BPMN, EPC, CPN, ...	Business Requirements	Instantiated
3. Concretization	BPEL, BPMN, EPC, CPN, ...	Information System	Runnable
4. Implementation	Computational Model	Execution	Software-implemented

Figure 7.4.: PF states.

The bidirectional state transitions between the Runnable and Software-implemented states are justified by the loading of the software system with a subset of the information system processes and by the "unloading" of some of those processes, becoming processes in a "vegetative" Runnable state. Ideally, the software implementation should be accomplished just by parametrization, without the need to write new code. There is also the possibility that some Generic PF can directly be moved into the Runnable PF state. This situation occurs when the organization does not want to make any customization of a generic BP-RM, or when the organization wants to adopt fully some IAvO. Desirably, the Runnable and the Software-implemented states should represent the same amount of business content.

The BIM, generically summarized in Table 7.1, during its first three phases supports models of business processes. Because a software implementation is in its scope, during the last phase of BIM there is a need for a computational model. In the Selection and Definition phases, activities are occurring in the domain of business requirements. During the Concretization phase, an information system for the organization is designed including both manual and software-implemented business processes. The PF states are directly related with the BIM phases.

7.2.2. Activities Inside the Phases

Inside each BIM phase, the process roles perform process-related functions. UML actors and high-level use cases are used to explicit that collaborations.

The activities for the Selection phase are presented in Figure 7.5. When executing the Selection phase,

the existing BP-RMs, IAvOs, and OAs should be analyzed in order to devise and pick useful business content for the organization, like in a benchmarking activity. All these three analytical activities can be performed in parallel. When they are completed, the selection of a generic PF is finished.
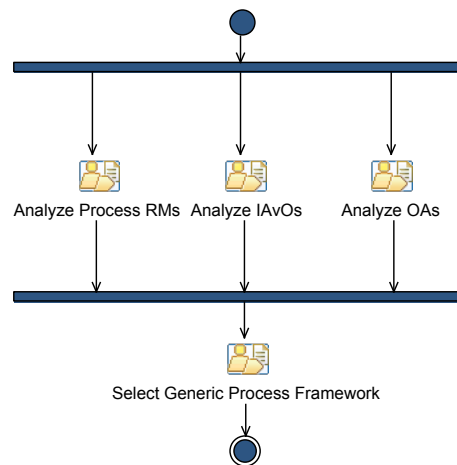

Figure 7.5.: Activities in the Selection phase.

The process role Business Analyst (see Figure 7.6) chooses from the market an adequate BP-RM (e.g. SCOR). He should consider if there is an IAvO that addresses the business processes covered by the project. The Business Analyst and the Client define which OAs can be included for the PF. At the end of the BIM Selection phase, the first Quality Assessment (QA 1), as depicted in Figure 7.3, should be performed.
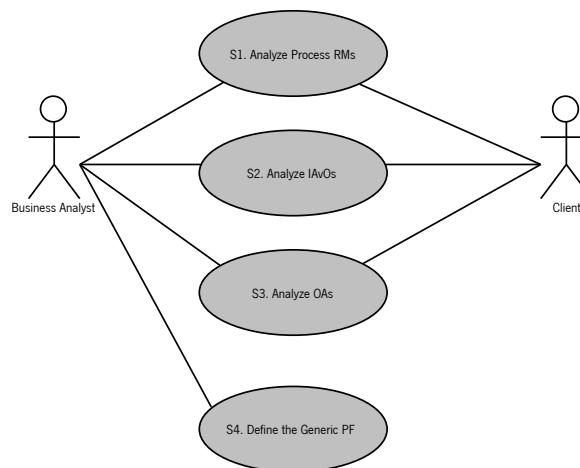

Figure 7.6.: Use cases during the Selection phase.

The most important deliverable generated during the Selection phase is the choice of a mature and excellent PF, ready for use in the next phase. The Business Analyst is responsible for the quality of this artifact.

Besides the intrinsic business quality of the PF, its representation must also be chosen. Using busi-

ness process languages, mixing them with informal graphical notation, or even using some textual descriptions, can be accepted if the resulting notation brings clarity to the model of the Generic PF.

The Generic PF selected in the Selection phase is used to reach an Instantiated PF during the Definition phase. The Instantiated PF should include the client's view and the future strategy for his own organization. Business-driven development (Mitra, 2005) can be used to create a proper alignment of the vision of the organization with IT tools.

One key aspect of the Definition phase (see Figure 7.7) is the explicit utilization of an immaterial process role represented by the Generic PF. We propose that even when the Client or the Business Process Architect do not explicitly consider world-class business processes, and for the sake of the client organization, the project team should consider the Generic PF role (see Figure 7.8) demands in order to derive a proper Instantiated PF.
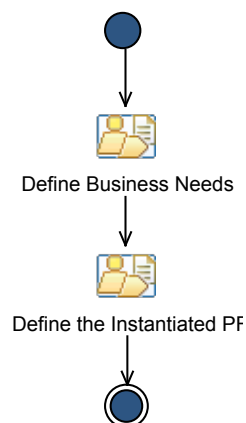

Figure 7.7.: Activities in the Definition phase.

The Client is not always able to decide what is the best solution for his own organization. This situation can be triggered by the ignorance of best-practices outside his own business environment or by human representatives of the Client that do not always put the organization's best interests on top of their own personal interests.

This immaterial process actor should provide a kind of resilient implementation of the values of the organization for the project team to consider.
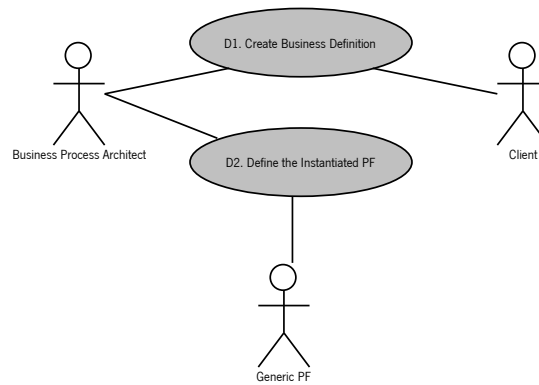
Figure 7.8.: Use cases during the Definition phase.

After a BIM project, the complete information system, consisting of both software-implemented and manual processes, needs regular audits. The PF in its Runnable state is vulnerable to end users that do not want to use the business processes of the organization, normally expressing it by creating personal worksheets and databases.

Thus, we propose a quantitative audit to be performed periodically (e.g. yearly), in order to check:

- the compliance of the current Software-implemented PF with respect to the Instantiated PF designed during the BIM project. This check has as precondition that the same business environment applies, otherwise a new run of BIM is needed. This activity should prevent changes in the Software-implemented PF without an explicit validation;

- if the manual processes are still executed the way they were planned, avoiding again changes that may endanger the business goals.

Managers should not confuse undesired business process changes, like the ones just presented, with the ability to cope with continuous improvement activities. Whilst the former may turn the result of an activity against their own host organization (e.g. the optimization of a production process that leads to an increased product stock without a proper planning of the available physical space), the latter are mandatory in any organization that seeks business excellence.

The goal of the third BIM phase, Concretization (see Figure 7.9), is the design of the Runnable PF, based on the Instantiated PF obtained in the Definition phase.
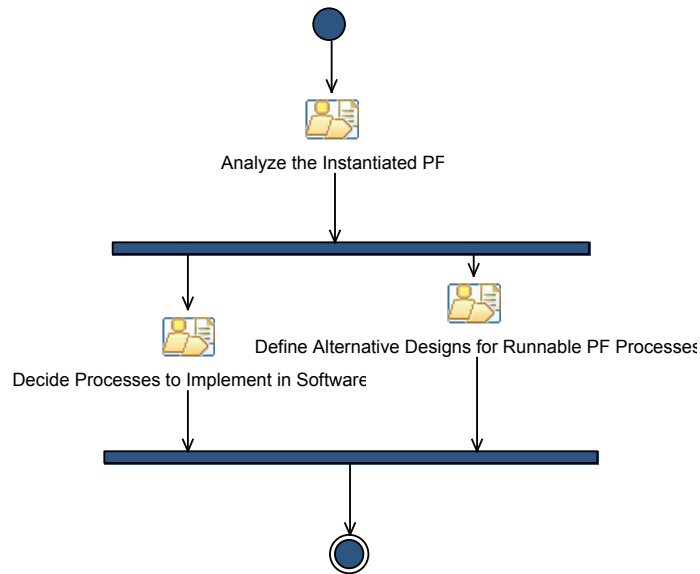
Figure 7.9.: Activities in the Concretization phase.

The Business Process Architect role (see Figure 7.10) should includes into the Runnable PF the alternative designs for the same business processes, as well as the decisions about which business processes will have a software implementation.
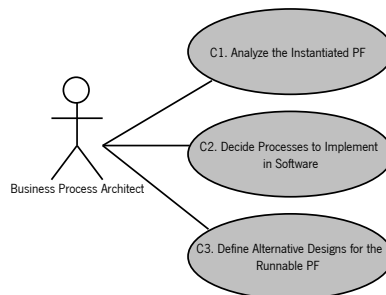

Figure 7.10.: Use cases during the Concretization phase.

Finally, during the Implementation phase (Figure 7.11), the Runnable PF is implemented in software. At this point, two distinct approaches can be taken to solve the problem of generating a software solution. The first is to generate code starting with previous defined models. The second is to use already developed code, customized to cope with the requirements. With the latter approach, mixed vendor pre-developed software can be used, if compliant with the Runnable PF requirements.

In this scenario, the client would not be tied to a particular COTS software vendor. Neither he would have to rely on web services for which a proper service level agreement is unfeasible. We believe that having mixed vendor software to support business operations inside an organization brings benefits
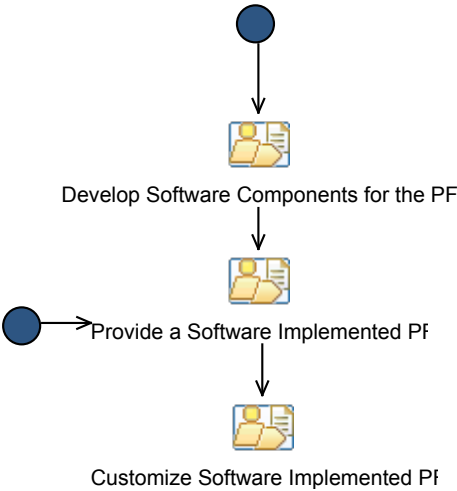
Figure 7.11.: Activities in the Implementation phase.

for the quality of the implemented solution, because the client can have the ability to load and unload software implementations for a part or for a complete business process.

During the BIM Implementation phase, the designer of the behavior for the Software-implemented PF is the Business Process Architect and not the Software Engineer (see Figure 7.12). The latter role should only be concerned with providing a running software infrastructure for business processes to run. The existence of pre-developed components (OBOs) indicates that additional (outside the BIM project) software development activities may be performed under the responsibility of the Software Engineer.
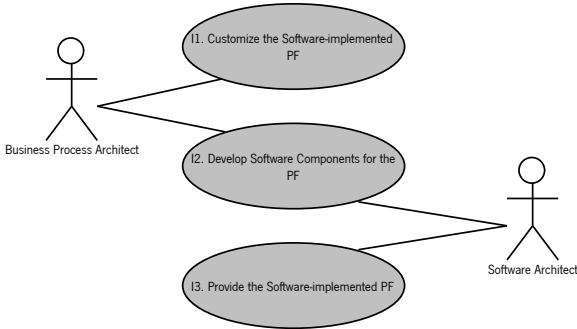

Figure 7.12.: Use cases during the Implementation phase.

The Software-implemented PF is customizable and is the cornerstone to compare the complete OBO functionality and the process orchestrations with the requirements for the Runnable PF. The conflicts between the Software-implemented PF and the Runnable PF can be detected, when the latter requires more functionalities than the ones available in the OBOs. This conflict may demand the acquisition or the development of more OBOs, or the extension of the existing ones.

191

7.3. Supporting Technology

BIM projects may benefit, in time reduction and in improvements on the resulting quality of the software product, from having a proper technological support during the development time and during the operation time.

The technological support for the development time is related with the technologies and tools that manipulate IAvOs, OAs, and the PF, in the latter mainly by the ability to visualize the business processes.

The technological support for operation time is more related with the composition of a proper software environment for OBOs and for the software execution of the PF.

For operation time, the current software development methods and software products may not be suitable for current business needs, as explained in Section 3.4. Despite the very important role that the software development processes and methodologies have on the resulting quality of a software product, the higher contributors are the quality of the components and of the integration achieved in the software product. The use of pre-built software components is an approach that allows components to be tested and improved before their use in a project. For that, if pre-built components are properly used and integrated, they are a decisive factor to improve the quality of a software product.

7.3.1. Technology for Development Time

Despite models of business process can be expressed in a variety of languages, due to their complexity, size, and different types of users it is advisable to represent them in easily understandable graphical representations, possibly complemented with textual descriptions. However, business experts have difficulties to understand some graphical languages, like the CPNs, making their use hard. Easily understandable graphical languages facilitate the activities of business process experts and, thus, they trigger better results for the activities of the business experts. For that, the PF in all of its states (see Figure 7.4) should be visualized recurring to a tool in order to allow business processes experts, even without software engineering skills, to use it. Currently in BIM, the recommendation for the language to describe and execute business processes is the BPEL. We use the Eclipse BPEL designer (see Appendix D) as the tool for editing the PF. Nevertheless, Eclipse BPEL Designer was conceived mainly to design and deploy BPEL processes, and does not cover all the required functionalities (see Section 6.3.2) expected from a PF managing tool. For instance, the PF state must be manually managed. A complete tool to visualize and manage the PF during the development time, should expose the following functionality:

- graphically creating and editing the business processes;
- creating and editing OBOs orchestrations;
- creating and editing the configuration files for the Software-implemented PF state;
- translating OBOs, and their orchestrations, into a graphical business processes representation;

- importing and instantiating generic BP-RMs;

- introducing OAs into the PF, possibly via Apache Camel routes;

- managing the variability of OBOs inside the same high-level business process, at the Runnable PF state.

BIM demonstrates that the same model of business processes can be used to describe the PF during the complete life-cycle of a BIM project. When different languages and models are used for the different PF states, model transformations should be performed to handle the creation, or the updating, of objects in the target model based on objects from the original model. Model transformations techniques and languages can also be used to support the synchronization between the original and the target models, providing the basis to convergence between these models at different PF states, like by using the QVT language (OMG, 2011c).

Even so, model transformations caused only by the necessity to translate a model into another model (i.e. a model more easily executed or understood) are not adding value to the amount of represented real world objects inside a project, and thus they are waste and project teams should avoid such transformations.

7.3.2. Technology for Operation Time

After project time, tools are helpful to support the management of the business software during its operations. In BIM, at operation time, the PF will be in the Software-implemented state. The main use of the Software-implemented PF is to provide the foundation for the business software to support business processes, according to the earlier customized PF.

Next, we present some high-level requirements for a supporting tool for the PF. To state the requirements (functional and non-functional) we used the FURPS+ model (Grady, 1992). Thus, at the software-implemented state, the functional requirements for a tool to support the PF at operation time, are:

- loading OBOs into the PF;

- unloading OBOs from the PF;

- activating loaded OBOs;

- deactivating loaded OBOs;

- customizing the restrictions on the usage of the available functionality of OBOs;

- manage the orchestrations of OBOs (e.g. via BPEL);

- manage the OAs (e.g. via Apache Camel routes).

The non-functional requirements are:

- usability: should be usable by business experts with basic IT skills;

- reliability: no special requirement;

- performance: no special requirement;

- scalability: the tool should be able to incorporate new OBOs and orchestrations into the PF. It should also scale to support either the implemented business processes of a small business or those from a large corporation;

- synchronization and model integrity: the tool to manage the PF should always be synchronized with the running software. Thus, any change in the composition of the OBOs landscape, as well as the changes in their orchestrations, should be reflected in the PF. Synchronization should also allows that the changes made in the PF in the previous states to be reflected in the Software-implemented PF. Changes in object orchestrations may have the potential to be immediately applied into the running software, but changes in business processes may demand the creation or acquisition of new OBOs, and thus requiring a higher synchronization time. Users of this supporting PF tool should also be informed about model inconsistencies in order to correct them. The purpose of the synchronization mechanisms is to keep integrity among all models during BIM operation time;

- openness: support for the manipulation of OBOs from different software vendors.

To maintain the consistency among the business models during all the life-cycle of BIM, we suggest the use of Java-like interfaces to describe the behavior of activities contained in the business processes. This way, during all BIM phases, the concerns can be traced across models. Then, when in a software system, the OBOs from different vendors can be interchanged without compromising the business content and the software architecture. This interchangeability based on traversal interfaces can be a solution to create a software architecture compliant with the business processes and, at the same time, to provide a vendor-independent software system.

One of the main characteristics of the Software-implemented PF is its ability to reuse components from distinct vendors. Considering common business software architectures (e.g. for ERPs), the organization that creates the ERP product commonly does not provide all the needed implementation details. Later on, the same organization, or its partners, creates new components to extend the standard ERP functionality based on the ERP core technology. This approach has some advantages, like the consistency of the ERP with the newly developed components, or the speediness when using a standard customization. However, some problems are also triggered by this approach, namely:

- if a client organization wants to benefit from the best components of the commercially available business software, it needs to set up a complex integration project. Such a project will require expertise in different software packages, and the ability to integrate different components;

- if the business process requirements induce some special behavior, not present in the ERP system, then non-standard software must be created;

- during ERP upgrades, non-standard software is always problematic to maintain;

- the client organization, for its own software landscape, is always in a lower control position than the business software vendor. This can be perceived when the support is terminated by the software vendor for some deprecated version of a software product, forcing the client

organization to move into the new software products, most likely with no improvement on the business operations;

- normally, a client organization has to pay for complete ERP licenses and fees but only uses a portion of the complete ERP system functionality.

Thus, a plausible solution for a proper vendor-independent business supporting software is the use of OSS. Currently, many frameworks are available and they are used to support demanding business operations with professional quality level (e.g. Apache ActiveMQ is used in the Large Hadron Collider Project of the CERN to reliably move operational monitoring data between nodes without imposing technology on users (FuseSource, 2012)).

7.3.3. Architectures for the Software-implemented PF

In this section, we present three suggestions for high-level software architectures for the software-implemented PF, all based on free software.

A Three-Tier Architecture Based on the Spring Framework   In the architecture presented in Figure 7.13, the business software system is composed of three tiers:

1. the presentation tier, where the presentation logic, data, and views reside;

2. the business tier, composed of the business logic, the OBOs, and of course, the Software-implemented PF;

3. the persistence tier, including all the data services, as well as an ORM framework.

This architecture can use several layers inside each tier. The complete software stack can be implemented recurring to several customized OSS software frameworks together with own developed software components. An application server hosts the business tier.

The main design and architectural decisions are:

- the user interface is based on web browsers, to avoid developing proprietary GUI clients and to eliminate problematic deployments of new features. Additional technologies can be used to support interaction of other clients to the application server, like RFC;

- the user interfaces are handled with a framework, like the Apache Struts (Roughley, 2007) or the Spring MVC (Johnson et al., 2005), which implement the Model-View-Controller pattern (Krasner & Pope, 1988). Both frameworks allow the existence of user interfaces where the viewing is detached from the control and the data, leading to a better and more reusable software system;

- the Spring framework (Johnson et al., 2005) is used as the business tier main framework. Spring prevents, among other problems, the development of boilerplate code by providing an infrastructure to create connections between Java classes before operation time;

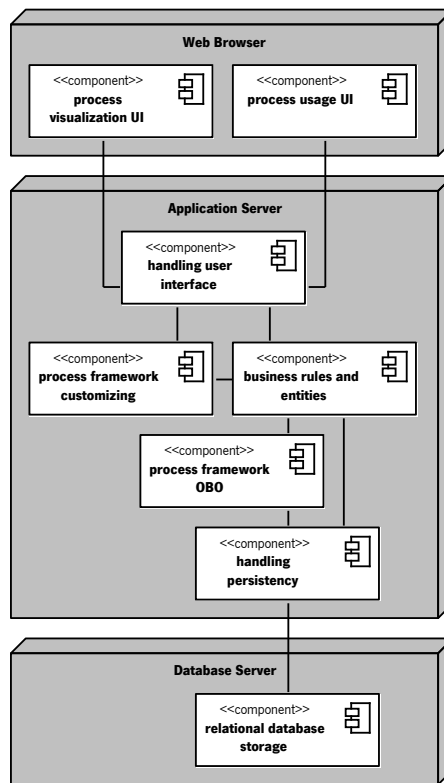- by using Spring, the relations between OBOs can be handled by the configuration files of the

Figure 7.13.: A generic deployment diagram for the Software-implemented PF recurring to the Spring framework.

framework (e.g. the ApplicationContext file), instead of being hard-coded into the OBOs;

- OBOs can be implemented recurring to Spring beans realized as Plain Old Java Objects (POJOs) (Fowler et al., 2000);

- an application server, like the IBM WASCE (IBM, 2009) or the Apache Geronimo[1], is used to host the OBOs and the Software-implemented PF;

- an ORM framework, like Hibernate (Bauer & King, 2006), is used to minimize the used capacity in activities related with the data persistency and to improve the consistency between "live" objects and their associated relational database storage.

In this architecture, whenever an OBO is deployed into the application server, all of its relations must be reconstructed in the ApplicationContext file. With this PF software implementation, model transformations are needed during the BIM project time. These transformations can be automatized with heuristics, like the ones proposed by Fernandes & Machado (2001) or Liang (2003), or with model-driven approaches, like the MDA.

A significant number of business software systems and platforms usually exist within an organization. This diversity requires the integration of different data formats and requires distinct communication protocols. In recent years, several technologies, such as enterprise application integration (EAI),

---

[1]http://geronimo.apache.org/

business-to-business (B2B), SOA, and web services, propose to solve the diversity related problems. These technologies address some of the integration problems, but software solutions that support them tend to be proprietary, expensive, and time-consuming to implement (Community, 2008). As opposite, the ESB approach provides distributed messaging, routing, business process orchestration, transactions, reliability, and security.

Next, we present two proposals for architectures based on different versions of the Apache ServiceMix ESB.

An Architecture Based on a JBI ESB (ServiceMix 3)   A more sophisticated architecture to implement the PF is to use an ESB based on the JBI architecture. In this architecture, we use the SMX3 (see Section 3.2.2). With the use of SMX3, ODE, and BPEL, the need for model transformations in the PF could be eliminated, since the same BPEL model can be used in all of the four BIM phases. Therefore, BPEL can be used to describe the Generic, the Instantiated, the Runnable, and the Software-implemented PFs. During the transitions between BIM phases, SMX3, ODE, and BPEL allow the business semantics to be changed in the states of the PF, keeping unchanged the language syntax. Thus, one can describe, or pick from the repository of the Runnable PF, a business process in BPEL and drop it on the SMX3, allowing the PF to change from the Runnable state to the Software-implemented state.

In this implementation, OBOs are Service Engines (SE) and the OBOs orchestrations are the configurations deployed into the ODE. These two implementations define a new Domain Specific Language (DSL) (van Deursen et al., 2000) in the business domain of the adopted BP-RM. This means that the less abstract parts of the processes of the Generic PF must also exist in the SMX3 as SEs. Since SMX3 is widely accepted, open source, and based on open standards it can provide a quality, flexible, and low cost solution to implement the PF.

This architecture assumes that all the needed OBOs (derived from the DSL of the Generic PF) are implemented as SEs in the SMX3. If this is not the case, then one must acquire the needed OBOs or initiate activities to create them.

An Architecture Based on an OSGi ESB (ServiceMix 4)   This architecture uses the SMX4 (see Section 3.2.2). In this architecture, like in the SMX3-based one, the SMX4, ODE, and BPEL, together overcome the need for model transformations in the PF. This architecture is used on Centauro 2.0 Action (Section 6.3).

7.4. Conclusions

As processes are becoming more and more automated, their management will become automated as well (Snabe et al., 2008). In the near future, organizations must reach an internal level of maturity to

allow focusing on building and improving excellent business processes, on top of focusing only on the daily operational problems related with their processes and their respective software implementations.

This chapter presents a high-level methodology to implement properly a set of business processes in software. At the present, we designed BIM to act as a guide using a phased development process. Also, the explicitly inclusion of new concepts, such as IAvO, OA, PF, and OBO, which are typically not considered when developing business software, can spot the importance and trigger the discussions on their underlying concepts.

IAvOs allow entrepreneurs, namely the inexperienced ones, to start a business with a higher probability of success, and/or to reduce the time needed to set up the internal structure of an organization by defining a core set of business processes and business roles.

OAs can help on configuring business architectures to cope with cross-cutting business concerns, like quality standards, without significant effort.

PFs can be the basis to transpose smoothly requirements and definitions from the business domain into the software domain.

OBOs constitute an important concept, namely because they allow the client organization to use a mixed vendor software solutions to support its business processes. OBOs foster the reuse of software components.

BIM also promotes the use of BP-RMs during projects to implement business processes, resulting on the incorporation of best practices by an organization, and thus improving the business operations.

Organizations must be immune to proud internal actors that do not release the organization from their own ideas, even when other better alternative ideas coming from benchmarked processes are offered. This behavior from internal actors can be a serious obstacle for the improvement of the internal structures of an organization.

BIM is a wide-scope methodology, since it deals with business models and at the same time with software implementations. For that, BIM is proposed currently in a high abstraction level, so that it can consider several concepts from the domains it covers. This way, BIM can remain independent of technologies, notations, and methods.

BIM brings some advantages over current methodologies for business software development:

- BIM has a holistic scope, since it starts from generic BP-RMs and ends with software solutions;
- it avoids model transformations by using the same model for business processes during all the life-cycle;
- it promotes that a software system can include components from distinct software vendors, by using the best OBOs from each vendor to compose a valuable solution for the client organization;
- it can significantly reduce the time to implement a software solution, since the very same model to describe and execute business processes is used along the complete life-cycle of BIM;

- it is adaptable to different BP-RMs and software architectures;

- it addresses both the business and the software domains with a traceable artifact, the PF;

- it explicitly proposes automation techniques.

BIM need further improvements, namely:

- on the supporting tools (e.g. to support the visualization of PF);

- on the guidance to properly tailor the methodology to a specific project taking into consideration the possible methods and techniques that can be used during a BIM project, the skills of the development team, the business context of the organization, the business requirements for the process framework, or the available implementation time;

We expect additional improvements to BIM as feedback from its use in other demanding projects, in distinct organizations, in different business domains, executed by distinct project teams, and, above all, with the complete life-cycle finished.

As with any other project, a BIM-based project should pay special care to the surrounding environment. The project stakeholders should guarantee to the project team a proper staffing, monetary resources, and time, as well as their own commitment.

We also showed that all the concepts and processes used in BIM can be implemented recurring to high-quality OSS frameworks. Despite BP-RMs are widely available, IAvOs need to be further developed. OBOs, and the PF, can be easily materialized mainly by the use of OSGi bundles (in SMX4) and the ODE BPEL orchestration engine.

Each software development activity has a cost, mainly triggered by requirements expressed by the clients. For that, attached to each IAvO could be calculated an estimation of the costs of each functional and non-functional requirement. The implementation of a requirement needs resources, which translates into a cost value. Before a project team implements such requirement, the stakeholders should be aware of its expected benefits, as well as of its costs. If this preliminary evaluation is not made, it may occur that requirements without business value are implemented instead of the more valuable ones. Such a cost evaluation inside IAvOs could give entrepreneurs a better decision and prioritization support to implement a proper sequence of the business needs. In addition, to evaluate the feasibility of the implementation of a requirement, the cost of acquiring or developing a set of OBOs to support it together with the customization effort should be balanced against the returned benefit out of its implementation.

Part IV - Epilogue

8. Conclusions

This chapter presents the synthesis of the lessons learned with the use of the Action Research approach, together with a final comparison of the evolution of the enablers and results of all Actions. We also present a critical analysis, based on the attainment of the goals of the thesis and an evaluation of the planned and the real dates of the activities of the thesis. We terminate by proposing future work that can be followed to extend the results obtained.

We conducted this thesis under the following two premises:

1. In any human activity, if excellence is the ultimate goal then standards are not enough.

In this thesis, we have applied this premise in the following situations:

- to the use of business process reference models, by adding additional content and organizational aspects;
- to the development processes, by adding an explicit use and tailoring of reference models;
- to the approach to conduct projects, by removing from project time the main software development effort;
- to the resulting software product, by creating a software process framework to run business software supporting business processes;
- to the Action Research approach, by adding quantifiable assessments and evaluation of stopping conditions.

2. The process to create business software representing business processes can be streamlined and optimized.

The level of abstraction provided by general-purpose programming languages is not sufficient to represent business processes in lean and clear ways. It is possible to propose and create development processes and software frameworks to accommodate business processes, in an easier way than current approaches used for software development processes.

It was always present in our mind that in a (near) future, it should be possible to implement business processes in software in straightforward and immediate manners.

8. Conclusions

It is a waste of time and resources, for instance, to create slide shows presentations, supposedly representing business processes, followed by costly and time-consuming projects, and at the end having a high risk that the implemented business processes are not the desired ones or that the business actors do not use the implemented business processes.

8.1. Lessons Learned

Models of business processes are key pieces of knowledge to improve the performance of organizations. The stakeholders of an organization must clearly decide how deep and in what manner they want to implement a BPM approach for their organization. If organizations use business process models in a superficial way (e.g. only for quality certification purposes) then organizations miss the opportunity to streamline and focus their activities in their core business. If organizations use business processes in a partial way, by not covering all of their functional areas, then the governance of business processes do not brings full benefits. Organizations should have in place governance actions to describe, integrate, and manage business activities both in a deep and in a comprehensive way in order to bring the highest value for their clients and for themselves.

Transforming a hierarchical organization into a process-oriented organization is a difficult task because the promoters of the change, and decision makers, are exactly the ones who will lose their importance inside the organization. However, if such barrier is eliminated, then managers of a process-oriented organization need a complete and integrated body of knowledge for their business processes. The body of knowledge should range from abstract models to implementations in software, in order to provide managers a complete perception and control over their own organization.

Business process models, and their respective software implementations, must represent exactly the same business content. For that, when thinking about business processes at high abstraction levels, it is an obvious mistake to forget issues related with software implementations. The same applies when implementing software, by forgetting how high level business processes are designed.

The ultimate goal for the efficiency of the implementation in software of business process models is that no model transformation at all would be necessary to run those models. If this goal is not attainable, then the existence of proper model transformations can bring significant benefits for the quality of the software product, like reducing the opportunity to have mistakes in such model transformations. Consequently, bringing some form of automation into model transformations improves the correctness of the final software product by implementing the business requirements more accurately. For that, it is necessary that the implementation in software of business process models be regulated by a development process, like BIM, that can help on transforming business process models into running software.

Business process reference models are excellent repositories of business knowledge. For that, it is highly recommended that this knowledge is used when defining the requirements of business soft-

ware. Nevertheless, organizations must always have the possibility to tailor reference models in order to embed their own business content. BIM explicitly promotes this possibility.

We positively evaluate the use of the Action Research approach in the context of software development. This evaluation is based on the existence of an intrinsic improvement cycle and by the steps included in the improvement cycle, which starts with a statement of the purpose of the Action, followed by the plan and execution of the activities, continues with the adjustments derived from the lessons learned, and ends by situating the activities in a social context. Additionally, since the development of business software is deeply related with organizations and, this way, also with persons, perfectly matches with the social reasoning included in Action Research.

We suggest improvements on the Action Research approach when used for software development purposes, namely by creating evaluations to assess the way the process was conducted and the characteristics of the resulting product. Additionally, we suggest and use the checking of a stop condition in order to have a more concrete notion of when the research efforts could have been stopped. In all evaluations, we propose and use quantifiable approaches, in order to reduce subjectiveness.

This thesis demonstrates the pertinence of using MES in industrial contexts. MES can bridge embedded software systems with ERP systems, by implementing lower abstraction levels of the business process models than those present in ERPs with the additional ability to interconnect to embedded software systems, often with real-time concerns.

After the realization of this thesis, we are convinced that:

- defined and explicitly managed business processes are a valuable asset for the organization;
- organizations must align the format and the content of the business process models with their mission, vision, and strategic goals;
- business software must seamlessly implement business processes models, including alternative designs for the same business process;

RUP showed adequate to the development of business software. However, RUP does not explicitly address the use of proper business content and the reuse of business software, and, for that, BIM is proposed. UML, being a de facto standard, is also adequate for the development of business software. In Centauro 2.0, we changed from UML to BPEL, as the language to depict business processes, due to the difficulty of having tools to support executable UML models (e.g. fUML (OMG, 2013)).

With BIM, we do not use code generation as the solution to obtain the software implementation of business process models. Instead, customization and orchestration of a software process framework is preferred. With this approach, the dependency on highly skilled software engineers is decoupled

from project time. The software process framework can be created, or preferentially reused, before the project starts. This way, the time needed for the project is reduced, as well as the quality of the resulting software product is enhanced because tests and error corrections are carried on before the project time. BIM strategy is to focus business experts on the most valuable purpose of a business software project: bring benefit to the organization through the implementation of adequate business processes.

In this thesis, we propose the Seppo software architecture as the software process framework to support business processes. The Seppo framework is implemented with free software and it includes support for non-functional requirements able to cope with MES implementations. Seppo also supports the use of OBOs and their orchestrations in BPEL, being a suitable framework to implement business process models in software. Additionally, Seppo can handle the concerns traversal to organizations, OAs, and thus reduces the complexity of the business process models. We see Seppo has a framework that eliminates the "missing link" between business process models and their implementation in software.

Nevertheless, Seppo, OBOs, business process orchestrations in BPEL, and OAs were incipiently tested because the Centauro 2.0 project was developed and tested in a prototypal environment.

8.2. Comments on the Actions

Table 8.1 presents the total score of the Enablers of all Actions. The scores express how the project teams conducted the Actions. From the highest possible score of 28 points (7 attributes * 4 maximum score per attribute), the Action with best score is SOL, followed ex-aequo by SIIA, Centauro 1.0, and Centauro 2.0. The Centauro 2.0 is penalized by the attribute "Implemented", which scored very low due to the surrounding social environment. The good results in all Actions (average of 22.6) are explained by the use of good development processes, such as RUP, and mainly by the quality of the development teams. Centauro 1.0 is a special Action because of the use of the F3 process, integrated with Bosch guidelines but incomplete in the technical content related with software development. Despite these constraints, the development team could overcome the limitations of F3.

Figure 8.1 shows a graphical representation of the Enablers of all actions. The soundness of the approach of SOL and Centauro 2.0 achieves the highest scores, while Centauro 1.0 had the lowest, explained by the use of RUP and BIM on the first two Actions and by the use of F3 on the Centauro 2.0 Action. The integration of the approach had a top value on Centauro 1.0, due to the tight connection to Bosch central IT directives. The implementation of the deployment has the lowest score in Centauro 2.0, caused by the social environment constraints occurred during that Action.

Table 8.2 presents the values for the Results of all Actions. It is worthwhile noticing the results related with requirements. The distinct project teams were able to implement all the functional requirements requested by the stakeholders, with the exception of Centauro 2.0 where only one functional require-

Table 8.1.: Scores for the Enablers of all Actions.

Element	Attributes to Score	PWage	SIIA	SOL	Centauro 1.0	Centauro 2.0
Approach	Sound	3	3	4	2	4
Approach	Integrated	2	3	3	4	3
Deployment	Implemented	2	4	4	4	1
Deployment	Systematic	3	3	3	2	4
Assessment and Review	Measurement	1	2	3	3	3
Assessment and Review	Learning	4	4	4	4	4
Assessment and Review	Improvement	4	4	4	4	4
Total		**19**	**23**	**25**	**23**	**23**
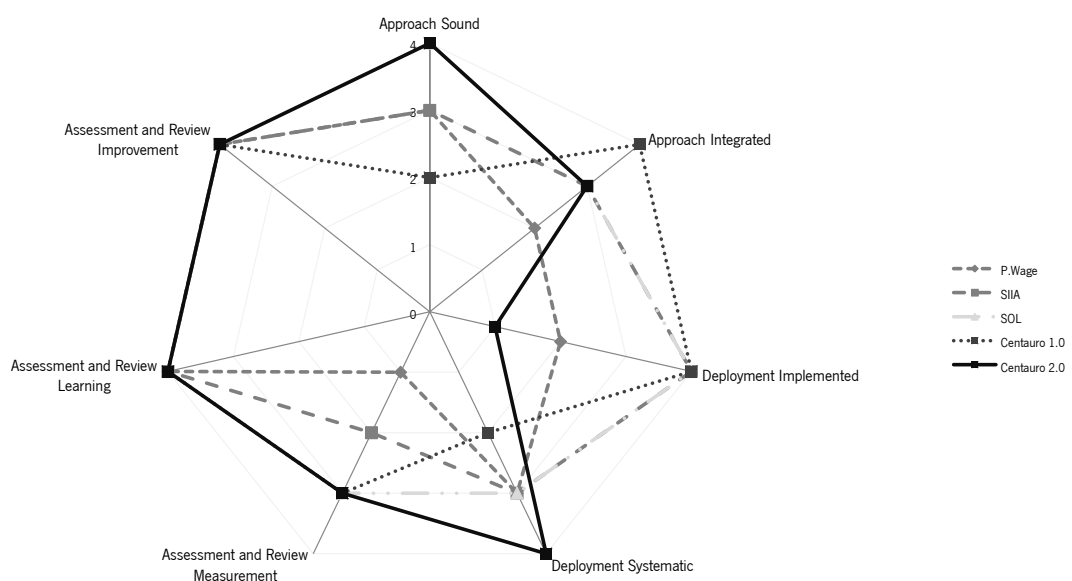
Figure 8.1.: Comparison of the Enablers of the Actions.

ment was implemented. The 100% results in functional requirements are a consequence of the social environment existing in Bosch CMP for projects, where this aspect is of top priority. The same 100% results are expressed in the non-functional requirements, where only two (MTBF and QA) requirements were not implemented in SIIA. The number of used software frameworks was increasing from Action to Action, resulting in more than 15 software frameworks in the Centauro 2.0.

8.3. Critical Analysis

In Table 8.3, for each Action we present the evaluations of the attainment of the goals of the thesis. In Centauro 2.0, we achieved the main goal (MG), namely by proposing a methodology (BIM) able to generate information and software systems for process oriented-organizations from business process

8. Conclusions

Table 8.2.: Values for the Results of all Actions.

Id	Metric Scope	Dimension of Analysis	Metric Name	Unit	PWage	SIIA	SOL	Centauro 1.0	Centauro 2.0
D1	Development Process	Requirements	Functional Requirements	%	17/17 (100%)	21/21 (100%)	22/20 (110%)	11/11 (100%)	1/11 (9%)
D2		Requirements	Non-Functional Requirements	%	5/5 (100%)	11/13 (85%)	12/12 (100%)	8/8 (100%)	8/8 (100%); 4/4 (100%)
D3		Quality	Quality Assessments	%	90%	90%	80%	n.e.	n.e.
D4		Cost	Effort	%	60/40 (150%)	498/300 (166%)	493/470 (105%)	28/28 (100%)	n.e.
D5		Time	Duration	%	24/24 (100%)	420/270 (156%)	660/354 (186%)	16/16 (100%)	n.e.
P1	Product	Complexity	SLOC	SLOC	1132	28518	16265	9490	n.e.
P2		Complexity	Used Software Frameworks	USF	0	5	12	>11	>15
P3		Soundness	Change Requests	CR	n.e.	4	0	3	n.e.
U1	Usage	Reliability	MTTR	min	n.e.	5	3	3	n.e.
U2		Reliability	MTBF	min	n.e.	720	>43200	>30240	n.e.

(n.e.: not evaluated)

models. As shown in Chapter 7, the BIM methodology uses business process reference models as input and generates orchestrations, for instance in BPEL, of already available software components (OBOs) allowing, this way, the creation of executable models of business processes.

The secondary goal (SG1) was accomplished in all Actions, except in Centauro 2.0 because no deployment of the software product was done, and thus preventing the evaluation.

For the same reason, the evaluation of the time needed to execute BIM (SG2) was only partially evaluated, but with positive results. The specification and implementation of a software framework to support the execution of business processes, the BPEL orchestrations, and reuse of OBOs (SG3) was accomplished with Seppo in Centauro 2.0.

The descriptions contained in BIM, based on the EPF metamodel, and the BPEL graphical representation of business processes (SG4) allow business experts to be standalone when defining and implementing information and software systems.

Since BIM starts with BP-RMs and ends with running software, for instance recurring to the Seppo framework, the SG5 is also accomplished.

Finally, we achieved the composition of software solutions based on free software and originated by different suppliers (SG6) by proposing, and using, Seppo and OBOs. OBOs have the characteristic to be interchangeable if the replacing component is expressing the same business content (in relation to the customized BP-RMs).

Table 8.3.: Evaluation of stopping conditions for all Actions.

Goal Id	Goal Description	SIIA	SOL	Centauro 1.0	Centauro 2.0
MG	To propose a methodology to generate immediately executable information and software systems for process-oriented organizations, starting with models of business requirements	not achieved	partially	not achieved	achieved
SG1	The resulting software products must have quality	achieved	achieved	achieved	not evaluated
SG2	The proposed methodology should allow development projects with a short execution of time	not achieved	not achieved	not achieved	partially
SG3	To specify and to implement a software framework to support the development of business software with the methodology	not achieved	partially	partially	achieved
SG4	The resulting methodologies and software products should be able to be used by business experts without intervention of software experts	not achieved	not achieved	not achieved	achieved
SG5	The proposed methodology should be holistic, ranging from business processes descriptions to running software	not achieved	partially	partially	achieved
SG6	To promote the composition of software solutions based on free software and originated from different vendors	not achieved	achieved	achieved	achieved

Table 8.4 shows the comparison between the initially planned dates and the real dates when the phases of our thesis were finished. The initial plan has a 6 months larger duration than the standard four years for a PhD thesis. This extension derives from the fact that we planned to perform the thesis in a part-time scheme, in addition to our normal job at Bosch CMP. Despite the planned and real dates for the "Preparation" and "Thesis Proposals" phases match, the reality is that we have revisited and updated many times the initial content of those phases. The "Execution of Actions" started earlier than the start of the work for the thesis, because SIIA project was already ongoing when the thesis was officially approved. Due to the Centauro 2.0 social constraints, the end of the execution of the Actions was delayed three years.

Table 8.4.: Execution of the phases of the PhD.

Phase	Name	Planned Begin Date	Real Begin Date	Planned End Date	Real End Date	Description
1	preparation	01.11.2006	01.11.2006	31.12.2007	31.12.2007	theoretical Research to draw the state-of-the-art
2	thesis proposal	01.01.2007	01.01.2007	31.12.2009	31.12.2009	define the contribute, in theory, methodologies, and tools
3	execution of Actions	01.11.2006	(01.10.2005)	31.10.2010	30.04.2013	implement and analyze the Actions
4	conclusions and text revision	01.01.2010	01.05.2013	15.12.2010	28.02.2014	extract conclusions from the Actions and make proposals for improvements
5	thesis presentation	1.st semester 2011	1.st or 2.nd quarter 2014	1.st semester 2011	1.st or 2.nd quarter 2014	public discussion of the thesis

8.4. Future Work

The most important proposals for the future work are related with BIM. BIM needs to be improved based on the feedback from more users, to incorporate characteristics collected out of additional practical work. BIM also needs to be fully tested during complex projects with many participants, in order to properly assess the expected benefits in the reduction of the time needed to finish a project for the development of business software and the quality of the final software product. The current level of abstraction of activities and artifacts present in BIM needs also to prove their adequacy for BIM users. The balance between a prescriptive approach and a more agile one can only be inferred after a significant number of BIM executions. Additionally, BIM needs to be tested with other BP-RMs other than SCOR.

In the context of university classes at under-graduate level, we expect to use BIM in projects performed by teams of students, based on real world problems and clients.

As future research topic, we have the intention to create IAvOs, namely for process-oriented software development organizations, in order to provide the community a basis to quickly implement new business companies. The same intention applies to the creation of BP-RMs.

Seppo can be extended in order to accommodate additional business process languages, other than BPEL, namely the BPMN (OMG, 2011b) through the use of other execution engines other than ODE, like Activiti[1] or JBoss jBPM[2].

Additionally, we intend to finish the proposal for a new version of the 4SRS technique (named "Patterned 4SRS") to derive candidate software architectures based on functional requirements. Among others, we intend to use "Patterned 4SRS" on the development of OBOs. "Patterned 4SRS" aims to improve the quality of the final software product by fulfilling adequately the functional requirements. "Patterned 4SRS" differs from 4SRS by proposing an earlier classification (in the domain of the problem) and then a proper choice of the most adequate design pattern. 4SRS always uses the Model-View-Controller (MVC) pattern to transform functional requirements into the logical architecture. Patterned 4SRS starts with the classification of use cases, taking into account some characteristics of the problem domain. We adopt design patterns as the mean to allow proper transition between the problem and the solution domains. We expect "Patterned 4SRS" to be able to provide guidance in the following dimensions:

- Apply distinct design patterns according to the use case nature. In the first step, use cases should be tagged according to their nature. Afterward, the classified use case is used to pick one or more design patterns previously considered successful according to the classification tags applied for a particular use case. In such a way, the set of use cases in a system or in a

---

[1]www.activiti.org
[2]www.jboss.org/jbpm

particular component (e.g. OBO) will be designed recurring to distinct design patterns.

- Include the use of architectural patterns. Starting with non-functional requirements, and using a similar approach to that of the functional requirements, the system architecture can be derived.

- Map the resulting conceptual design into a system architecture.

Bibliography

Abdallah, S., & Galal-Edeen, G.H. 2006. Towards a Framework for Enterprise Architecture Frameworks Comparison and Selection. In: The fourth international conference on informatics and systems (infos2006).

Abran, A., & Moore, J.W. 2004. Guide to the Software Engineering Body of Knowledge. IEEE Computer Society.

ACM. 2012. Computing classification system.

Acuna, S.T., & Juristo, N. 2005. Software process modeling. Vol. 10. Springer.

Adobe. 2008a. Blazeds developer guide.

Adobe. 2008b (August). Introducing cairngorm.

Ahern, D., Clouse, A., & Turner, R. 2008. Cmmi® distilled: a practical introduction to integrated process improvement. Addison-Wesley Professional.

Alavi, M., & Leidner, D. E. 2001. Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. Mis quarterly, **25**(1), 107–136.

Alexander, C. 1979. The Timeless Way of Building. Oxford University Press New York.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. 1977. A pattern language: towns, buildings, construction. Oxford University Press.

Alliance, OSGi. 2012 (September). Osgi adoption.

Alliance, The OSGi. 2009 (June). Osgi service platform core specification 4.2.

Ambler, S., Nalbone, J., & Vizdos, M. 2005. The enterprise unified process: Extending the rational unified process. Prentice Hall Press.

Ambler, S. W. 2006. The agile unified process (aup).

Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Gordijn, J., Grégoire, B., Schmitt, M., Dubois, E., Abels, S., et al. 2006. Towards a reference ontology for business models. Conceptual modeling-er 2006, 482–496.

Antony, J. 2004. Some pros and cons of six sigma: an academic perspective. The tqm magazine, **16**(4), 303–306.

Apache. 2009a. Apache camel manual. 2.0.0 edn. Apache Software Foundation.

Apache. 2009b. Apache ode user guide. Apache Software Foundation.

Apache. 2011. Apache karaf users' guide. Version 2.3.0 edn. Apache Software Foundation.

Apache Software Foundation. 2012 (July). Apache felix framework usage documentation. v4.0.3 edn. Apache Software Foundation.

APDSI. 2005. Glossário da Sociedade da Informação. Links.

Armistead, C., Pritchard, J.P., & Machin, S. 1999. Strategic business process management for organisational effectiveness. Long range planning, **32**(1), 96–106.

Avgeriou, P., & Zdun, U. 2005 (July). Architectural patterns revisited - a pattern language. Pages 1–39 of: Proceedings of 10th european conference on pattern languages of programs (europlop 2005).

Avison, David E, Lau, Francis, Myers, Michael D, & Nielsen, Peter Axel. 1999. Action research. Communications of the acm, **42**(1), 94–97.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (eds). 2003. The description logic handbook: Theory, implementation and applications. Cambridge University Press.

Baburoglu, O.N., & Ravn, I. 1992. Normative action research. Organization studies, **13**(1), 019.

Bartlett, N. 2009. Osgi in practice.

Baskerville, R., & Myers, M.D. 2004. Special issue on action research in information systems: Making IS research relevant to practice. Mis quarterly, **28**(3), 329–335.

Baskerville, R.L. 1999. Investigating information systems with action research. Communications of the ais, **2**(3es), 4.

Bass, L., Clements, P., & Kazman, R. 2003. Software architecture in practice. Addison-Wesley Professional.

Bauer, C., & King, G. 2006. Java Persistence with Hibernate. 2nd edition edn. Manning Publications Co.

Beatty, R.C., & Williams, C.D. 2006. ERP II: best practices for successfully implementing an ERP upgrade. Communications of the acm, **49**(3), 105–109.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. 2001. Manifesto for agile software development. The agile alliance, February.

Benedict, R. 1946. The chrysanthemum and the sword; patterns of japanese culture. Houghton Mifflin.

Benington, H.D. 1983. Production of large computer programs. Annals of the history of computing, **5**(4), 350–361.

Berners-Lee, T., Hendler, J., & Lassila, O. 2001. Scientific publishing on the semantic web. Scientific american, May.

Berry, D. 2004. Advice for Finishing that Damn Ph.D. University of Waterloo, Canada.

Bezivin, J. 2005. On the unification power of models. Software and systems modeling, **4**(2), 171–188.

BIRT, Eclipse. 2007. Business intelligence and reporting tools (birt).

Boehm, Barry W, Madachy, Ray, Steece, Bert, et al. 2000. Software cost estimation with cocomo ii. Prentice Hall PTR.

Boehm, B.W. 1988. A spiral model of software development and enhancement. Computer, **21**(5),

61–72.

Booch, G. 2000. The future of software. Pages 3–3 of: International conference on software engineering, vol. 22.

Bradshaw, D., Kennedy, M., & West, C. 2005. Oracle BPEL Process Manager Developer's Guide 10g Release 2. Oracle.

Bridgeland, D.M., & Zahavi, R. 2009. Business modeling: a practical guide to realizing business value. Morgan Kaufmann.

Brown, D., & Bahrs, P. 2009. Enterprise architecture for systems engineers. Ibm rational edge, **June**(June).

Brown, W.J., Malveau, R.C., McCormick III, H.W, & Mowbray, T.J. 1998. Antipatterns: refactoring software, architectures, and projects in crisis. Wiley.

Buschmann, F., Henney, K., & Schmidt, D.C. 2007. Pattern oriented software architecture: On patterns and pattern languages. Vol. 6. Wiley.

Campbell, D., Stonehouse, G., & Houston, B. 2002. Business strategy: an introduction. Second edition edn. Elsevier Butterworth-Heinemann.

Carvalho, Joao A. 2004. Doutoramento: Um grau Academico. Department of Information Systems, University of Minho.

Chaffee, E.E. 1985. Three models of strategy. The academy of management review, **10**(1), 89–98.

Chandler, A.D. 1962. Strategy and Structure: Chapters in the History of the American Enterprise. Beard Books Inc.

Chandra, C., & Tumanyan, A. 2007. Ontology-driven information system for supply chain management. Ontology handbook, 697–726.

Chappell, D. 2004. Enterprise service bus. O'Reilly Media, Incorporated.

Chen, S.K., Chung, J.Y., Cohen, M.A., Fu, S.S., & Gottemukkala, V. 2003 (Jan. 14). Dynamic business process automation system using xml documents. US Patent 6,507,856.

Chu, W., & Qian, D. 2007. Semantics based enterprise modeling for automated service discovery and service composition. Pages 439–445 of: Asia-pacific service computing conference, the 2nd ieee. IEEE.

Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. 2003. Documenting software architectures: views and beyond. Pages 740–741 of: Software engineering, 2003. proceedings. 25th international conference on. IEEE.

Cockburn, A. 1998. Surviving object-oriented projects: a manager's guide. Addison Wesley.

Combemale, B., Crégut, X., Caplain, A., & Coulette, B. 2006. Towards a rigorous process modeling with SPEM. Pages 530–533 of: 8th international conference on enterprise information systems: Databases and information systems integration.

Community, Apache ServiceMix. 2008 (February). Apache servicemix 3.x users' guide. web page.

Corporation, Microsoft. 2011 (July). Visual basic 6.0 resource center. Electronic.

Council, S.C. 2008a. Supply Chain Operations Reference Model (SCOR), Version 9.0.

Council, S.C. 2008b. Supply Chain Operations Reference-model, Version 9.0.

Council, Supply Chain. 2010 (September). Scor online access. electronic.

Council, USA Chief Information Officer. 2001 (February). A practical guide to federal enterprise architecture 1.0.

Curtis, B., Kellner, M.I., & Over, J. 1992. Process modeling. Communications of the acm, **35**(9), 75–90.

Dalal, N.P., Kamath, M., Kolarik, W.J., & Sivaraman, E. 2004. Toward an integrated framework for modeling enterprise processes. Communications of the acm, **47**(3), 83–87.

Dale, S. 2007. Holistic bpm: From theory to reality. In: Keynote presentation, 5th international conference on business process management, issn.

Daneva, M. 2003. Six Degrees of Success or Failure in ERP Requirements Engineering: Experiences with the ASAP Process. In: International workshop on cots and product software: Why requirements are so important, vol. 11.

Davenport, T., & Short, J. 1990. The new industrial engineering: Information technology and business process redesign. cambridge, massachusetts: Center of. Information systems research, mass. institute of technology, sloan school of management.

Davenport, T.H. 1993. Process Innovation: Reengineering Work Through Information Technology. Harvard Business School Press.

De Ugarte, B.S., Artiba, A., & Pellerin, R. 2009. Manufacturing execution system–a literature review. Production planning and control, **20**(6), 525–539.

Delen, D., Dalal, N.P., & Benjamin, P.C. 2005. Integrated modeling: the key to holistic understanding of the enterprise. Communications of the acm, **48**(4), 107–112.

Deming, W. E. 1982. Out of the Crisis. Cambridge: Massachusetts institute of technology.

Deming, W. E. 2000. Out of the Crisis. MIT Press.

Dennis, Jack. 1975. The design and construction of software systems. Pages 12–28 of: Bauer, F., Dennis, J., Waite, W., Gotlieb, C., Graham, R., Griffiths, M., Helms, H., Morton, B., Poole, P., & Tsichritzis, D. (eds), Software engineering. Lecture Notes in Computer Science, vol. 30. Springer Berlin / Heidelberg.

Devedzic, V. 2002. Software Patterns. Handbook of Software Engineering & Knowledge Engineering.

Dirksen, J. 2010. Getting started with servicemix 4. eletronic, www.dzone.com. Refcardz 65.

Dixit, A.K., & Nalebuff, B. 1991. Thinking Strategically: The Competitive Edge in Business, Politics, and Everyday Life. Norton.

Duarte, F.J., Fernandes, J.M., & Machado, R.J. 2007. Reference Modeling for Business Systems Analysis. Idea Group Pub., an imprint of Idea Group Inc. Chap. Business modeling in process-oriented organizations for RUP-based software development.

Duarte, Francisco J. 2002 (July). Engenharia de software orientada aos processos. Msc. thesis, Universidade do Minho.

Duncan, W.R. 1996. PMBOK–A Guide to the Project Management Body of Knowledge. Zda: Project

management institute (pmi).

EABOK. 2004. Eabok. The MITRE Corporation.

Eclipse. 2012 (May). Bpel designer project.

Eclipse Foundation. 2012 (September). Getting started with equinox. v3.8.1 edn. Eclipse Foundation.

Eeles, P. 2001. Capturing architectural requirements. The rational edge, **November**, 2011.

EFQM. 2001. EFQM Excellence Model - Large Companies, Operational and Business Units version. European Foundation for Quality Management.

EFQM. 2010. Introducing the efqm excellence model 2010. online.

EFQM. 2011 (May). Efqm members. online.

Elzinga, D.J., Horak, T., Lee, C.Y., & Bruner, C. 1995. Business process management: survey and methodology. Ieee transactions on engineering management, **42**(2), 119–128.

Encyclopaedia Britannica. 2010 (June). Encyclopaedia Britannica.

EPF. 2011 (January). Eclipse Process Framework (EPF).

EPF. 2012. Openup v1.5.1.4.

Favre, J.M. 2004. Towards a basic theory to model model driven engineering. In: 3rd workshop in software model engineering, wisme. Citeseer.

Fernandes, J.M., & Duarte, F.J. 2004. Using RUP for Process-Oriented Organisations. 5th int. conf. on product focused software process improvement (profes 2004), lecture notes in computer science, **3009**, 348–62.

Fernandes, J.M., & Duarte, F.J. 2005. A Reference Framework for Process-oriented Software Development Organizations. Software and systems modeling (sosym), **4**(1), 94–105. DOI 10.1007/s10270-004-0063-0.

Fernandes, J.M., & Machado, R.J. 2001. From Use Cases to Objects: An Industrial Information Systems Case Study Analysis. Pages 319–28 of: 7th international conference on object-oriented information systems (oois 01).

Fettke, P., Loos, P., & Zwicker, J. 2005. Business Process Reference Models: Survey and Classification. Workshop on Business Process Reference Models (BPMR 2005), 1–15.

Firesmith, D.G., & Henderson-Sellers, B. 2002. The open process framework: An introduction. Addison-Wesley Professional.

Foundation, Apache Software. 2011 (March). Apache servicemix 4.3.

Fowler, M. 1997. Analysis patterns: reusable object models. Addison-Wesley.

Fowler, M. 2003. Patterns of enterprise application architecture. Addison-Wesley Professional.

Fowler, M. 2004. Inversion of control containers and the dependency injection pattern. Url: http://martinfowler.com/articles/injection.html.

Fowler, M., Parsons, R., & MacKenzie, J. 2000. Pojo, an acronym for: Plain old java object.

France, R., & Rumpe, B. 2005. Domain specific modeling. Software and systems modeling, **4**(1), 1–3.

Fraternali, Piero, Rossi, Gustavo, & Sánchez-Figueroa, Fernando. 2010. Rich internet applications.

Internet computing, ieee, **14**(3), 9–12.

Fuse. 2011 (September). Fuse esb product introduction v4.1.1.

FuseSource. 2012. The large hadron collider project and fuse message broker.

Gable, G.G. 1994. Integrating case study and survey research methods: an example in information systems. European journal of information systems, **3**(2), 112–126.

Galler, Bernard A. 1962. Definition of software. Commun. acm, **5**(1), 6–.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. 1993. Design patterns: Abstraction and reuse of object-oriented design. Ecoop93 object-oriented programming, 406–431.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. 1995. Design patterns: elements of reusable object-oriented software.

Gamma, Erich, & Beck, Kent. 2006. Junit.

Gardner, T. 2003. Uml modelling of automated business processes with a mapping to bpel4ws. In: Proceedings of the first european workshop on object orientation and web services at ecoop, vol. 2003. Citeseer.

Garshol, L.M., Moore, G., & SC34, JTC1 /. 2008 (June). Iso iec 13250-2 topic maps data model.

Georgakopoulos, D., Hornick, M., & Sheth, A. 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and parallel databases, **3**(2), 119–153.

Gershenfeld, N., Krikorian, R., & Cohen, D. 2004. The internet of things. Scientific american, **291**(4), 76–81.

Ghezzi, C., Jazayeri, M., & Mandrioli, D. 1991. Fundamentals of Software Engineering. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Gordijn, J. 2004. e-business model ontologies. book chapter contribution to" e-business modelling using the e3value ontology", wendy curry (editor), 98–128.

Grady, R.B. 1992. Practical software metrics for project management and process improvement. Prentice-Hall, Inc.

Graham, I., Henderson-Sellers, B., & Younessi, H. 1997. The open process specification. Acm Press.

Greenfield, J., & Short, K. 2004. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley.

Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., & Shan, M.C. 2004. Business process intelligence. Computers in industry, **53**(3), 321–343.

Gruber, T.R. 1993. A translation approach to portable ontology specifications. Knowledge acquisition, **5**(2), 199–220.

Gruhn, V., & Wellen, U. 2001. Process Landscaping: Modelling Distributed Processes and Proving Properties of Distributed Process Models. Lecture notes in computer science, **2128**(S 103).

Hadley, M., & Sandoz, P. 2009. Jax-rs: Java api for restful web services. Jsr, jcp, september.

Hagan, P. 2004. Historical developments in ea. Eabok, January, 22–25.

Hamel, G. 2000. Leading the Revolution, Harvard Business School Press. Boston. ma.

Hammer, M. 1990. Reengineering Work: Don't Automate, Obliterate. Harvard business review, **68**(4), 104–112.

Hammer, M. 1997. Beyond Reengineering: How the Processed-Centered Organization is Changing Our Work and Our Lives. Harperbusiness: New york, ny, usa.

Hammer, M., & Champy, J. 1993. Reengineering the corporation. Harper Business.

Harkins, B.L., Middleton, E.S., & Mushin, DA. 1999. Linking the plant floor to the enterprise: benefits and pitfalls. Hydrocarbon processing, **78**, 49–56.

Hars, A. 1994. Referenzdatenmodelle: Grundlagen effizienter Datenmodellierung. Gabler, wiesbaden.

Hartmann, Edward. 1992. Successfully installing tpm in a non-japanese plant: total productive maintenance. TPM Press London.

Haumer, P. 2005. IBM Rational Method Composer: Part 1: Key concepts. The rational edge, **12**, 15.

Helkiö, P., Seppälä, A., & Syd, O. 2006. Evaluation of Intalio BPM tool. Special course in information system integration.

Henderson-Sellers, B. 2002. Process metamodelling and process construction: examples using the open process framework (opf). Annals of software engineering, **14**(1), 341–362.

Hepp, M., Leymann, F., Domingue, J., Wahler, A., & Fensel, D. 2005. Semantic business process management: A vision towards using semantic web services for business process management. Pages 535–540 of: e-business engineering, 2005. icebe 2005. ieee international conference on. IEEE.

Higgins, B. 2005. Ground rules for managing business process integration projects. Ibm rational edge, **July**.

Hirschheim, R.A., Klein, H.K., & Lyytinen, K. 1995. Information systems development and data modeling: conceptual and philosophical foundations. Vol. 9. Cambridge Univ Pr.

Hofer, C.W. 1973. Some preliminary research on patterns of strategic behavior. Pages 46–59 of: Academy of management proceedings, vol. 33.

Hohpe, G., & Woolf, B. 2002. Enterprise integration patterns. Enterprise integration patterns: Designing, building, and deploying messaging solutions. addison-wesley.

Hohpe, G., & Woolf, B. 2004. Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional.

Hruby, P., Kiehn, J., & Scheller, C.V. 2006. Model-driven design using business patterns. Springer.

IBM. 2009. Websphere application server community edition v2.1.1 documentation. IBM.

IBM. 2012. Websphere process server for multiplatforms, version 7.0.

IBM Rational. 2006. Ibm rational unified process (rup) v7.0.1.

IEEE SWEBOK 3. 2008 (February). Software measurement knowledge area (swebok 3 draft).

IEEE SWEBOK 3. 2012a. Computing foundations knowledge area (swebok 3 reviewed draft).

IEEE SWEBOK 3. 2012b. Mathematical foundations knowledge area (swebok 3 reviewed draft).

IEEE SWEBOK 3. 2012c. Software engineering models and methods knowledge area (swebok 3 reviewed draft).

IEEE SWEBOK 3. 2012d. Software engineering professional practice knowledge area (swebok 3 reviewed draft).

IEEE SWEBOK 3. 2012e. Software security specialized knowledge area (swebok 3 draft).

Institute, Project Management. 2004. A guide to the project management body of knowledge: PMBOK Guide.

ISA. 2005. Ansi/isa-95.00.03-2005 enterprise-control system integration, part 3: Models of manufacturing operations management.

ISA. 2007. Ansi/isa-95.00.05-2007 integration, part 5: Business-to-manufacturing transactions.

ISA. 2010a. Ansi/isa-95.00.01-2010 (iec 62264-1 mod) enterprise-control system integration - part 1: Models and terminology.

ISA. 2010b. Ansi/isa-95.00.02-2010 (iec 62264-2 mod) enterprise-control system integration - part 2: Object model attributes.

ISO/IEC. 2007 (October). Common logic, a framework for a family of logic-based languages.

ISO/IEC/IEEE. 2010 (December). 24765-2010 - systems and software engineering – vocabulary.

Jackson, M. 2001. Problem frames: analyzing and structuring software development problems. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Jacobson, I., Booch, G., & Rumbaugh, J. 1999. The Unified Software Development Process. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Jensen, K. 1992. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. Eatcs monographs in computer science, **26**.

Jensen, K., & Kristensen, L.M. 2009. Coloured petri nets: Modelling and validation of concurrent systems. Springer.

Johansson, H.J., Patrick, M., Pendlebury, J., & W., Wheeler. 1993. Business process reengineering: breakpoint strategies for market dominance. Wiley.

Johnson, R., Hoeller, J., Arendsen, A., Risberg, T., & Sampaleanu, C. 2005. Professional Java Development with the Spring Framework. John Wiley & Sons.

Johnson, R., Hoeller, J., Arendsen, A., Sampaleanu, C., Harrop, R., Risberg, T., Davison, D., Kopylenko, D., Pollack, M., Templier, T., et al. 2012. The spring framework-reference documentation.

Jorysz, HR, & Vernadat, FB. 1990. Cim-osa part 1: total enterprise modelling and function view. International journal of computer integrated manufacturing, **3**(3-4), 144–156.

Jost. 2006 (October). The age of the cio is over.

Juric, M.B., Mathew, B., & Sarang, P.G. 2004. Business process execution language for web services. Pakt Publishing.

Kalleberg, A.L., & Berg, I.E. 1987. Work and industry: Structures, markets, and processes. Springer.

Kaplan, R.S., & Norton, D.P. 2008. Mastering the management system. harvard business review, **86**(1), 62.

Kay, J.A. 1995. Why firms succeed. Oxford University Press, USA.

Kelly, M.B. 2003. The telemanagement forum's enhanced telecom operations map (eTOM). Journal

of network and systems management, **11**(1), 109–119.

Kent, S. 2002. Model driven engineering. Pages 286–298 of: Integrated formal methods. Springer.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C.V., Longtier, J.M., & Irwin, J. 1997. Aspect-oriented programming. Springer-verlag lncs, **1241**, 16.

Kim, W.C., & Mauborgne, R. 2005. Blue ocean strategy: How to create uncontested market space and make the competition irrelevant. Harvard Business School Pr.

Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., & Trickovic, I. 2005. Ws-bpel extension for people–bpel4people. Joint white paper, ibm and sap, **183**, 184.

Ko, R.K.L., Lee, S.S.G., & Lee, E.W. 2009. Business process management (bpm) standards: a survey. Business process management journal, **15**(5), 744–791.

Krasner, G.E., & Pope, S.T. 1988. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. Journal of object-oriented programming, **1**(3), 26–49.

Kroll, P. 2005. Introducing IBM Rational Method Composer. The rational edge.

Kruchten, P. 2000. The rational unified process: an introduction. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Kruchten, P. 2004. The rational unified process: an introduction. Addison-Wesley Professional.

Laney, R., Barroca, L., Jackson, M., & Nuseibeh, B. 2004. Composing requirements using problem frames. Pages 122–131 of: Requirements engineering conference, 2004. proceedings. 12th ieee international.

Lee, J., Lee, D., & Kang, S. 2007. An overview of the business process maturity model (bpmm). Advances in web and network technologies, and information management, 384–395.

Lee, RG, & Dale, BG. 1998. Business process management: a review and evaluation. Journal, vol, **4**(3), 214–225.

Leibert, J., Muroski, M., Fraser, J., Davis, T. L., Towle, S. Tebbenhoff, P., Navarro-Robertroy, S., Huff, D., Chance, C., Wells, M., Woods, C., Kukla, D., Hakanson, B., & Zailyk, S. 1997. Mes functionalities and mrp to mes data flow possibilities. Mesa international white paper.

Lewin, K. 1947. Frontiers in group dynamics. Human relations, **1**(2), 143.

Leymann, F., et al. 2001. Web services flow language (wsfl 1.0).

Liang, Y. 2003. From use cases to classes: a way of building object model with UML. Information and software technology, **45**(2), 83–93.

Loos, P., & Fettke, P. 2010 (September). Rmk reference models catalogs project. electronic.

Louis, A., & Dutoo, M. 2008 (July). Choosing between routing and orchestration in an esb. InfoQ.

Lu, R., & Sadiq, S. 2007. A survey of comparative business process modeling approaches. Pages 82–94 of: Business information systems. Springer.

Marshall, C. 2000. Enterprise modeling with UML: designing successful software through business analysis. Addison-Wesley Professional.

McCarthy, W.E. 1982. The rea accounting model: A generalized framework for accounting systems in

a shared data environment. Accounting review, 554–578.

McCracken, D.D., & Jackson, M.A. 1982. Life cycle concept considered harmful. Acm sigsoft software engineering notes, **7**(2), 29–32.

McIlroy, M.D., Buxton, JM, Naur, P., & Randell, B. 1968. Mass-produced software components. Pages 88–98 of: Proceedings of the 1st international conference on software engineering, garmisch pattenkirchen, germany. sn.

Meersman, R. 1999. Semantic ontology tools in is design. Foundations of intelligent systems, 30–45.

Melenovsky, M.J., & Sinur, J. 2006. Bpm maturity model identifies six phases for successful bpm adoption. Gartner research, stamford.

Mellor, S.J., Clark, A.N., & Futagami, T. 2003. Model-driven development. Ieee software, 14–18.

Mendling, J., Moser, M., & Neumann, G. 2006. Transformation of yepc business process models to yawl. Pages 1262–1266 of: Proceedings of the 2006 acm symposium on applied computing. ACM.

Meservy, T.O., & Fenstermacher, K.D. 2005. Transforming software development: an MDA road map. Computer, **38**(9), 52–58.

Microsoft. 2010. Bpel import (biztalk server sample). Msdn.

Microsoft Navision. 2005 (May). Automotive manufacturer deploys integrated erp solution and boosts customer service in just 15 days.

Miller, J., Mukerji, J., et al. 2001. Model driven architecture (mda). Object management group, **Draft Specification ormsc/2001-07-01**.

Miller, J., Mukerji, J., et al. 2003. MDA Guide Version 1.0.1.

Mitra, T. 2005. Business-driven development. Ibm developer works.

Myers, M.D. 2008. Qualitative research in business & management. Sage Publications Ltd.

Nagios. 2013 (July). Nagios quickstart installation guides.

Noy, N.F., McGuinness, D.L., et al. 2001. Ontology development 101: A guide to creating your first ontology.

Nuseibeh, B. 2001. Weaving together requirements and architectures. Computer, **34**(3), 115–119.

OASIS. 2006 (May). Business-centric methodology (bcm).

OASIS. 2007 (April). Web services business process execution language v2.0.

of Defense (USA), Department. 2010 (August). Dodaf 2.02.

OGC. 2008. Best management practice: Itil v3 and iso/iec 20000.

OMG. 2000 (June). Currency specification.

OMG. 2006 (January). Meta-object facility (mof) core specification.

OMG. 2008a (June). Business process maturity model (bpmm), version 1.0.

OMG. 2008b (April). Software & systems process engineering metamodel specification (spem).

OMG. 2009a (January). Business process modeling notation (bpmn) 1.2.

OMG. 2009b (May). Ontology definition metamodel.

OMG. 2011a (November). Alert management service (almas) specification.

OMG. 2011b. Business Process Model and Notation (BPMN), version 2.0. Omg document number, formal/2011-01-03.

OMG. 2011c (Jan). Meta object facility (mof) 2.0 query/view/transformation specification.

OMG. 2011d (August). Omg unified modeling language (omg uml), superstructure.

OMG. 2013 (August). Semantics of a foundational subset for executable uml models (fuml).

Oracle. 2011 (December). Jsr 224: Java api for xml-based web services (jax-ws).

Oracle. 2012. Oracle bpel process manager overview. Oracle technology network.

Oracle. 2014. Oracle unified method (oum). oracle∏s full lifecycle method for deploying oracle-based business solutions. Oracle white paper, February.

Oracle PeopleSoft. 2007. Peoplesoft enterprise - rapidstart for healthcare.

Osterwalder, A., Pigneur, Y., et al. 2002. An e-business model ontology for modeling e-business. Pages 17–19 of: 15th bled electronic commerce conference. Bled, Slovenia.

Osterwalder, A., Pigneur, Y., & Tucci, C.L. 2005. Clarifying Business Models: Origins, present and Future of the Concept. Communications of ais, **16**.

Ould, M.A. 1995. Business Processes: Modelling and analysis for re-engineering and improvement. John Wiley & Sons, Chichester.

Papazoglou, M.P., Traverso, P., Dustdar, S., & Leymann, F. 2007. Service-oriented computing: State of the art and research challenges. Computer, **40**(11), 38–45.

Pearlson, K.E., & Saunders, C.S. 2006. Managing and using information systems: A strategic approach. Wiley.

Peraire, C., & Pannone, R. 2005. The ibm rational unified process for cots-based projects: An introduction. developerworks, August.

Pernici, B., & Weske, M. 2006. Business process management. In: Guest editorial of data & knowledge engineering 56. Elsevier.

Pidcock, W. 2003. What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model? http://www.metamodel.com/article.php?story=20030115211223271, **January**.

Pinto, H.S., & Martins, J.P. 2004. Ontologies: How can they be built? Knowledge and information systems, **6**(4), 441–464.

Porter, M. 1985. Competitive Advantage: Creating and Sustaining Superior Performance. New york: Free press.

Porter, M.E. 1980. Competitive strategy: techniques for analyzing industries and competitors. Free Press.

Porter, M.E., & Millar, V.E. 1985. How information gives you competitive advantage. Harvard business review, **63**(4), 149–160.

Porter, M.E., & Millar, V.E. 1999. How Information Gives You Competitive Advantage: The Information Revolution Is Transforming the Nature of Competition. Knowledge and special libraries, 85.

Pressman, Roger S. 1988. Making software engineering happen: a guide for instituting the technology.

Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Pressman, R.S., & Ince, D. 1992. Software engineering: a practitioner's approach. Vol. 5. McGraw-hill New York.

Recker, J.C., Rosemann, M., Indulska, M., & Green, P. 2009. Business process modeling: a comparative analysis. Journal of the association for information systems, **10**(4), 333–363.

Rosemann, M., & de Bruin, T. 2005. Application of a holistic model for determining bpm maturity. Business process trends.

Ross, J.W., Weill, P., & Robertson, D. 2006. Enterprise architecture as strategy: Creating a foundation for business execution. Harvard Business School Pr.

Roughley, I. 2007. Starting Struts 2.

Royce, W. 1970. Managing the development of large software systems: concepts and techniques. In: Proceedings of ieee westcon 1970.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. 1991. Object-oriented modeling and design. Prentice-Hall, Inc. Upper Saddle River, NJ, USA.

Rumbaugh, J., Jacobson, I., & Booch, G. 1998. The Unified Modeling Language reference manual. Addison-Wesley Longman Ltd. Essex, UK, UK.

Rummler, G.A., & Brache, A.P. 1995. Improving performance: How to manage the white space on the organization chart. Jossey-Bass San Francisco.

Santos, Nuno, Duarte, Francisco J, Machado, Ricardo J, & Fernandes, Joao M. 2013. A transformation of business process models into software-executable models using mda. Pages 147–167 of: Software quality. increasing value in software and systems development. Springer.

SAP. 2004. Sap .net connector.

Scheer, August-Wilhelm, Jost, Wolfram, & Guengoez, O. 2007. A reference model for industrial enterprises. Reference modeling for business systems analysis, 167–181.

Scheer, A.W. 1999. ARIS–House of Business Engineering: Konzept zur Beschreibung und Ausf "uhrung von Referenzmodellen. Referenzmodellierung. state-of-the-art und entwicklungsperspektiven. heidelberg, 2–21.

Scheer, A.W. 2000. Aris-business process modeling. Springer-Verlag New York, Inc. Secaucus, NJ, USA.

Schekkerman, J. 2003. How to survive in the jungle of enterprise architecture frameworks: Creating or choosing an enterprise architecture framework. Architecture.

Schmidt, D.C. 2006. Guest editor's introduction: Model-driven engineering. Computer, **39**(2), 25–31.

Schmidt, D.C., Fayad, M., & Johnson, R.E. 1996. Software patterns. Communications of the acm, **39**(10), 37–39.

Scholten, B. 2009. Mes guide for executives: Why and how to select, implement, and maintain a manufacturing execution system. International Society for Automation.

Schuster, R., & Motal, T. 2009. From e3-value to rea: Modeling multi-party e-business collaborations. Pages 202–208 of: Commerce and enterprise computing, 2009. cec'09. ieee conference on.

IEEE.

Scrum Alliance. 2012. Scrum is an innovative approach to getting work done.

Selic, B. 2011. The theory and practice of modeling language design for model-based software engineering - a personal perspective. Generative and transformational techniques in software engineering iii, 290–321.

semanticweb.org. 2012 (January). Ontologies on semanticweb.org.

Sewall, S.J. 2003 (November). Executive Justification for Adopting Model Driven Architecture (MDA).

Shalloway, A., & Trott, J. 2002. Design Patterns Explained: A New Perspective on Object-Oriented Design. 2.nd edn. Addison-Wesley Professional.

Shapiro, R.M. 2006. Xpdl 2.0: Integrating process interchange and bpmn. Workflow handbook, 183–194.

Shaw, M. 1996. Some patterns for software architectures. Pattern languages of program design, **2**, 255–269.

Shaw, M., & Garlan, D. 1996. Software architecture: perspectives on an emerging discipline. Prentice Hall.

Shingo, S., Shingᴏ, S., & Dillon, A.P. 1989. A study of the Toyota production system from an industrial engineering viewpoint. Productivity Press.

Siau, K. 2007. The future of information systems engineering. Requirements engineering, **12**(4), 199–201.

Simoes, J. 2006. Analise da implementacao de cmm nivel 2. Msc, University of Minho.

Smith, H., & Fingar, P. 2002. Business Process Management: the Third Wave. Meghan-Kiffer Press.

Snabe, J.H., Rosenberg, A., Møller, C., & Scavillo, M. 2008. Business process management: The sap roadmap. Galileo Press, SAP Press.

Spencer, J., et al. 2004. Togaf enterprise edition version 8.1.

Spencer, M. 1999. Gpl open source framework for building communications applications.

Sprinkle, J., & Karsai, G.G. 2004. A domain-specific visual language for domain model evolution. Journal of visual languages & computing, **15**(3-4), 291–307.

Staab, S., Walter, T., Gr
"oner, G., & Parreiras, F. 2010. Model Driven Engineering with Ontology Technologies. Reasoning web. semantic technologies for software engineering, 62–98.

Staab, Steffen, & Studer, R. 2004. Handbook on ontologies (international handbooks on information systems). Springer.

Stahl, T., & Völter, M. 2006. Model-driven Software Development: Technology, Engineering, Management. John Wiley & Sons.

Steeples, M.M. 1993. The corporate guide to the Malcolm Baldrige National Quality Award: proven strategies for building quality into your organization. Irwin Professional Publishing.

Sugimori, Y, Kusunoki, K, Cho, F, & Uchikawa, S. 1977. Toyota production system and kanban system materialization of just-in-time and respect-for-human system. The international journal

of production research, **15**(6), 553–564.

Sutherland, J. 1993. The roots of scrum: How the japanese experience changed global.

Takeuchi, H., & Nonaka, I. 1986. The new new product development game. Harvard business review, **64**(1), 137–146.

Team, CMMI Product. 2010. Capability maturity model integration for development, version 1.3. TECHNICAL REPORT CMU/SEI-2010-TR-033. Software Engineering Institute, Carnegie Mellon University.

Temnenco, Vitalie. 2007. Togaf or not togaf: Extending enterprise architecture beyond rup. Rational edge, **January**.

Ten-Hove, R., & Walker, P. 2005 (August). Java business integration (jbi) 1.0-jsr 208 final release.

Thatte, S. 2001. Xlang: Web services for business process design. Microsoft corporation, 13.

Thomas, O. 2006. Understanding the term reference model in information systems research: history, literature analysis and explanation. Lecture notes in computer science, **3812**, 484.

Topi, H., Valacich, J.S., Wright, R.T., Kaiser, K.M., Nunamaker Jr, J.F., Sipior, J.C., & Vreede, G.J. 2010. Is 2010: curriculum guidelines for undergraduate degree programs in information systems.

Trist, E. 1976. Engaging with large-scale systems. Plenum, New York.

Urbaczewski, Lise, & Mrdalj, Stevan. 2006. A comparison of enterprise architectures frameworks. Issues in information systems, **VII, No. 2**.

USAgov. 2002 (December). E-Government Act of 2002. USA PUBLIC LAW 107 347.

Uschold, M. 2003. Where are the semantics in the semantic web? Ai magazine, **24**(3), 25.

van der Aalst, W. 2003a. Challenges in business process management: Verification of business processes using Petri nets. Bulletin of the eatcs, **80**, 174–199.

van der Aalst, W., Ter Hofstede, A., & Weske, M. 2003. Business process management: A survey. Business process management, 1019–1019.

van der Aalst, W.M.P. 2003b. Patterns and XPDL: A critical evaluation of the XML process definition language. Qut technical report fit-tr-2003-06.

van der Aalst, W.M.P., & Lassen, K.B. 2005. Translating workflow nets to bpel. Beta, Research School for Operations Management and Logistics.

Van Der Aalst, W.M.P., & Ter Hofstede, A.H.M. 2005. Yawl: yet another workflow language. Information systems, **30**(4), 245–275. DOI 10.1016/j.is.2004.02.002.

van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., & Wohed, P. 2002. Pattern-based analysis of bpml (and wsci). Tech. rept.

van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Kiepuszewski, B., & Barros, A.P. 2003. Workflow patterns. Distributed and parallel databases, **14**(1), 5–51.

van Deursen, A., Klint, P., & Visser, J. 2000. Domain-specific languages: An annotated bibliography. Acm sigplan notices, **35**(6), 26–36.

Van Dyk, L. 2000. Manufacturing execution systems. Ph.D. thesis, University of Pretoria.

Vasconcelos, A., Caetano, A., Neves, J., Sinogas, P., Mendes, R., & Tribolet, J. 2001. A framework for modeling strategy, business processes and information systems. Pages 69–80 of: Enterprise distributed object computing conference, 2001. edoc'01. proceedings. fifth ieee international. IEEE.

VDI. 2007 (December). VDI guideline: VDI 5600 - Manufacturing Execution Systems (MES).

Vernadat, F.B. 1996. Enterprise Modeling and Intergration: principles and applications. Springer.

Vlissides, J.M., Coplien, J.O., & Kerth, N.L. 1996. Pattern languages of program design. Vol. 2. Addison-Wesley Longman Publishing Co., Inc.

Vollmer, K. 2011. Enterprise service bus, q2 2011. Tech. rept. Q2 2011. The Forrester Wave.

Von Krogh, G., Nonaka, I., & Aben, M. 2001. Making the most of your company's knowledge: a strategic framework. Long range planning, **34**(4), 421–439.

von Neumann, J., & Morgenstern, O. 1944. Theory of games and economic behavior. Princeton University Press.

W3C. 2004 (February). Resource description framework (rdf): Concepts and abstract syntax.

W3C. 2008 (January). Sparql query language for rdf.

W3C. 2009 (October). Owl 2 web ontology language structural specification and functional-style syntax.

Wand, Y., & Weber, R. 1990. An ontological model of an information system. Ieee transactions on software engineering, **16**(11), 1282–1292.

Wand, Y., & Weber, R. 2002. Research commentary: information systems and conceptual modeling-a research agenda. Information systems research, **13**(4), 363–376.

Weber, I., Haller, J., & Mulle, J.A. 2008. Automated derivation of executable business processes from choreographies in virtual organisations. International journal of business process integration and management, **3**(2), 85–95.

Weske, M. 2007. Business Process Management: Concepts, Languages, Architectures. Springer-Verlag New York Inc.

Whytock, S. 1993. The development life-cycle. Wiley series in software based systems, 81–96.

Womack, James P, Jones, Daniel T, & Roos, Daniel. 1990. The machine that changed the world: the story of lean production. New york: Rawson associates.

Woolf, B. 2007. Esb-oriented architecture: The wrong approach to adopting soa. Ibm developerworks, September.

Yin, R.K. 1984. Case study research: Design and methods. Beverly Hills, California: Sage publications, INC.

Zachman, J.A. 1987. A framework for information systems architecture. Ibm systems journal, **26**(3), 276–292.

Zairi, M. 1997. Business process management: a boundaryless approach to modern competitiveness. Journal, vol, **3**(1), 64–80.

Part V - Appendices

A. BIM Formalization in EPF

In this appendix, we show several views of the formalization of BIM in the EPF.

Presentation Name	Index	Predecessors
BIM Life-cycle	0	
Selection	1	
Selection Iteration [1..n]	2	
Analyze Process RMs	3	
Check Available Business Process Reference Models (BP-RM)	4	
Choose Candidate Business process Reference Models (BP-RM)	5	4
Analyze IAvOs	6	
Check Available Instantaneously Available Organizations (IAvO)	7	
Choose Candidate Instantaneously Available Organizations (IAvO)	8	7
Analyze OAs	9	
Check Available Organizational Aspects (OA)	10	
Choose Candidate Oraganizational Aspects (OA)	11	10
Define Generic Process Framework	12	3,6,9
Define the Generic Process Framework	13	
QA1	14	2
Definition	15	1
Definition Iteration [1..n]	16	
Define Business Needs	17	
Create Business Definition	18	
Define the Instantiated PF	19	
Define the Instantiated Process Framework	20	
QA2	21	16
Concretization	22	15
Concretization Iteration [1..n]	23	
Analyze the Instantiated PF	24	
Define the Requirements to Implement in the Runnable Process Framework	25	
Decide Processes to Implement in Software	26	
Define Business Processes to Implement in Software	27	
Define Alternative Designs for Runnable PF Processes	28	
Define Alternative Designs for the Runnable Process Framework	29	
QA3	30	
Implementation	31	22
Implementation Iteration [1..n]	32	
Customize Software Implemented PF	33	
Customize the Software-implemented Process Framework	34	
Develop Software Components for the PF	35	
Develop Software Components for the Process Framework	36	
Provide a Software Implemented PF	37	
Provide the Software-implemented Process Framework	38	
QA4	39	32

Figure A.1.: BIM work breakdown structure.

A. BIM Formalization in EPF

		Delivery Process
⊿ 🔂 BIM Life-cycle		Delivery Process
⊿ 🏠 Selection		Phase
⊿ 🔂 Selection Iteration [1..n]		Iteration
⊿ 📰 Analyze Process RMs		Activity
👥 Business Analyst		Role Descriptor
👥 Client		Role Descriptor
⊿ 📰 Analyze IAvOs		Activity
👥 Business Analyst		Role Descriptor
👥 Client		Role Descriptor
⊿ 📰 Analyze OAs		Activity
👥 Business Analyst		Role Descriptor
👥 Client		Role Descriptor
⊿ 📰 Define Generic Process Framework		Activity
👥 Business Analyst		Role Descriptor
📖 QA1		Milestone
⊿ 🏠 Definition		Phase
⊿ 🔂 Definition Iteration [1..n]		Iteration
⊿ 📰 Define Business Needs		Activity
👥 Business Process Architect		Role Descriptor
👥 Client		Role Descriptor
⊿ 📰 Define the Instantiated PF		Activity
👥 Business Process Architect		Role Descriptor
👥 Generic PF		Role Descriptor
📖 QA2		Milestone
⊿ 🏠 Concretization		Phase
⊿ 🔂 Concretization Iteration [1..n]		Iteration
⊿ 📰 Analyze the Instantiated PF		Activity
👥 Business Process Architect		Role Descriptor
⊿ 📰 Decide Processes to Implement in Software		Activity
👥 Business Process Architect		Role Descriptor
⊿ 📰 Define Alternative Designs for Runnable PF Processes		Activity
👥 Business Process Architect		Role Descriptor
📖 QA3		Milestone
⊿ 🏠 Implementation		Phase
⊿ 🔂 Implementation Iteration [1..n]		Iteration
⊿ 📰 Customize Software Implemented PF		Activity
👥 Business Process Architect		Role Descriptor
⊿ 📰 Develop Software Components for the PF		Activity
👥 Software Engineer		Role Descriptor
⊿ 📰 Provide a Software Implemented PF		Activity
👥 Software Engineer		Role Descriptor
📖 QA4		Milestone

Figure A.2.: BIM team allocation.

Figure A.3.: BIM domains.

Figure A.4.: BIM discipline: Business Requirements.

Discipline: Process Framework Management

Relationships

Tasks	
	• Define the Generic Process Framework
	• Customize the Software-implemented Process Framework
	• Define the Instantiated Process Framework
	• Define the Requirements to Implement in the Runnable Process Framework
	• Provide the Software-implemented Process Framework

Figure A.5.: BIM discipline: Process Framework (PF) Management.

Discipline: Information System

Relationships

Tasks	
	• Define Business Processes to Implement in Software
	• Define Alternative Designs for the Runnable Process Framework

Figure A.6.: BIM discipline: Information System.

Discipline: OBO Management

Relationships

Tasks	
	• Develop Software Components for the Process Framework

Figure A.7.: BIM discipline: OBO Management.

Role sets > BIM Roles

Role Set: BIM Roles

Expand All Sections   Collapse All Sections

Relationships

Roles	
	• Business Analyst
	• Software Engineer
	• Client
	• Business Process Architect
	• Generic PF

Back to top

Figure A.8.: BIM roles.
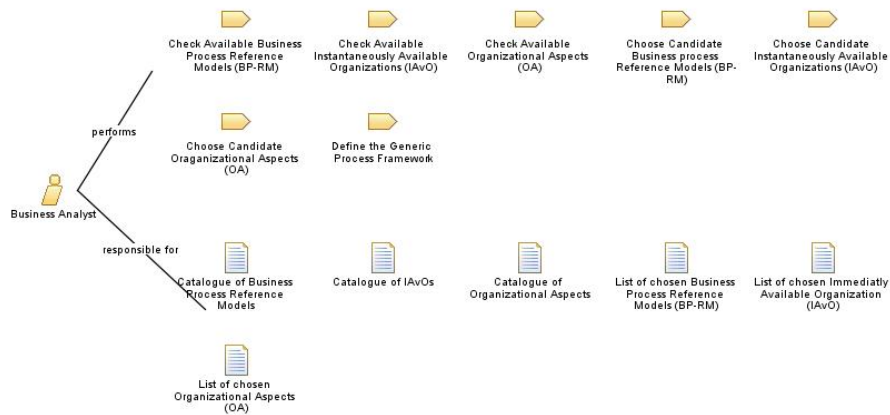
Figure A.9.: BIM role: Business Analyst.

Figure A.10.: BIM role: Software Engineer.

Figure A.11.: BIM role: Client.
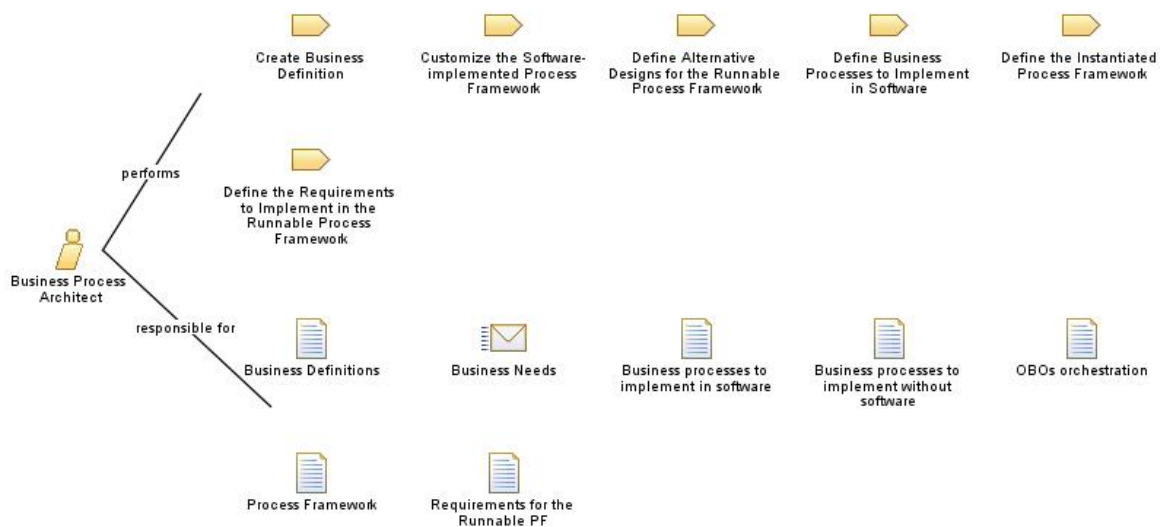
Figure A.12.: BIM role: Business Process Architect.
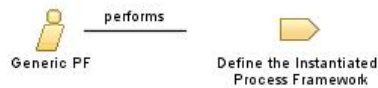
A. BIM Formalization in EPF

Figure A.13.: BIM role: Generic PF.

Figure A.14.: BIM artifacts.

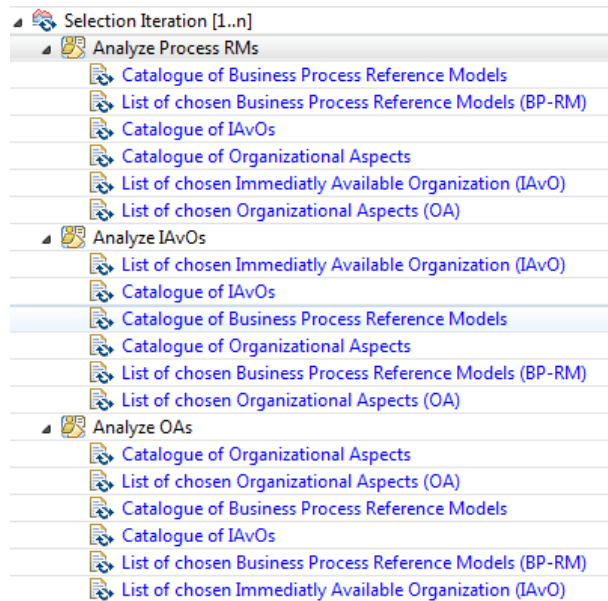Figure A.15.: BIM deliverables.

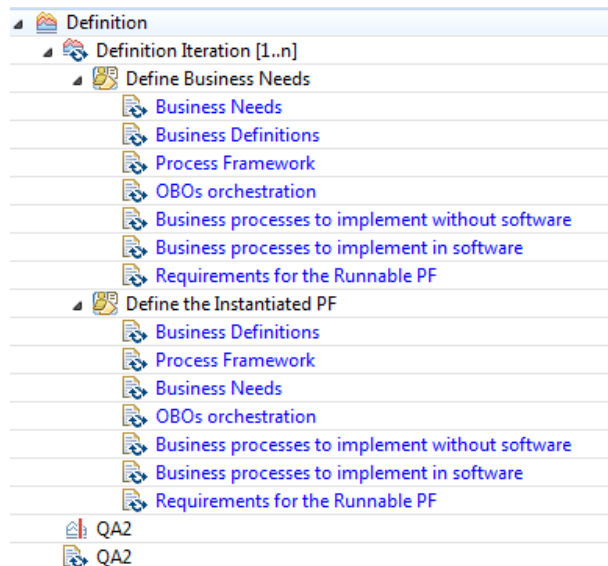Figure A.16.: BIM work product kinds.

Figure A.17.: BIM work product usage in the Selection phase.

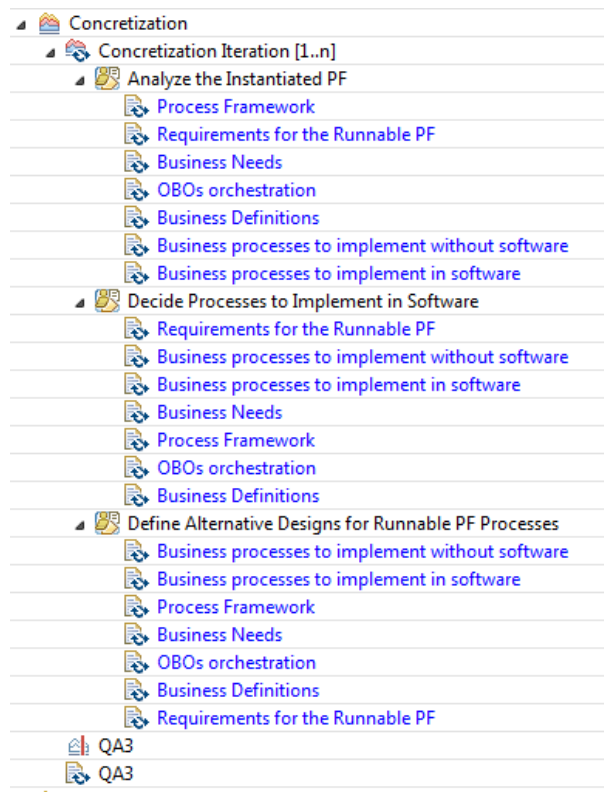Figure A.18.: BIM work product usage in the Definition phase.

Figure A.19.: BIM work product usage in the Concretization phase.

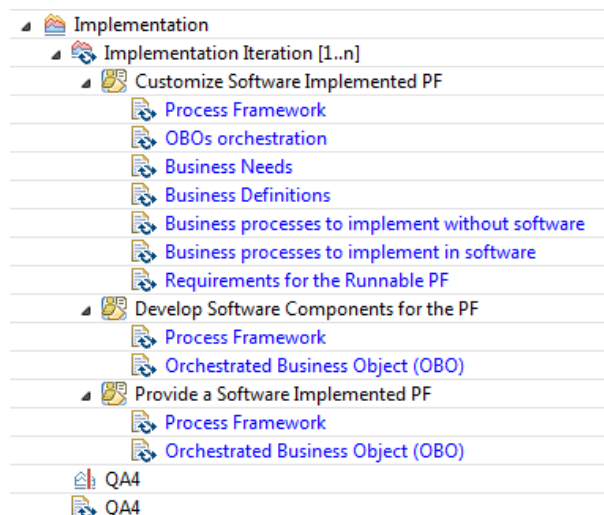

Figure A.20.: BIM work product usage in the Implementation phase.

B. Installing the Main Components of Seppo

The software frameworks used in this thesis are based on Java. For that, a Java Development Kit (JDK) needs to be previously installed.

B.1. Installing Apache SMX 4

Windows installation:

1. Download SMX4 (see Table D.1 to check the web site) and install.
2. Start with "servicemix.bat" at directory ".../bin"
3. At SMX4 console, install web console with the command: "features:install webconsole"

Once the web console is enabled, SMX4 can be accessed via HTTP from a remote machine:

1. In a browser, type: "http://localhost:8181/system/console".
2. Use "smx" as login, and "smx" as password (default values).

SMX4 can group OSGi bundles in "features". SMX4 provides a sub-shell to handle features. For instance, the command "features:list | grep camel" lists all the features in the SMX4 instance that contain the word "camel".

Similarly, the OSGi bundles can be managed via the OSGi sub-shell. The command "osgi:list | grep camel" lists all the bundles that have the word "camel" in the name.

The OSGi bundles can be stopped with the command "osgi:stop nnn", where nnn refers to the number corresponding to the bundle (extracted from the "osgi:list" command). The "osgi:stop" command changes the state from "Active" to "Resolved". The "osgi:start nnn" command changes the state from "Resolved" to "Active".

B.2. Installing Apache ODE in SMX4

To install ODE as a SMX4 OSGi bundle, in the SMX4 console, type:

1. "features:addUrl mvn:org.apache.ode/ode-jbi-karaf/1.3.6/xml/features"

B. Installing the Main Components of Seppo

2. "features:install ode"

The previous commands will install ODE with default settings (OpenJPA DAO, embedded Derby database).

From Apache (2009b): each deployment is a directory with all relevant deployment artifacts. At the minimum it will contain the deployment descriptor, one or more process definitions (BPEL or .cbp), WSDL and XSDs (excluding those compiled into the .cbp). It may also contain other files, such as SVGs or XSLs. The deployment descriptor is a file named deploy.xml (see the next paragraph for its description).

During deployment, the process engine loads all documents from the deployment descriptor. Loading documents allow it to reference processes, service and schema definitions using fully qualified names, and import based on namespaces instead of locations.

To deploy in ODE, just copy the whole directory containing your artifacts (the directory itself, not only its content) in the path %DEPLOYMENT_ROOT%/WEB-INF/processes

B.3. Installing Apache ActiveMQ in SMX4

The SMX4 brings an embedded ActiveMQ JMS broker.

1. To install the ActiveMQ web console, use the SMX4 web console or, alternatively, use the SMX console and type "features:install activemq-web-console"

2. This command will install and start the web console automatically with an embedded broker which you can access going to the following URL: "http://localhost:8181/activemqweb"

B.4. Installing Apache Camel in SMX4

The SMX4 brings embedded the Camel-related bundles. To use the Blueprint XML dialect, use the "features:install camel-blueprint" command to install the feature which includes all the required bundles.

One of the most simple ways to deploy a new route on SMX4, is by defining the route in a Blueprint XML file. The XML file should be copied into the deploy directory under the SMX4 root directory.

In Camel, a route is a step-by-step movement of a message:

1. From a "listening" endpoint in the role of consumer;

2. Through a processing component - EIP, processor, interceptor;

3. To a target endpoint in the role of producer.

In Camel, routes can be specified via different DSLs, such as Spring configuration files, Java, or Scala.

In Camel, endpoints are creators or receivers of messages (e.g. FTP servers, JMS brokers, or web services).

240

C. BPEL Source Code for "Process_D1.8"

```
1   <!-- Process_D1.8 BPEL Process [Generated by the Eclipse BPEL Designer] -->
2   <bpel:process name="Process_D1.8"
3       targetNamespace="Centauro2.0"
4         suppressJoinFailure="yes"
5         xmlns:tns="Centauro2.0"
6         xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/abstract"
7         >
8
9       <!-- Import the client WSDL -->
10      <bpel:import location="Process_D1.8Artifacts.wsdl" namespace="Centauro2.0"
11            importType="http://schemas.xmlsoap.org/wsdl/" />
12
13      <!-- ================================================================= -->
14      <!-- PARTNERLINKS                                                      -->
15      <!-- List of services participating in this BPEL process              -->
16      <!-- ================================================================= -->
17      <bpel:partnerLinks>
18          <!-- The 'client' role represents the requester of this service. -->
19          <bpel:partnerLink name="client"
20                    partnerLinkType="tns:Process_D1.8"
21                    myRole="Process_D1.8Provider"
22                    />
23      </bpel:partnerLinks>
24
25      <!-- ================================================================= -->
26      <!-- VARIABLES                                                         -->
27      <!-- List of messages and XML documents used within this BPEL process -->
28      <!-- ================================================================= -->
29      <bpel:variables>
30          <!-- Reference to the message passed as input during initiation -->
31          <bpel:variable name="input"
32                  messageType="tns:Process_D1.8RequestMessage"/>
33
34          <!--
35            Reference to the message that will be returned to the requester
36            -->
37          <bpel:variable name="output"
38                  messageType="tns:Process_D1.8ResponseMessage"/>
39      </bpel:variables>
40
41      <!-- ================================================================= -->
42      <!-- ORCHESTRATION LOGIC                                               -->
43      <!-- Set of activities coordinating the flow of messages across the    -->
44      <!-- services integrated within this business process                 -->
45      <!-- ================================================================= -->
46      <bpel:sequence name="main">
47
48          <!-- Receive input from requester.
49              Note: This maps to operation defined in Process_D1.8.wsdl
50              -->
51          <bpel:receive name="receiveInput" partnerLink="client"
52                  portType="tns:Process_D1.8"
53                  operation="process" variable="input"
54                  createInstance="yes"/>
55          <bpel:invoke name="OBO_Receive_Container_with_Products"></bpel:invoke>
56
57          <!-- Generate reply to synchronous request -->
58          <bpel:invoke name="OBO_Record_Product_Receipt"></bpel:invoke>
59          <bpel:invoke name="OBO_Determine_Put-away_Location"></bpel:invoke>
60          <bpel:invoke name="OBO_Putting_Away"></bpel:invoke>
61          <bpel:invoke name="OBO_Record_Warehouse_Location"></bpel:invoke>
62          <bpel:reply name="replyOutput"
63                  partnerLink="client"
64                  portType="tns:Process_D1.8"
```

C. BPEL Source Code for "Process_D1.8"

```
65              operation="process"
66              variable="output"
67              />
68        </bpel:sequence>
69   </bpel:process>
```

D. List of Software Tools and Frameworks

Table D.1.: Most important software tools and frameworks used (last version).

Tool	Version	Owner	License	URL	Functionality provided for the project
Lyx	2.0.7	The Lyx team	GPL	http://www.lyx.org	Document processor that encourages an approach to writing based on the structure of the documents and not simply their appearance.
MiKTEX	2.9	Donald Knuth	FSF/Debian	http://www.miktex.org/	Implementation of TEX/LATEX
JabRef	2.10	The JabRef team	GPL	http://jabref.sourceforge.net/	Bibliography reference manager.
Inkscape	0.47	The Inkscape team	GPL	http://www.inkscape.org	SVG image creation and manipulation
Notepad++	5.9.2	Don Ho	GPL	http://www.notepad-plus-plus.org	Text file editing
Eclipse	4.3.2	Eclipse Foundation	Eclipse	http://www.eclipse.org/	Java IDE
Java JDK	1.7	Oracle	Oracle	http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html	Java Development Kit
Apache ServiceMix	4.5.4	Apache	Apache 2.0	http://servicemix.apache.org/	Enterprise Service Bus
Apache Camel	2.13	Apache	Apache 2.0	http://camel.apache.org/	Integration framework based on known EIP
Apache ActiveMQ	5.9.0	Apache	Apache 2.0	http://activemq.apache.org/	Messaging and Integration Patterns server
Apache ODE	1.3.6	Apache	Apache 2.0	http://ode.apache.org/	Engine to execute business processes written following the WS-BPEL standard
Eclipse BPEL Designer	0.5.0	Eclipse Foundation	Eclipse	http://www.eclipse.org/bpel/	BPEL IDE
Hermes JMS	1.14	Colin Crist	FSF	http://www.hermesjms.com/confluence/display/HJMS/Home	Extensible console to interact with JMS providers, such as ActiveMQ
MySQL Community Edition	5.5	Oracle	GNU v2.0	http://www.mysql.com/products/community/	Relational database used in projects
IBM Rational Method Composer	7.1	IBM	IBM	http://www-03.ibm.com/software/products/en/rmc/	RUP tailoring
SOAPUI	3.6.1	Eviware	GNU v2.1	http://sourceforge.net/projects/soapui/files/soapui/3.6.1/soapUI-x32-3_6_1.exe/download	Tool to manage SOAP connections for web services exposed in bundles
CLOC	1.53	alnd	GNU v2.1	http://sourceforge.net/projects/cloc/	Counting Source Lines of Code
Spring	4.0.3	GoPivotal	Apache 2.0	http://projects.spring.io/spring-framework/	Core support for dependency injection, transaction management, web applications, data access, messaging, testing and more.
Visual Paradigm	10. 1 CE	Visual Paradigm	Free for non-commercial use.	http://www.visual-paradigm.com/	Scalable Vector Graphics image creation and UML diagram creation to document the thesis

E. Characterization of Reference Models

E.1. Characterization of the SCOR Reference Model

Table E.1.: SCOR 8.0 reference model characterization (according to (Loos & Fettke, 2010)).

Reference (Primary Reference / Secondary Reference)	Supply-Chain Council: Supply-Chain Operations Reference-model (SCOR) 8.0. Retrieved 14-08, 2007, from http://www.supply-chain.org, 2008. (Holten, R.; Melchert, F.: Das Supply Chain Operations Reference (SCOR)-Modell. In: J. Becker; R. Knackstedt (Eds.): Wissensmanagement mit Referenzmodellen: Konzepte fur die Anwendungssystem- und Organisationsgestaltung (pp. 207-226). Heidelberg: Physica-Verlag, 2002.; Stephens, S.: The Supply Chain Council and the Supply Chain Operations Reference Model. In: Supply Chain Management, 1(2001)1, pp. 9-13.; Sperrle, R.P.: Das SCOR-Modell. Beitrag zur 5. Fachtagung Referenzmodellierung 2001, Dresden. from Foliensatz: http://www.wi.uni-muenster.de/is/Tagung/Ref2001/Folien08.pdf, Abruf am 2005-01-04, 2001.; Huang, S.H.; Sheoran, S.K.; Wang, G.: A review and analysis of supply chain operations reference (SCOR) model. In: Supply Chain Management - An International Journal, 9(2004)1, pp. 23-29.; Soffer, P.; Wand, Y.: Goal-Driven Analysis of Process Model Validity. In: A. Persson; J. Stirna (Hrsg.): Advanced Information Systems Engineering, 16th International Conference, CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings. Berlin, Heidelberg 2004, S. 521-535.)
Criteria	Value
Model Type	Generic Type
Name	SCOR, Supply Chain Operations Reference Model
Origin:	Practice
Responsibility for Modeling (Personal Responsibility / Organizational Responsibility):	(- / Supply Chain Council Inc.)
Access:	Limited
Tool Support:	Yes
Domain:	Function: Supply Chain Management
Modeling Language(s):	Wokflow Diagram, Graphical and Verbal Description
Modeling Framework:	Yes
Number of Diagrams:	24
Number of Views:	1
Process-related Size:	90*
Organization-related Size:	-
Data-related Size:	-
Function-related Size:	-
Output-related Size:	-
Construction Method:	no statement
Evaluation Method:	Critical Argumentation (Holten, Melchert 2002) / (No/No) Listing of Critical Aspects (Sperrle 2001) / (No/No) Analytical Hierarchy Process (AHP) (Huang, Sheoran, Ge Wang 2004) / (Partly/Yes) Validity Analysis Using Bunge's Ontology of an Example Process (Soffer, Wand 2004) / (Yes/Yes)
Application Method(s):	Best Practice and Metrics Description
Reuse and Customization:	Specialization
Use Case(s):	Multiple (Stephens 2001) / (No/No/No) Multiple (Sperrle 2001) / (No/No/No)
Abstract:	

E.2. Characterization of the Fernandes/Duarte Reference Model

Table E.3.: Fernandes/Duarte reference model characterization (according to (Loos & Fettke, 2010)).

Reference (Primary Reference / Secondary Reference)	Fernandes, J. M.; Duarte, F. J.: A reference framework for process-oriented software development organizations. In: Software and Systems Modeling 4 (2005), S. 94-105.
Criteria	Value
Model Type	Generic Type
Name	
Origin:	Science
Responsibility for Modeling (Personal Responsibility / Organizational Responsibility):	Fernandes, J.M.; Duarte, F.J.
Access:	Open
Tool Support:	No
Domain:	Institution: Software Development Organizations
Modeling Language(s):	UML Activity Diagrams
Modeling Framework:	Yes
Number of Diagrams:	1
Number of Views:	1
Process-related Size:	8
Organization-related Size:	-
Data-related Size:	-
Function-related Size:	-
Output-related Size:	-
Construction Method:	Based on the Rational Unified Process (RUP)
Evaluation Method:	Case Study (Fernandes, Duarte 2005) / (No/No)
Application Method(s):	Procedure Model
Reuse and Customization:	no statement
Use Case(s):	1 (Fernandes, Duarte 2005) / (No/Partly/No)
Abstract:	