Towards using BPM Patterns in Requirements Elicitation

by

Mohamed AbdElRazik

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In an increasingly changing environment, different organizations are trying to improve their agility and efficiency by improving their business processes; thus, business process management has been gaining momentum for the last decade. The first step in business process management is the modeling of business processes. Business Process Modeling (BPM), in itself, is very important because it captures business requirements, allows for better understanding of a business and its processes, facilitates communication between business analysts and IT people, and pinpoints deficiencies in processes. It also serves as a basis for automation of these processes. But business process modeling comes with its own challenges since it is a time-consuming, complicated, and error-prone task. As a result, producing a high quality, precise business process model is not easy. BPM patterns, which are general reusable solutions to commonly occurring problems in business process modeling, have been proposed to address these challenges. In this research, we conducted an exploratory study about requirements engineering practices in a large organization. This study identified key challenges in requirements engineering and showed how business process modeling is currently being conducted. Then, we created a survey of the different BPM pattern catalogs existing in the literature. Finally, we presented one of the BPM pattern catalogs in a clear format along with examples of each pattern. The ultimate objective is to allow business analysts to effectively use BPM patterns while creating precise BP models.

Acknowledgements

I would like to thank

Dr. Krzysztof Czarnecki	for being my supervisor and for his valuable guidance and support
Dr. Paul Ward and Dr. Patrick Lam	for being my readers and for their worthy comments and suggestions that helped improve this thesis
Dr. Daniel Berry	for the collaborative work and his valuable guidance
Dr. Michal Antkiewicz	for the collaborative work, support and the friendship
My lab mates	for providing a great atmosphere to work in, for their support, and for the great time and discussions I have had at Generative Software Development Lab
My friends in Canada	for their immense help, support, and all the good times (especially Ahmed Farahat and Ahmed Wasfy)
All the graduate students who voted for me	to represent them at the University Senate and as the Vice President External of the Graduate Students Association. Although these responsibilities consumed much time but they helped me to stay in touch with life outside of my research area and to learn much more about life in academia

Thank you all!

Dedication

I would like to dedicate this thesis to those I love, to my family.
To my parents, my brother and my sister.

Table of Contents

List of Tables ix

List of Figures x

Nomenclature xii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Objectives . . . 2
- 1.3 Research Contribution . . . 2
- 1.4 Thesis Organization . . . 3

2 Background and Related Work 4
- 2.1 Requirements Engineering . . . 4
- 2.2 Business Process Modeling . . . 4
  - 2.2.1 Patterns . . . 5
  - 2.2.2 Business Process Modeling Patterns . . . 5
  - 2.2.3 Disambiguation . . . 6
    - 2.2.3.1 Business Process Patterns . . . 6
    - 2.2.3.2 Process Patterns . . . 6
    - 2.2.3.3 Business Patterns . . . 6
    - 2.2.3.4 Business Model Patterns . . . 7
    - 2.2.3.5 Modeling Patterns . . . 7
  - 2.2.4 Other BPM-related Patterns . . . 7

3 Exploratory Study 9
- 3.1 Introduction . . . 9
- 3.2 Methodology . . . 10
  - 3.2.1 Data collection activities . . . 10
  - 3.2.2 Data processing and analysis activities . . . 12
- 3.3 Results . . . 14
- 3.4 Observations related to BPM . . . 18
- 3.5 Limitations of the exploratory study . . . 18
  - 3.5.1 Threats to external validity . . . 18
  - 3.5.2 Threats to internal validity . . . 19

3.6 Summary . . . 19

4 Survey of BPM Patterns **21**
4.1 Pattern Catalogs . . . 21
4.1.1 Taxonomy . . . 21
4.1.2 Micro Patterns . . . 22
4.1.2.1 Control Flow Patterns . . . 23
4.1.2.2 Data Patterns . . . 25
4.1.2.3 Resource Patterns . . . 26
4.1.2.4 Exception Handling patterns . . . 27
4.1.2.5 Summary . . . 28
4.1.3 Medium Sized Patterns . . . 28
4.1.3.1 Generic Process Fragments . . . 29
4.1.3.2 Domain-Specific Process Fragments . . . 30
4.1.3.3 Compliance Patterns . . . 31
4.1.4 Macro Patterns / Reference Models . . . 32
4.2 Anti-Pattern Catalogs . . . 33
4.2.1 Micro-Anti-Patterns . . . 34
4.2.1.1 Control Flow Anti-Patterns . . . 34
4.2.1.2 Data Flow Anti-Patterns . . . 34
4.2.2 Medium Sized Anti-Patterns . . . 35
4.3 Discussion . . . 36
4.3.1 Dependencies between the Classification Dimensions . . . 36
4.3.2 Micro vs Macro . . . 36
4.3.3 How to make use of the Taxonomy . . . 37
4.3.4 Limitations to the BPM patterns survey and taxonomy . . . 37
4.4 Summary . . . 37

5 BPM Patterns with Examples **39**
5.1 Basic Control-Flow Patterns . . . 39
5.1.1 Sequence pattern . . . 40
5.1.2 Parallel Split (Fork) . . . 41
5.1.3 Synchronization (Join, AND-join) . . . 41
5.1.4 Exclusive Choice (Decision) . . . 42
5.1.5 Simple Merge (XOR-Join, Asynchronous join, Merge) . . . 43
5.1.6 Summary . . . 44
5.2 Advanced Branching and Synchronization Patterns . . . 45
5.2.1 Multi-Choice . . . 45
5.2.2 Structured Synchronizing Merge . . . 45
5.2.3 Multi Merge . . . 46
5.2.4 Structured Discriminator (1-out-of-M-join) . . . 47
5.3 Structural Patterns . . . 48
5.3.1 Arbitrary Cycles . . . 48
5.3.2 Implicit Termination . . . 49
5.4 Multiple Instance Patterns . . . 50

5.4.1 Multiple Instances without Synchronization 50
5.4.2 Multiple Instances with a priori Design-Time Knowledge . . 50
5.4.3 Multiple Instances with a priori Run-Time Knowledge . . . 51
5.4.4 Multiple instances without a priori Run-Time Knowledge . 52
5.4.5 Comparing multiple instances patterns 52
5.5 State-based Patterns 53
5.5.1 Deferred Choice 53
5.5.2 Interleaved Parallel Routing 54
5.5.3 Milestone 54
5.6 Cancellation Patterns 55
5.6.1 Cancel Activity 55
5.6.2 Cancel Case 56
5.7 Summary 57

6 Conclusion **58**
6.1 Summary of Contributions 58
6.2 Recommendations for Future Work 59

Appendix A **60**

References **61**

List of Tables

3.1 Details of data collection activities 10
3.2 Summary of data collection activities 12
3.3 Summary of types and amount of collected data 13

4.1 Papers analyzing support for control-flow patterns in modeling languages 24
4.2 Literature for Micro Pattern Catalogs. 28

List of Figures

3.1 "Requirements engineering is where the informal meets the formal." 16

4.1 Classification Dimensions 21
4.2 Micro Patterns 22
4.3 Control Flow Patterns 23
4.4 Data Patterns 25
4.5 Resource Patterns 26
4.6 Exception Handling Patterns 27
4.7 Medium Sized Patterns 28
4.8 Medium Sized Generic Patterns 29
4.9 Notification Pattern [96] 29
4.10 Medium Sized Domain Specific Patterns 30
4.11 Compliance Patterns 31
4.12 Macro Patterns 32
4.13 Union of all BPM Pattern Catalogs 36
4.14 Relationship between Size, Abstraction Level and Universality . . . 37

5.1 Prepare Coffee/Tea Process 40
5.2 Sequence Pattern 40
5.3 Parallel Split Pattern 41
5.4 Join Pattern 42
5.5 Decision Pattern 43
5.6 Merge Pattern 43
5.7 Graphical representation of basic control flow patterns [47] 44
5.8 Multi-choice pattern (based on an example from [80]) 45
5.9 Structured synchronizing merge (based on an example from [80]) . . 46
5.10 Multi merge pattern (based on an example from [80]) 47
5.11 Structured Discriminator pattern (based on an example from [80]) . 48
5.12 Arbitrary cycles pattern 49
5.13 Implicit Termination pattern 49
5.14 Multiple Instances without synchronization 50
5.15 Multiple Instances with a priori Design-Time Knowledge 51
5.16 Multiple Instances with a priori Run-Time Knowledge 52
5.17 Multiple Instances without a priori Run-Time Knowledge 52
5.18 Deferred choice pattern 54
5.19 Cancel Activity pattern (based on an example from [80]) 55

5.20 Cancel case pattern (based on an example from [80]) 56
5.21 Cancel case pattern with subporcess expanded (based on an example from [80]) 57

Nomenclature

BA Business Analyst

BPM Business Process Modeling

BPMN Business Process Modeling Notation

IT Information Technology

LOVEM Line of Visibility Enterprise Modeling

RE Requirements Engineering

SOA Service Oriented Architecture

Chapter 1

Introduction

Building successful enterprise software requires an effective collaboration among Business and IT stakeholders. Business Analysts (BAs) capture requirements in business terms. Software engineers interpret these requirements and translate them into the domain of software technology. Yet the different backgrounds and skill sets of these two stakeholder groups naturally lead to the proverbial Business-IT divide. The two groups across the divide speak different languages, each with their own terms and semantics. It is not uncommon to see projects fail because of poorly understood and miscommunicated requirements across the Business-IT divide [2].

Business Process Management is a discipline and technology that has been gaining attention and been increasingly adopted over the last decade. Some of the key selling points for such technology are that it helps bridge the Business-IT gap, improves productivity, and minimizes miscommunication. Despite the fact that the market for Business Process Management has been growing, the issues it is trying to address have not yet been resolved. This is not to say Business Process Management is not successful, but many challenges remain. We believe that it has not yet achieved its full potential.

1.1 Motivation

Requirements Engineering (RE) is about defining precise, complete software requirements. Having such software requirements early on in the software lifecycle can save a lot of time, effort, and money. In his article "Software Defect Reduction Top 10 List," Barry Boehm stated that "finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase" [18]. He mentioned this fact as early as 1976, when he said, "Clearly, it pays off to invest effort in finding requirements errors early and correcting them in, say, 1 man-hour rather than waiting to find the error during operations and having to spend 100 man-hours correcting it" [19].

The first motivation for this thesis is to understand how requirements engineering activities are being applied by practitioners in enterprise IT and to identify the challenges they face. Business Process Modeling (BPM) captures requirements of

a business and promotes communication between business and IT people, so that the process model can be automated using an SOA Web Services composition.

One of the challenges faced by BPM is achieving a balance between precision and understandability of a business process model i.e., how much a process model is precise and yet easy to understand by both business and IT people.

To simplify models while keeping precision we can use BPM patterns. BPM patterns are general reusable solutions for commonly occurring problems faced by business modelers. Many different catalogs of BPM patterns exist, but these patterns are still being investigated by the research community and have not been widely adopted. This is the reason that we are proposing an exemplar of some patterns selected from the published pattern catalogs.

1.2 Research Objectives

The main objectives of this research are to identify those challenges faced by practitioners in requirement engineering in enterprise IT in general and those challenges related to business process modeling in particular; to conduct a survey of business process modeling catalogs in literature; to create a taxonomy for these catalogs; to present one of these pattern catalogs in an easy to understand format; and to provide examples that demonstrate each pattern. The ultimate objectives are to draw attention to the need of using business process modeling patterns in practice and to allow business analysts to be able to effectively use business process modeling patterns while creating business process models.

1.3 Research Contribution

This research contributes the following:

- an empirical research study on requirements engineering practices in a large corporation was conducted, resulting in the publication of a technical paper and conference paper [17]. This research study identifies
  - 10 key pain points ranked from highest to lowest priority. These are the challenges faced by Business Analysts in dealing with Requirements Engineering in practice.
  - 5 tool features wish list from Business Analysts' points of view ranked from highest to lowest priority. These are the desired features for requirements management tools.
  - 6 tool features wish list from developers' points of view ranked from highest to lowest priority. These are the desired features for requirements management tools.
  - the challenges associated with the creation and use of business process models in practice.

- the first survey and classification of BPM pattern catalogs in literature were conducted resulting in the submission of a paper;
- an exemplar for one of the BPM pattern catalogs was created.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. In chapter 2, we provide background information about previous work related to this research. In chapter 3, in an exploratory study conducted over a period of nine months, we investigate the current practices of Requirement Engineering (RE) in a large IT department and present our observations. Chapter 4 presents the first survey of many different BPM pattern catalogs in literature. Also this chapter proposes a taxonomy for BPM patterns. Chapter 5 presents the control-flow pattern catalog while providing some exemplars. In chapter 6, we conclude this research by giving an overview of the activities achieved and summarize the research contributions. Then we highlight the limitations of our research followed by recommendations for future work.

Chapter 2

Background and Related Work

2.1 Requirements Engineering

Requirements engineering can be defined as "the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families." [1]

Business process models are considered one of the tools to capture requirements.

2.2 Business Process Modeling

Davenport [25] defines a business process as "a structured and measured set of activities designed to produce a specific output for a particular customer or market". Business process models are formal or semi-formal models of business processes. Usually, visual languages like the Business Process Modeling Notation (BPMN) are used for creating business process models. Such models can support discussion among different stakeholders and help them to get a better understanding of the processes in an organization. Important use cases for business process models are the documentation of existing or planned processes, for example documenting compliance with existing regulations. The models can be analyzed to pinpoint possible improvements of the modeled processes. They can also provide a formal basis for the automation of processes.

Business process models describe different aspects of a business process including

- the functional and control aspect—describing the activities to be carried out and the logical and temporal relations between them;
- the data aspect—describing creation, change, transport and consumption of data;

[1] Zave, P. (1997). "Classification of Research Efforts in Requirements Engineering", ACM Computing Surveys, 29(4): 315-321

- the organizational aspect—describing the organizational roles in an organization and their responsibilities;
- the resources aspect—describing resources that are used, consumed or created during the execution of the process.

The creation of business process models is a manual task that requires expert knowledge both in the domain to be modeled and in the formal business process modeling language. Patterns—documented solutions to recurring problems—can be used to reduce difficulties in creating business process models.

2.2.1 Patterns

The idea of studying commonly occurring problems in a particular domain and documenting general reusable solutions for these problems as patterns can be traced back to the work by Christopher Alexander [6] in the 1970s in the field of architecture. Alexander also introduced the concept of pattern languages, a system of patterns that explains the relationships between them. Although Alexander's work was in the field of architecture, the idea was generic enough to be used in other domains. By the mid 1990s, patterns gained popularity in software engineering as well. In 1994, the Gang-of-Four book [31] introduced object oriented design patterns. It contains a catalog of software design patterns with their names, intent, motivation, applicability, implementation and sample code.

In addition, patterns were used in numerous other areas like software architecture and enterprise application integration. As noted by Barros et al., "patterns have proved invaluable in the reuse of requirements, design and programming knowledge. They were traditionally the province of software design, but have recently emerged in the BPM field" [12].

2.2.2 Business Process Modeling Patterns

We define business process modeling patterns as "general reusable solutions to commonly occurring problems in the area of constructing a formal or semi-formal conceptual business process model". This "general reusable solution" (i.e. the pattern) can be as a small as a construct or a model fragment that solves a "commonly occurring problem", or it can be even as large as a reference model that describes a group of related processes in a certain business domain.

Literature contains several pattern catalogs in the area of BPM. However, to the best of our knowledge, there is no published work that reviews the state of the art in BPM patterns. A researcher who wants to study BPM patterns or a business process modeler who is interested in applying BPM patterns would have to locate and read numerous sources in order to learn about existing pattern catalogs.

Chapter 4 tries to fill in that gap while proposing a taxonomy for different BPM patterns. The taxonomy presented there is a first attempt to formulate a BPM patterns taxonomy.

2.2.3 Disambiguation

In this section, we will compare business process modeling patterns according to our definition with several apparently similar concepts that have been published in the literature.

2.2.3.1 Business Process Patterns

In the existing literature, the term "business process modeling pattern" is not used in a consistent way. For example, Kavanagh [46] and Jung and Sprenger [44] used this term for describing patterns which explain how to develop or improve business processes. In this case, the pattern refers to *the actual process*; an example for such a pattern would be "Avoid manual tasks that can be done by a computer system". Formally, this kind of pattern is different from what we call business process modeling pattern in this article. Business process modeling patterns (according to our definition) support the creation or modification of business process *models* in a (semi-)formal modeling language.

However, in practice it is hard to distinguish between patterns for defining good business processes and patterns for creating good business process models. In particular, if the purpose of modeling is to depict to-be processes instead of just documenting the current as-is situation, this kind of pattern can support the work of a business process modeler very well.

For the sake of clarity, we suggest the use of the term **business process patterns** for patterns that are related to (re)designing business processes (consistently with [13]), because the focus of such pattern is on the business process and not on its model. An overview of such business process patterns (called "best practices" by Reijers and Mansar) can be found in [74]; they are not subject of the survey in this article.

2.2.3.2 Process Patterns

The term process patterns, introduced by Ambler [7] is used by many authors for describing best practices in software engineering processes. Such patterns can be aligned with different methodologies (such as the Rational Unified Process). They cover all issues of software development, including coding, testing, quality assurance, service delivery, documentation, maintenance, etc.

2.2.3.3 Business Patterns

Hruby [41] states that there are patterns for concepts that can be found in almost all business software applications. Such concepts include economic resources, economic agents, economic events, commitments, and contracts. The patterns describing these concepts are called "business patterns" by Hruby. In a similar way, Eriksson and Penker [28] used the term "business pattern" for the general concept of patterns for modeling businesses, from business architecture to processes, business rules, and business goals.

The IBM Patterns for E-Business [4] refer to architectural patterns that help develop web-based applications. In the context of [4], business patterns are defined as "patterns for the interaction between users, businesses, and data."

2.2.3.4 Business Model Patterns

In business science, the term "business model pattern" is used for a typical pattern describing the basic economic principles of an organization. Osterwalder and Pigneur [68] define the term business model as follows: "A business model describes the rationale of how an organization creates, delivers, and captures value." The process aspect is part of this broad definition, but a business model also includes several other aspects (like financial aspects, customer relationship management, management of distribution channels, strategic partnerships etc.) [32]. A well-known example of a business model pattern described in [68] is the "Long Tail" business model pattern which became popular in the area of Internet business. Its focus is on "offering a large number of niche products, each of which sells relatively infrequently".

2.2.3.5 Modeling Patterns

Modeling patterns do not directly refer to models but to the process of creating models. The aim of such modeling patterns is to improve the modeling process; desirable results of such patterns include faster generation of models, using fewer resources and with better quality. An example of a pattern catalog for enterprise modeling is [93].

2.2.4 Other BPM-related Patterns

Several kinds of patterns that are related to BPM have been published. In this thesis, we focus on a special kind of such patterns: *configurable model fragments* from which a business process model in a modeling language like BPMN can be constructed. Other patterns, for example those describing properties of a model, are not the subject of our survey. However, in this section we give a very short overview about existing pattern catalogs.

Quality constraints in a business process model have been compiled as **Process Quality Patterns**. This term has been used by Foerster [30] for behavioral requirements, e.g. "in each execution of the process, a task *Sending Goods* has to be preceded by a task *Payment*". Other authors prefer the terms **Compliance Patterns** [27] or **(Process) Property Specification Patterns** [121] (in line with the work of Dwyer at al. [26]). The special case of security property specification is covered by **Security Patterns** [120].

Patterns for co-occurrence and relationships between activities in a business process model have been called **Action Patterns** in [90].

Patterns dealing with temporal constraints that have to be fulfilled by a business process instance (called **Workflow Time Patterns**) have been compiled in [53].

Thom et al. [113] introduced **Change Patterns** that aim to raise the level of abstraction when applying changes to a business process model.

Service Interaction Patterns [12, 11] define patterns for interaction between web services in a choreography or an orchestration. Barros et al. [12, 11] give an overview of these patterns; a more detailed compendium has been published by van der Aalst et al. [106].

Chapter 3

Exploratory Study

In this chapter, we present an exploratory study that was conducted in a large IT department over a period of 9 months. In section 3.1, we introduce the study; section 3.2 outlines the work achieved; and section 3.3 presents the results of the exploratory study. Section 3.4 presents our observations about business process modeling. We conclude the chapter with section 3.5 where we present the limitations of the study. This chapter is based on the study report [8].

3.1 Introduction

This exploratory study was conducted as a consulting project for an IT department with few thousands employees serving an organization with around 70,000 employees. This study was conducted over 9 months from April 2009 through January 2010, at two sites.

This study [8] focused on Business Analysts (BAs). There were about 800 BAs in the IT department. The study had two main goals: 1. to evaluate requirements engineering practices, and 2. to determine features of requirements gathering and management tools needed to support requirements engineering.

The study had an exploratory nature: rather than studying a particular issue or problem; the intention was to be inclusive and identify as many issues as possible. Throughout the study, we conducted a series of focus groups followed by semi-structured interviews with business analysts (BAs); business systems analysts (BSAs); development personnel (including developers and architects); and Quality Assurance & Testing (QAT) members. All of these belonged to the Information Technology department mentioned earlier. Different stakeholder statements were analyzed qualitatively and coded to find out a set of key findings. The findings include pain points, good practices, requirements engineering process, reusability, process models and a ranked list of desired features for requirements tooling among other things.

3.2 Methodology

The work achieved within this project included data collection, preparation of collected data for coding (e.g. transcribing recordings), conceptualizing and analyzing the data.

3.2.1 Data collection activities

Data collection took place in two different sites (cities), where the industrial partner has its IT department located.

Table 3.1 presents the details of data collection activities.

#	Description	Site
1	on-site observations of full-day JAD session (with 26 participants)	Site 1
1	on-site observations of full-day JAD sessions (with 13 participants)	Site 2
1	focus group with BAs/BSAs	Site 1
1	focus group with BAs/BSAs	Site 2
7	interviews with BAs/BSAs with varied experience	Site 1
3	interviews with BAs/BSAs with varied experience	Site 2
1	focus group with software developers	Site 1
1	focus group with workflow modelers	Site 1
1	focus group with testers	Site 2
8	interviews with non-BAs	Site 1,2

Table 3.1: Details of data collection activities

We first conducted on-site observations of two requirements document review sessions to give us a fresh outsider's view that has not been biased by any other interactions (i.e., focus groups and interviews). Our objective was to observe what happens in practice and to minimize any potential changes in the behavior of those being observed. To that end, the purpose of the observation was not revealed to the observed. We were sent the requirements documents to be studied in advance to provide us with background about the project and the session. During observation of the session, we neither contributed to the discussion nor commented on anything being said. We were passively observing how the meeting was conducted and how different roles interacted. We did some note taking, and we documented our observations immediately after each session in a mind map. We noticed the large scope of the projects being reviewed and the large number of stakeholders involved. For example, our first meeting had 22 participants in the meeting room and 4 additional ones on the phone. The second meeting had 13 participants in the meeting room. The participants had very diverse roles, and each was sharing his or her unique perspective.

We selected study subjects to cover different role perspectives. We conducted focus groups and interviews with BAs, Business System Analysts (BSAs), software

developers, architects, project leaders, and testers. We refer to BAs and BSAs collectively as BSAs. Since our focus was on requirements, we selected more subjects playing the BSA role than other roles. We selected BSAs with differing amounts of experience and from the two development locations. For the roles with fewer subjects, we preferred more experienced individuals.

We categorized BSAs into three different experience categories:

- Category 1: business analysts new to the organization but with business analysis background from outside;
- Category 2: business analysts new to the role with no previous experience in business analysis; and
- Category 3: senior business analysts.

We were presented with a list of potential interviewees and we chose participants to cover each category. Some of the proposed interviewees were unavailable and, in one case, the person did not show up for the interview.

We had a list of questions prepared for the focus groups and the interviews. In addition, we spontaneously asked questions in response to the evolution of the conversation.

We conducted focus groups after the on-site observations. We planned each focus group to take about 2 hours. For each focus group, we prepared a list of topics based on the observations and study goals. For each topic, we listed a few questions to stimulate discussion. At the beginning of each focus group, we introduced ourselves and stated the objective of the study. We encouraged participants to freely and honestly talk about their experiences and practices they apply in their day-to-day work instead of just the practices that are recommended but not performed in reality. To encourage openness and honesty, we promised to remove all references to specific persons and any information that could be used to identify a person. During the actual discussion, we presented a topic together with a list of questions and we moderated the discussion. We allowed the discussion to flow freely beyond the questions we provided and encouraged the participants to talk freely about all the issues related to the goals of the study that they felt were important. We requested and received permission to voice record the session. Recordings were transcribed and coded as described below.

We conducted individual interviews after the focus groups. We ran each interview within a 45 minute slot, planning only 30 minutes for the actual interview. Based on the focus groups, with a fresh recollection and before coding, we prepared a set of questions covering five areas.

The objective of the question set was to stimulate the interview, but we asked any question only if its topic had not yet been covered. Most questions were open-ended with the aim of generating more interaction, and the interviewees were allowed to digress and discuss what they found important. We stated the objective of the study at the beginning of each interview and we encouraged the interviewees to answer the questions based on their personal experiences from the projects in which they participated at the bank.

For all interviews, the sections and questions prepared in advance were the same. During an interview, we focused on questions, or variations thereof, that were relevant to the experience category of the interviewee. For example, when interviewing a senior BSA, the focus was on how the process changed over the years. Based on his or her experiences we asked the following:

- What are the characterizations of a successful project?
- What are the best practices?
- What are the pain points?

When interviewing a BSA with business analysis experience from outside, we tried to find out how the interviewee's experience differs from his or her previous experience. When interviewing a BSA new to the role, we focused on how he or she learned to perform his or her job, e.g., with training or mentoring.

Towards the end of each interview, we presented a proposed list of six features of requirements engineering tools and asked the interviewee to rank the features by priority. Each interviewee was free to add new features to or remove the ones he or she felt were not useful from the list.

In the beginning of each interview, we asked for permission to record the interview. Later, the recordings were transcribed and conceptualized.

After conducting 2 focus groups and 10 interviews with BSAs, we moved to Phase 2, in which we conducted 3 focus groups and 8 interviews with developers. Those activities were conducted in the same way as in Phase 1 with a few differences:

1. the questions were prepared based on the results of analysis of Phase 1, and
2. the set of tool features was changed.

Table 3.2, gives a summary of the data collection activities.

#	Description
2	on-site observations
5	focus groups
18	interviews

Table 3.2: Summary of data collection activities

3.2.2 Data processing and analysis activities

Our main data processing and analysis activities were transcription of recorded focus groups and interviews, taking notes directly from the recordings, conceptualizing the transcripts and notes, analyzing the conceptualizations, and writing the report. This section explains each activity in detail.

Table 3.3 presents a summary of types and amount of collected data.

Type	Amount	Comment
Recordings	23 hours	For all of the 5 focus groups and the 18 interviews
Transcriptions	181 pages	For 2 focus groups and 11 interviews
Notes	42 pages	For 3 focus groups and 7 interviews
Conceptualization	712 features	For 18 interviews and 5 focus groups

Table 3.3: Summary of types and amount of collected data

We chose to transcribe the focus group and interview recordings. Transcription helps in analyzing the data because it eliminates the need to rewind the recordings when different sections need to revisited. We made a fairly precise transcription, leaving out word and sentence repetitions, e.g., "Yes, yes", fill words, or statements off topic. One hour of recording took 5-6 hours to transcribe.

Due to time constraints, in Phase 2 we decided to take notes instead of doing transcriptions. This sped up the process.

We used conceptualization to extract the main concepts (or statements) from each interview or focus group. This technique involves labelling sections of the transcripts. A label, such as "communication- ThroughClientsBAs#150", represents a discrete idea such as "No direct access to the client due to the reorganization that requires to go through client's BAs". We refer to labels as features. Features were structured hierarchically; i.e. some features were categorized under other features. We sometimes introduced a new feature in order to group a number of related features. We kept data in a table with a row being a feature and a column being an individual interview or focus group. We marked the cell at a row and column if the column's interview or focus group raised the point coded by the row's feature. The process of developing the feature hierarchies was iterative. When conceptualizing a transcript, the features and their structures that were created during conceptualizations of previous transcripts were revised to accommodate any newly discovered concepts, and the conceptualizations of the previous transcripts were revised to reflect the new features. The development of features was driven by the content of the interviews and focus groups rather than by predefined goals.

We provided a short explanatory description for each feature. Our intention was to come up with clear and concise descriptions. During conceptualizing, we always searched for an existing feature that accurately represented the transcript statement at hand. If we found a matching feature, we studied the feature's description. If both the feature and the description agreed with the statement, we marked the cell at the feature's row and the transcript's column, effectively saying that the transcript made a statement represented by the feature. If only the feature matched the transcript's statement, we modified the feature's description to cover also the new statement. Otherwise, we simply added a new feature with its own description to cover the new statement. Sometimes, the statements by subjects were imprecise or ambiguous. We did not follow up with clarifications after conceptualization.

Analysis of the conceptualized transcripts is where trends and consensus can be observed. The conceptualization reveals what issues are common or different among two or more transcripts. We used a simple quantitative analysis for this purpose.

For each feature, we provided the total number of interviewees making the point coded by the feature. We provided also a breakdown for each experience category of interviewees based on the category members' experiences. Finally, we provided the total support for each feature, defined as the sum of the number of interviewees making the point coded by the feature and the number of focus groups making the same point. We adjusted the sum to account for possible overlap between the interviewees and focus group members. If an interviewee made a point and a focus group that included the same interviewee as a participant made the same point, we subtracted one from the sum, i.e., effectively not counting the second contribution of the overlapping focus group. Also, we used the same weight for each interviewee and focus group, as we do not see any objective way to assign differing weights. Consequently, a higher support value for a feature indicates a greater consensus among the subjects for the given feature. This analysis approach can be seen as a form of data triangulation, attempting to confirm findings via multiple sources.

Towards the end of each interview, we presented a list of 6 proposed features of requirements engineering tools and asked the interviewee to prioritize the proposed features. The interviewees were free to add new features or remove the ones they felt were not useful from the list. Each interviewee ranked these features starting from 1 as the most important from his or her viewpoint. One interviewee added one feature and another interviewee added two. Thus, the total number of features is now 9. When doing the conceptualization for this part, we gave a feature that was considered the most important by the interviewee the value 9; we gave 8 for the second most important feature; etc. The features that were crossed out were given the value of 0. Based on this conceptualization, the feature with the highest total is the one that was considered by the interviewees the most important feature for a requirements engineering tool.

3.3 Results

In this section we discuss the main issues that we found by analyzing the collected data.

The findings are ordered first by the cumulative support indicated by "..a" (total number of activities that all features mentioned in a given issue were stated in) and second by the total number of features mentioned in the issue indicated by "..f". For example, issues 6 and 7 have the same cumulative support (13) but different number of features (19 and 17, respectively).

We found ten key pain points:

1. Inconsistent methods (20a, 22f)
2. Fragmentation of information over documents and lack of traceability (20a, 22f)
3. Lack of BA/BSA/Client training (19a, 25f)
4. Not enough time for business analysis (17a, 18f)

5. Lack of timely communication between BAs and stakeholders from Development and QAT (14a, 19f)

6. Best practices developed through long-term relationships among some client, BAs/BSAs, Development, and QAT project teams are not disseminated to other project teams (13a, 20f)

7. Communication via client's BAs (13a, 17f)

8. Lack of enterprise-wide process and system documentation (12a, 17f)

9. Challenges with document lifecycle and change management (10a, 11f)

10. Testers not having enough time for testing (1a, 5f)

At a first glance these issues seems not related to BPM and patterns, but we have to remember that the objective of the study was broad explaining the breadth of the findings. For example, finding number 1 "Inconsistent methods" and number 3 "Lack of BA/BSA/Client training" highlight the need for method consistency and training, which are highly connected, since we cannot expect to have a consistent method without training.

With a closer look at our findings, we can find that BPM in general and BPM patterns can help address, at least partially, some of these findings. For example finding 4 "Not enough time for business analysis" and finding number 5 "Lack of timely communication between BAs and stakeholders from Development and QAT" can be directly addressed by BPM patterns assuming that using BPM patterns will save time while improving communication.

Additionally, we cross checked the findings with the Business Analysis Community of Practice (BA CoP) at the organization and they stated that

1. we did not miss any major issues,

2. we were correct with the findings, and

3. they perceived the report as valuable.

We identified a wish list of 5 tool features (from BAs point of view)

1. recording and linking to rationale,

2. support for search-based and explicit traceability,

3. non-invasive document-template conformance,

4. document versioning, and

5. support for sign off.

Also we identified 6 tool features wish list (from developers point of view)

1. repository of business rules, regulations, and conceptual models,

2. recording and linking to rationale,
3. support for search-based traceability,
4. screen mockups,
5. explicit traceability, and
6. document versioning.

From what we have heard in the focus groups and interviews, management seems to be operating under the assumption that it can control how much Requirements Analysis (RA) is being done. The reality is that there is no escaping doing enough RA to write the code, and if not enough time has been allocated to allow the BSAs to do the RA before writing some Requirements Specification (RS) — generally in the form of an External Design Document (EDD) — to be given to the developers and testers, then the developers and testers will do the additional RA as they do their jobs. In other words, there is always enough time to do the RA. The project makes enough time whether management has allocated it or not.

Michael Jackson [43] once said that "Requirements engineering is where the informal meets the formal." Figure 3.1 shows a project's time line, with the raw client ideas on the far left and the code and test cases on the far right. A test case is an input and its expected output.

Figure 3.1: "Requirements engineering is where the informal meets the formal."

Client ideas are necessarily informal. They may not have even been put into words. They are just thoughts. Code and test cases are necessarily formal, not just because a program is as formal a language as any mathematical notation, but also because the meaning of any utterance in code and in input data for a program is fully defined, in the first case, by what the machine code generated by the compiler does, and in the second case, by the output generated by the executing program reading the input.

Somewhere along the project time line, the informal client ideas must be converted into something formal enough that the code and test cases can be written. Short of not writing any code or test cases, this conversion is inescapable. Note that to do this conversion, the requirements for the code must be understood. One cannot write code if he does not know what the code is supposed to do. One cannot write a test case, without knowing what output to expect from the code's reading its input. We can therefore understand that Jackson quote is saying that determining requirements at some point is unavoidable.

There are two extremes about when this conversion happens:

1. at the far right, during coding and test case writing. Each developer determines for herself what the code does at each point in the coding that a decision about the code's behavior must be made. Each test-case writer determines for himself what the expected output is for the considered input. In this case, the code and test cases end up being the RS.

2. at the far left, when the client is presenting her raw ideas to BAs. The client and BAs discuss the system thoroughly, with the BSs asking the client what she means any time they do not understand what they have heard so far. Not understanding includes the BAs' realization that there are details missing from what they understand. The output that the BAs produce is the RS expressing as precisely as possible the requirements for the system that they believe the client requires. Ideally, the developers should be able to write the code directly from the RS without having to make any requirement decisions, and the test-case writers should be able to write each of a covering set of test cases from the RS without having to make any requirement decisions. Moreover, some times, the developers are in a distant place, such as India, making direct communication with the client even more difficult and making the completeness of the RS all the more essential.

In between these extremes, there are countless other places in which some project personnel may make a requirements decision to fill in on details that are not captured in whatever RS is generated by the BSAs after whatever RA they have managed to do.

Thus, no matter what, if client ideas have been converted to code and test cases, requirements had to have been determined.

The veracity of this observation is independent of the lifecycle model that is being used:

1. Clearly the observation holds in a non-iterative model.

2. In an iterative model, the observation holds both over the total sequence of iterations and over each iteration.

The key is that whatever lifecycle is chosen, its RE must be allowed to run its course [17], and the client must be available for consultation the whole time of RE. If a waterfall lifecycle is followed, then if RE is not allowed to run its course, then RD continues anyway, but

1. the wrong people do the RD,

2. they do not have access to the client, and

3. the newly discovered requirements are called creep when they are really only the requirements that were not found by the time RE was cut short.

If an Agile or iterative lifecycle is chosen, then the development must have a scope that is small enough that the client can be readily available for questioning during the whole development.

3.4 Observations related to BPM

In this section we focus on our observations related to BPM.

BAs use a charting tool (e.g. Visio) or a presentation tool (e.g. Powerpoint) to capture BP models. Although such tools gives a great deal of flexibility they know nothing about the semantics of the BP model. In addition, such tools create informal BP models that do not adhere to any formal language.

Since management became aware of this problem, there was an initiative few years ago to introduce a commercial BP modeling tool. That initiative totally failed. The tool was not simple and intuitive enough for BAs. Not only did the average BA felt that the tool was designed for the technical people but even BAs with many years of experience and intimate knowledge of BPM didn't like using it.

We have noticed that BAs, when asked about BPM as a technology and concepts they mentioned that they did not want to use it. But with investigation, we discovered that, due to market hype they are confused. They do not differentiate between the concepts and its implementation in the tools. This became clear when we understand that almost all BAs use informal BP modeling but under another name (they use a very lightweight version of the Line of Visibility Enterprise Modeling (LOVEM) methodology [72]). And they consider business process modeling as a very valuable tool to help them understand/capture the business requirements and as a means of documentation to help transfer the requirements to the IT people.

Also we have noticed that BAs do not use a standard graphical notation. And whenever they need to model something new for them they invent new constructs on the fly or they describe it textually.

One more very important observation is that both BAs and IT people were not aware of the existence of BPM patterns. Consequently they have never tried to use BPM patterns. Another problem is that they do not use a tool that understands the semantics of BP models. On top of this the tools available in the industry are either still lacking or have very limited support to BPM patterns.

3.5 Limitations of the exploratory study

The empirical research study conducted has some threats to validity.

3.5.1 Threats to external validity

External validity is the extent to which the findings can be generalized to the whole organization where the study was conducted and other organizations.

The study was exploratory, designed to cover a broad range of issues and uncover details related to the goals of the study and at the same time the study was conducted in a very large organization with a large number of applications (systems) and development groups. Since we could meet only a limited number of people from a few teams, this has an impact on the external validity of the study. So we cannot claim any completeness, that is, important issues may have been

missed, simply because we did not meet any people who would have told us about them. Additionally, the data and findings may be biased towards the practices of the groups from which the study participants originated.

However, to improve the external validity of the study:

1. we performed the activities in two different sites to cover practices from different work environments,

2. we met with people who use both of the two main development methodologies in the organization,

3. we interviewed BAs who span the 3 categories described in the methods section; unfortunately, only 1 BA from category 1 was available for interview,

4. we conducted the focus group with software developers in one site and the interviews and the focus group with testers in the other site, and

5. we cross checked the report with the Business Analysis Community of Practice, which is a set of senior BAs from different units of the company.

3.5.2 Threats to internal validity

Internal validity is the extent to which the execution of the study results in correct and complete data and reliable findings.

One threat to internal validity is related to the data collection and processing activities: multiple translations. Data collection involved transcribing audio recordings from the interviews and focus groups and some information was lost in the process. Although we strove to be as exact as possible, we did not perform the transcriptions word for word, especially in the focus groups in which multiple people often spoke at the same time. We strove to preserve the actual meaning of the spoken word rather than to achieve exact representation. In Phase 2, we wrote down notes directly from the recordings instead of doing full transcriptions. This threat can be addressed by having two or more researchers verify each other's transcripts or notes.

Another threat to internal validity is related to calculating the support. In some cases, the support may be higher than in reality due to the fact that some of the focus group participants participated also in interviews, and they might have stated the same feature twice, increasing its support by 1. To address this threat we adjusted the total support.

3.6 Summary

In this chapter, we presented an exploratory study that was conducted in a large IT department for 9 months. After introducing the study, we explain the work achieved then presented the results of the exploratory study. Then we present what we observed in relation to business process modeling. We conclude the chapter with

a section where we present the limitations of the study. In the next chapter, we present a survey of different BPM pattern catalogs found in literature.

Chapter 4

Survey of BPM Patterns

This chapter presents a survey of different BPM pattern catalogs that exist in the literature. In section 4.1, we start by proposing a taxonomy for BPM patterns. Then we present different catalogs namely micro, medium-sized and macro pattern catalogs based on the taxonomy proposed. In section 4.2 we present the micro, medium anti-pattern catalogs (solutions to common problems that are known to have drawbacks).

4.1 Pattern Catalogs

4.1.1 Taxonomy

We identified three important dimensions for classifying BPM patterns (see Fig. 4.1).

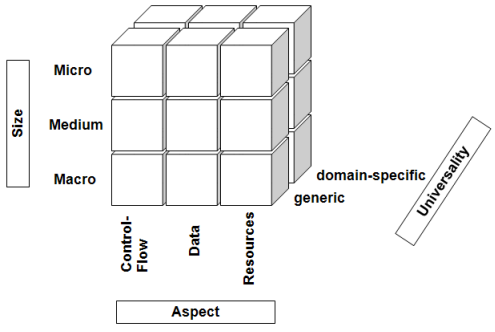

Figure 4.1: Classification Dimensions

Size The size of BPM patterns ranges from single constructs of a modeling language to complete reference models that serve as a blueprint for modeling all relevant business processes in some domain.

We differentiate between micro patterns (a construct or a small process fragment that is typically very generic), medium sized patterns (reusable model fragments)

and macro patterns (a group of process models that can be adapted to specific needs).

Medium and macro patterns do not only have to include a business process model in a language like BPMN. They can also include other elements like metrics for measuring business processes, a domain object model which assists the construction of software for the business processes [76, 70], etc.

Process Aspect As discussed in chapter. 2, business process models contain information about different aspects. We differentiate between patterns that deal with the control-flow aspect, patterns that deal with the data aspect and patterns that deal with the resources aspect (which includes human resources and hence the organizational aspect as well).

Universality While some patterns are specific to a certain business domain, others can be used universally and independently from a business domain. For this reason, we differentiate between domain-specific and non domain-specific patterns.

In the following sections, we will use the size as the main classification.

4.1.2 Micro Patterns

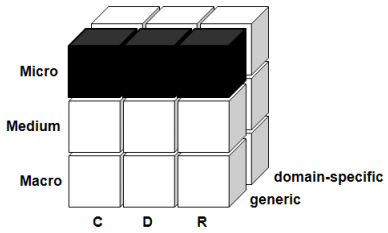

Figure 4.2: Micro Patterns

We categorize the published micro pattern catalogs (see Fig. 4.2) according to the aspect of BPM that they are concerned with:

- **Control flow patterns** deal with the invocation of activities and the temporal order between them.
- **Data patterns** deal with the various ways to represent, transfer and use data.
- **Resource patterns** deal with human and non-human resources.
- **Exception handling patterns** deal with different exception handling capabilities. When an exception has to be handled, both the control flow and the allocation of resources are affected. Therefore, these patterns have to cover the control flow aspect as well as the resource aspect.

4.1.2.1 Control Flow Patterns

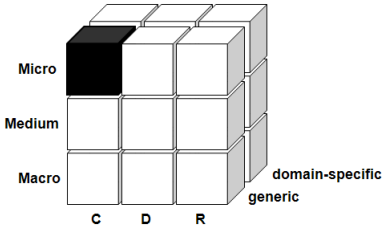

Figure 4.3: Control Flow Patterns

Other Names As the control flow was the first aspect of workflow / BPM languages for which a pattern catalog has been published, some early papers refer to control flow patterns as "workflow patterns". In newer publications, the term "workflow patterns" is used for all four kinds of micro-patterns.

Purpose Control-flow patterns describe the control flow perspective of workflow systems or business process models. The control-flow perspective is concerned with the invocation of activities and the temporal order between activities.

Example The pattern "Exclusive Choice" defines a point in the workflow (or business process model) where—based on a decision or based on data—one of several branches is chosen.

Important Papers Kiepuszewski's thesis [47] is the founding work for control flow patterns and one of the earliest works on BPM patterns in general. It contains 20 control flow patterns. Later this work was expanded to include 43 patterns [80]. In Russell et al.'s paper [80], it has been shown that 32 out of the 43 control flow patterns are either a specialization or a composition of another pattern. By discussing these relations among the patterns, the pattern catalog evolves to a *language* of control flow patterns, giving a clear picture of how the different control flow patterns are related to each other. A formal conceptual basis for building such a language of control-flow patterns has been given by Börger [21].

Applications The control flow patterns described in Kiepuszewski's PhD thesis [47] have been successfully used for a comparison of commercially available and open source workflow management systems [108, 119]. By analyzing which control flow patterns are supported by the different tools, it was possible to draw conclusions on the suitability and the expressive power of the workflow management systems.

In the same way, business process modeling languages have been analyzed with regard to their support for control flow patterns as shown in Table 4.1. Such an analysis discusses the *expressiveness* of a modeling language. It is useful for deciding

whether a language can be used for a given purpose. However, it should be noted that for such an analysis other aspects (for example understandability, modularity or extendability) that are outside the scope of BPM pattern systems have to be considered as well.

In addition to serving as a framework for analyzing existing modeling languages, control flow patterns have also been used as a base for designing the language YAWL [107]. This workflow definition language was constructed such that it supports all major control flow patterns (which is usually not the case for other BPM languages).

The following table gives an overview of papers that evaluate process-based service composition languages and business process/workflow modeling languages with respect to their support for the control flow patterns.

modeling language	papers
BPEL4WS	[116, 103, 117]
BPMN	[104, 114]
Event-Driven Process Chains	[63]
Pi-Calculus	[71]
UML Activity Diagrams	[114, 118]
WSCI	[104]
XLANG	[103]
XPDL	[103, 107]
WSFL	[103]
YAWL	[118, 105]

Table 4.1: Papers analyzing support for control-flow patterns in modeling languages

All applications of control flow patterns discussed so far have the purpose to compare, select or design languages or tools.

Another application of control flow patterns is to integrate support for frequently used control flow patterns into the modeling tool. Gschwind et al. [37], researchers from the IBM Zurich Research Lab, report on an extension they have built on top of a commercial business process modeling tool. This extension helps the modeler to select and use modeling fragments that form frequently used control-flow patterns. The authors state that "many time-consuming editing operations can be replaced by a single click". Mazanek and Minas [61] discussed similar ideas.

4.1.2.2 Data Patterns

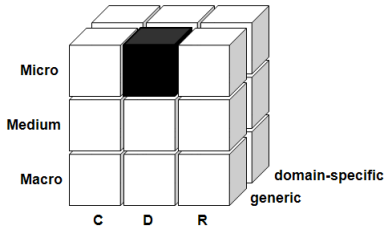

Figure 4.4: Data Patterns

Purpose Data patterns describe the data-flow perspective of workflow systems or business process models. The data-flow perspective is concerned with the various ways in which data is represented and used.

Example The pattern "Data Transfer by Reference—with Lock" describes the ability of a workflow (or BPM system) to communicate data elements between different components by passing a reference to where the location of the data element exists in some mutually accessible location [81, 82]. In order to ensure that only one activity can write the data at any point of time, the receiving component gets read-only or restricted access to the data element.

Important Papers Data patterns have been published by Russel et al. [81, 82]. The paper titled "Workflow Data Patterns: Identification, Representation and Tool Support"[82] provides a general overview while the technical report titled "Workflow Data Patterns" [81] explains all data patterns in detail. Altogether, the researchers identified 40 data patterns based on common characteristics that occur repeatedly in different workflow modeling paradigms. They categorized these characteristics into four distinct groups: data visibility, data interaction, data transfer, and data-based routing.

Applications As with control flow patterns, data patterns have been used for a comparison of both workflow management systems and modeling languages. Russell et al. [82] conducted a detailed review of six workflow systems, standards and web service composition languages with respect to their support for data patterns. Wohed et al. [115] analyzed the support of data patterns in three frequently used BPM languages (BPMN, UML Activity Diagrams and Oracle BPEL PM).

4.1.2.3 Resource Patterns

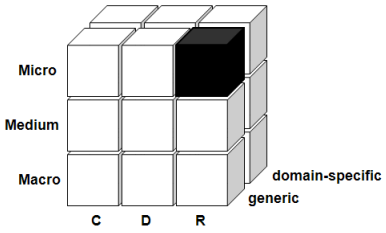

Figure 4.5: Resource Patterns

Purpose Resource patterns focus on the resource perspective, i.e., the people and machines actually doing the work.

Resources are required to execute activities; the definition given in [84] states that "a resource is an entity that is capable of doing work and can be classified as either human or non-human, i.e., a resource that does not correspond to an actual person - e.g. plant and equipment". It should be noted that this definition is much narrower than other definitions of resources used in business science (which would include for example natural and financial resources).

Example In the Shortest Queue pattern in [84], the system allocates the work item to the resource with the least number of work items already allocated to it; for example a heart bypass procedure is allocated to a surgeon with the least number of operations allocated to him/her.

Important Papers Russel, ter Hofstede and Edmond [84] give a brief overview of resource patterns; a complete and detailed review can be found in their paper titled "Workflow Resource Patterns" [83]. 43 patterns have been identified and grouped into a series of specific categories depending on the specific focus of each pattern. Both of the papers [84, 83] focused on human resources, although many of the concepts and patterns can be applied to non-human resources as well.

Details of a particular pattern [84], the "Resource Delegation" pattern, are discussed by Hsu and Wang [42], who also suggest a catalog of specific **delegation patterns**.

Applications The Resource patterns papers [83, 84] examine five commercial workflow systems in terms of support for resource patterns. We are not aware of a comparison of modeling languages with respect to support of resource patterns. However current languages offer rather limited support for modeling the resource perspective. Factors like skills or availability of resources are not included in current BPM languages. For example, originally the language BPEL4WS was lacking any direct support for resources. The standard BPEL4People [5] tries to resolve this

deficiency. Russel and van der Aalst [79, 87] evaluated the proposed BPEL4People standard in terms of the workflow resource patterns.

4.1.2.4 Exception Handling patterns

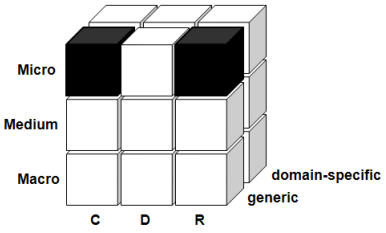

Figure 4.6: Exception Handling Patterns

Purpose A business process model should not only contain information about a "normal" execution of a process instance, it should also define how exceptional cases can be handled. As in software programming languages, such cases are named exceptions. Exception handling patterns deal with the different cases that may lead to exceptions during the execution of a business process and the various ways in which a workflow system can handle such exceptions.

Example A simple example for an exception handling pattern is the pattern called "Resource Unavailable Handling Strategy" [85]. In case of the unavailability of a human resource that is required for some activity, the current work item is suspended, the activity is reassigned to another person and then restarted from the beginning.

Important Papers Russel, van der Aalst and ter Hofstede provided a pattern-based classification framework for workflow exception handling in in [86]. A follow-up paper [85] includes evaluation details on the support of these patterns by several workflow systems and BPM languages.

Staudt Lerner et al. [55] complained that Exception handling patterns papers [86, 85] remain on a rather technical level and do not guide the modeler to choose the pattern that fits a given problem. Staudt Lerner [55] fills this gap by focusing on common problems and structuring the exception-handling pattern catalog such that a modeler can find an appropriate pattern for a given situation.

Applications In Russell et al. [85], eight workflow systems and the business process modeling languages XPDL, BPEL and BPMN have been examined for their support to exception patterns. Staudt [55] discusses the exception handling mechanisms that are included in UML 2.0 Activity Diagrams, BPMN, and LittleJIL and analyzes how often the patterns can be found in a collection of real-world models.

4.1.2.5 Summary

Table 4.2 shows the four categories of micro patterns with the key papers introducing each category.

Category	Key papers	
	Introduction and Overview	Detailed Description
Control flow Patterns	[108]	[80]
Data Patterns	[82]	[81]
Resource Patterns	[84]	[83]
Exception handling Patterns	[86, 55]	[85]

Table 4.2: Literature for Micro Pattern Catalogs.

4.1.3 Medium Sized Patterns

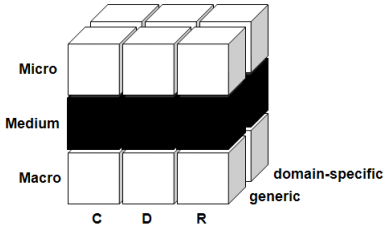

Figure 4.7: Medium Sized Patterns

Medium sized patterns describe generic solutions that can be used inside several business process models. They include more than one activity, but are smaller than a complete business process model. Medium sized patterns describe modeling of common situations like the "four-eyes principle" that can be found in various business process models. They potentially include all aspects of a business process model (functions, control flow, data, organization and resources), although not all of these aspects have been considered in published pattern catalogs.

We categorize the medium sized patterns according to their generality and differentiate between generic (non domain-specific) process fragments and domain-specific fragments. Also, we discuss the special case of compliance patterns that can be either domain-specific or not.

4.1.3.1 Generic Process Fragments

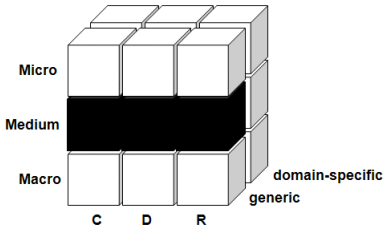

Figure 4.8: Medium Sized Generic Patterns

Purpose Generic (non domain-specific) process fragments describe the modeling of common situations that can occur in different application domains.

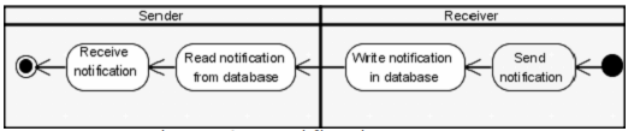

Figure 4.9: Notification Pattern [96]

Example As an example for a medium sized pattern, Fig. 4.9 shows the pattern "Notification" from [96], depicted in the Business Process Modeling Notation.

Important Papers Thom et al. [95] investigate the frequency with which certain non domain-specific patterns occur in practice. They have mined 190 workflow processes in more than 10 different organizations and claim that their set of patterns was necessary and sufficient to design all 190 real workflows. Thom et al.[96] show a set of requirements for process modeling tools such that these tools can support pattern reuse directly. Thom in her PhD thesis [97] presents a metamodel for business process based on patterns which can be used to create new business process models through reuse.

Motschnig al. [76] have compiled a small catalog of patterns for situations that can often be found in a business environment. The names of the patterns already give a good idea of their scope and include *Identify name/location*, *Publishing*, *Validate*, *Voting*, *Make an appointment*, *Register*, *Survey,* and *Meeting*. Their pattern catalog does not only consider the BPM perspective but also includes a domain object model which helps to construct the software that supports the business processes. A similar set of patterns—focusing on the domain object model—can be found in Paludo et al. [70].

Ould [69] describes seven large-scale patterns of organizational behavior such as delegation, reporting and contracts. They allow to model processes in the large.

Lonchamp [57] compiled a list of patterns for collaborative work with the focus on both the control-flow and the data perspective. These patterns deal with the collaborative production of documents and discuss topics like merging or reviewing documents.

Patterns that are related to modeling aspects of business contracts have been presented in Kabilan paper[45].

Applications Thom [97] describes the process of using a set of patterns for creating business process models. Thom also describes how her patterns can be mapped to formal BPEL4WS models, although we are not aware of a tool actually performing such a mapping.

Use of non domain-specific patterns has been shown to be effective in the analysis phase of the software development life cycle. Paludo et al. [70] presented a case study based on real-world data and demonstrated that a large part of the domain object model could be built by using patterns from their catalog.

4.1.3.2 Domain-Specific Process Fragments

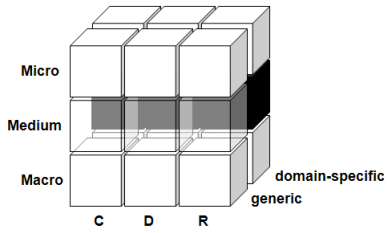

Figure 4.10: Medium Sized Domain Specific Patterns

Purpose Large organizations use hundreds or even thousands of diagrams to document their processes. It is very common that many such diagrams share the same building blocks. Reusing those building blocks can help to create business process models faster and in a more consistent way. Models can be created in a systematic fashion, and as a result the collection of business process models becomes easier to understand.

Example Stephenson and Bandara [92] have published a pattern from the healthcare domain which they call "Continuum of Care". The pattern documents the high-level process "delivery of health treatment services". It has been used for creating process models in various service areas such as radiation therapy, haematology and surgery.

Important Papers Kim et al. [48] describe a modeling mechanism using case-based reasoning for finding fragments that can be integrated into a business process

model. Partially matched models that have been retrieved from a repository are customized to create a new model.

Madhusudan [58] et al. introduced a similar approach that makes use of model fragments stored in a repository. However, instead of relying only on models that have been previously created in the same organization, their approach also works with prototypical patterns - templates that need to be adapted to a business process model instance. Madhusudan et al. state that their repository contains 60 such patterns of various organizational business processes. Their collection of patterns (called "prototypical cases" in [58]) has not been published. The article also contains a discussion of the implications of such a pattern-based approach on the business process model design cycle.

The modeling language PICTURE [16] is built on the concept of domain-specific medium sized patterns. It contains modeling constructs which are specifically designed for a domain. Examples for such constructs are “enter data into IT” or “Approve Document”. The supporters of PICTURE claim that it helps to simplify modeling significantly. Domain-specific versions of PICTURE have been developed for administrative processes [16] and for the banking sector [15].

Applications Domain-specific process fragments can be used inside an organization in order to avoid repeated work and inconsistencies among different but related models [48, 58].

Becker et al [16] argue that their pattern-based modeling language PICTURE allows to create models more easily. Because a PICTURE model contains domain knowledge in a semi-formalized notation, it can also be used for measurement purposes and for detecting processes that should be improved.

4.1.3.3 Compliance Patterns

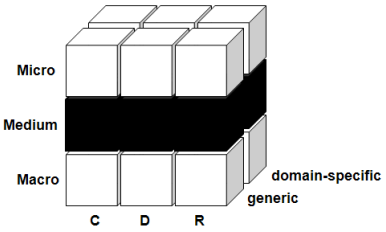

Figure 4.11: Compliance Patterns

Purpose Organizations have to cope with several requirements originating from regulations and standards. Regulations such as the Sarbanes-Oxley act impose compliance requirements that need to be reflected in the business process models of an organization. Compliance patterns are reusable model fragments that are known to adhere to the compliance requirements. Due to the complexity of the existing

regulations, these patterns deal with several aspects of BPM (including control-flow, the resources and decisions based on data values). Compliance patterns can be domain-specific or not.

Example A typical compliance pattern is the "Approval" pattern; based on certain conditions, a decision made by one participant needs to be approved by another one.

Important Papers Namiri and Stojanovic [65] have compiled two sets of patterns which they call "control patterns". Their high level control patterns can be used by compliance experts who are familiar with the domain. The system level control patterns represent a technical view of the former. Schumm et al. [88] present so-called "process fragments for compliance" that can be reused in several models.

Both mentioned papers do not only provide an overview of compliance patterns, they also present a concept that supports the whole life-cycle of a pattern. In particular, they make use of a pattern repository that is used for storage and retrieval of compliance patterns.

Applications Compliance patterns can be used for constructing business process models in highly-regulated domains like banking or healthcare. The patterns can be used not only for constructing compliant models, but also for detecting violations of the compliance rules during the actual execution of a process. An important area of application for compliance patterns are patterns related to security and authorization. Such patterns are discussed in [23] and (on a higher abstraction level) in [3].

4.1.4 Macro Patterns / Reference Models

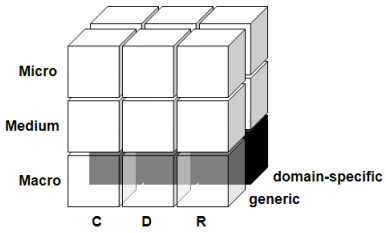

Figure 4.12: Macro Patterns

Other Names Macro patterns are also called reference models, business process frameworks or universal models.

Purpose Macro Patterns serve as a blueprint for the construction of business process models in a certain domain. Usually such patterns are available in the form of generic templates that need to be adapted to the circumstances in an organization. They include standard descriptions of processes, covering all process aspects (functions, control flow, data, organization and resources).

Example The SAP R/3 reference model [24] describes the capabilities of the SAP R/3 system from a business viewpoint, containing both a process model as a data model. The SAP R/3 reference model contains process models in 29 categories, such as Production, Treasury, Real Estate Management and Purchasing.

Important Papers One of the first approaches towards a macro pattern for business process models is the MIT Process Handbook [59]. It contains process descriptions in structured natural language and documents the dependencies between the processes.

A survey and classification of reference models for business processes (containing mainly reference model published in Germany) has been published by Fettke et al. [29]. For more information about reference models, we refer the reader to this survey.

Applications Reference models included into ERP systems such as the BAAN Dynamic Enterprise Modeler reference model or the SAP R/3 reference model serve as a comprehensive documentation of the ERP system and as a starting point for customizing the system. Customization starts with a high-level view (where the organizational structure and the value chain are modeled). It can later be refined into detailed descriptions at the level of single processes.

In general, reference models can be used to implement, measure, control and tune business processes. For these purposes, the reference model can contain much more than just a blueprint of business processes. Commonly, reference models contain a data model and methods to link the data model to the process model. They can also include performance attributes and metrics to measure process performance in a standardized way. For example, the Supply Chain Operations Reference Model [20] - a reference model that has been applied successfully in a large number of organizations around the world - contains more than 200 process elements (described in a textual and also in a semi-formal graphical form) as well as 550 metrics.

4.2 Anti-Pattern Catalogs

Anti-patterns are patterns that describe common mistakes, i.e. commonly reinvented bad solutions to recurring problems. In this section, we discuss the published anti-patterns for BPM.

4.2.1 Micro-Anti-Patterns

4.2.1.1 Control Flow Anti-Patterns

Purpose Control flow anti-patterns describe constructs in a business process model that lead to errors in the control flow. Technically spoken, such an error is defined as a violation of the soundness property [102].

Example An obvious example for a control flow anti-pattern is a combination between an exclusive choice (only one of many possible flows is executed) with a synchronization (a join node waiting for all incoming flows being completed).

Important Papers Several researchers discussed the patterns for which a wrong combination of control-flow elements can lead to errors in a business process model.

To our knowledge, the first such categorization of error patterns has been compiled at the University of Osaka. In Onoda et al.'s work [67], five so called deadlock-patterns are discussed. It has been shown, however, that one of these patterns does not have to correspond to a control-flow error [51].

Other papers discussing control-flow anti-patterns include [56, 111, 50]. The most comprehensive catalog of control-flow anti-patterns has been compiled by Gruhn and Laue [35].

Laue et al.[54] present a visual query language BPMN-Q which can be used for finding control flow anti-patterns. Using this language, not only errors are identified but also the erroneous parts of the business process diagram can be highlighted based on the detected anti-pattern.

Applications Control flow anti-patterns can be used by modelers as guidelines on how to avoid common problems. Such guidelines have been published by Koehler and Vanhatalo [50]. In this work, each anti-pattern was followed by one or several patterns that represent a correct solution to the modeling scenario. Smith [91] and Rozman et al. [78] also discussed such guidelines, mainly focusing on syntactic or control-flow errors in business process models. Similar guidelines - also including suggestions for wording and layout of BPMN models - have been published by Silingas and Mileviciene [89].

The patterns (in a formalized form) can also be used by automatic tools that can find (and sometimes also correct) modeling errors. Examples of such tools are discussed in [51] and [54]. Gruhn et al. [35] show that by using control-flow anti-patterns modeling errors can be found almost as accurately as by using static analysis. [34] shows some patterns where the readability of a model can be improved by substituting a control-flow anti-pattern by an alternative notation without changing the meaning of the model.

4.2.1.2 Data Flow Anti-Patterns

Purpose Data Flow Anti-Patterns describe constructs in a business process model for which problems in the data flow will occur.

Example An example for a data flow anti-pattern (called "Strongly Redundant Data Pattern" in [101]) describes the case that after a data element is written, it will never be read before it gets destroyed or the execution of the business process is completed.

Important Papers Sun [94] described three data-flow anti-patterns that can lead to problems in business process models (missing data, redundant data and conflict data). A more comprehensive list of anti-patterns has been published by Trčka et al. who discussed nine anti-patterns in [101]. Both [94] and [101] also describe methods for detecting instances of the anti-patterns at design time.

Koehler et al. [49] discuss anti-patterns related to the data-flow perspective; most of them are rather specific to the modeling language used in the *IBM Business Modeler* tool.

Applications As with control flow anti-patterns, data flow anti-pattern catalogs can be used for educating the modeler [49], but also for detecting data flow problems automatically [9, 94, 101, 62].

4.2.2 Medium Sized Anti-Patterns

Purpose Medium sized anti-patterns are concerned with locating *pragmatic* problems in a business process model. They work on a business level and require analyzing the labels that describe the meaning of activities and events used in a business process model. Often, they can also be used for detecting flaws in the actual process (not just in its model).

Example The PICTURE approach [14] allows detecting cases where a document is printed and later scanned again, which is most likely a problem in the actual process.

Important Papers Becker et al. [14] described anti-patterns from the banking sector. A similar discussion for anti-patterns in the area of public administration can be found in [16].

Gruhn and Laue [33] discussed three examples of patterns where the control flow in a business process model is modeled in an unstructured way. They analyze these patterns and show that the presence of certain kinds of patterns inside a model can give the modeler a hint for possible improvements of the model or the underlying business process.

Gruhn and Laue [36] used logic programming and text analysis for finding common pragmatic modeling errors like modeling a decision activity without modeling more than one possible result of this decision.

Applications The main purpose of medium sized anti-patterns is the improvement of actual business processes. They can be used together with business process patterns [74] as defined in Sect. 2.2.3.1.

The query language BPMN-Q allows searching for compliance violations in BPMN models [54]. Anti-patterns can be expressed as BPMN-Q queries.

4.3 Discussion

4.3.1 Dependencies between the Classification Dimensions

In the previous sections, we provided a classification scheme of BPM pattern catalogs according to three classification dimensions. The purpose of this classification is to document which kinds of patterns are useful for which purpose.

Note that there exist dependencies between the dimensions used in our classification scheme. Therefore the pattern catalogs can be found only in a subset of the possible combinations within the three classification dimensions (see Fig. 4.13).

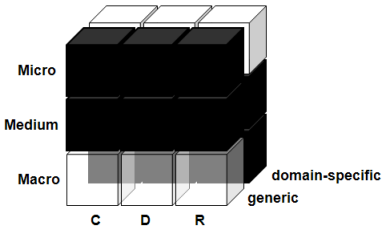

Figure 4.13: Union of all BPM Pattern Catalogs

4.3.2 Micro vs Macro

Micro patterns are small in size while having a very high level of abstraction and do not contain much business knowledge. This means that micro patterns are always non domain-specific. Hence, they are useful for assessing the expressiveness and suitability of different workflow/BPM/web service composition languages and standards. Also, these patterns were used to compare different BPM systems. Furthermore, they served as a base for constructing the YAWL system (providing both a modeling language and a workflow engine) which has been developed with the aim to support almost all micro-patterns [105].

On the other end of the scale, macro patterns are typically reference models which document many process models and the relationships between them. They contain a large amount of business knowledge, cover all aspects (functions, control flow, data and resources) and are typically industry specific. Macro patterns can be used by customizing reference models to the needs of an organization [110] or for a comparison with existing process models to pinpoint problems and improvement opportunities [40].

Medium sized patterns can be either domain-specific or not, and they can deal with any subsets of the BPM aspects.

Fig. 4.14 illustrates the relationship between different dimensions.

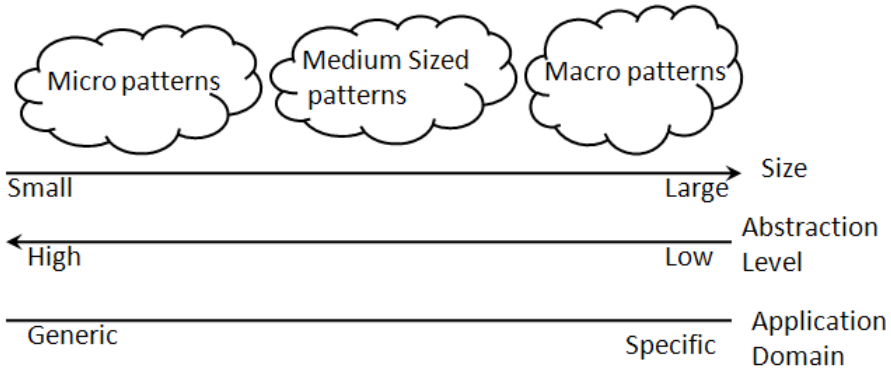

Figure 4.14: Relationship between Size, Abstraction Level and Universality

4.3.3 How to make use of the Taxonomy

Our taxonomy attempts to organize BPM patterns such that it becomes clear which kind of BPM pattern is useful for which purpose. By providing a kind of navigation between different pattern catalogs, one aim of our categorization is to start the integration of several BPM pattern catalogs into a BPM pattern language. Also the objective of the proposed taxonomy is to generate a wider debate about which criteria should be used for classifying BPM patterns.

For practitioners, a taxonomy can be helpful for locating patterns that fit their current needs.

4.3.4 Limitations to the BPM patterns survey and taxonomy

This is a first attempt to propose a BPM patterns taxonomy. We believe that there is a need to have a wider debate before the BPM research community can develop and agree on a definitive taxonomy. In addition, since there are many patterns in the literature and still there are others to be discovered/documented, this survey is not meant to be exhaustive but illustrative in regards to our proposed taxonomy.

4.4 Summary

In this chapter we presented a survey of BPM pattern catalogs and proposed a taxonomy for these patterns. BPM patterns differ in size, process aspect and universality. Micro patterns have been used mainly in comparing BPM standards and systems. Although it is useful to do so, but it is not clear what is the value of using a language/system with a feature if that feature is not used. Or it is not used when it is needed. Macro patterns are used in business modeling, since their

business knowledge content is rich but are usually large and hard to customize. Also, medium sized patterns have more business knowledge content compared to micro patterns, but are smaller in compared to macro patterns.

The next chapter presents an exemplar for one pattern catalog, the control flow micro patterns.

Chapter 5

BPM Patterns with Examples

This chapter presents the control flow BPM pattern catalog as mentioned in van der Aalst et al. paper [108]. The examples use Business Process Modeling Notation (BPMN).

5.1 Basic Control-Flow Patterns

In this section we cover the basic control-flow patterns. These patterns are considered the elementary patterns.

Basic control-flow patterns consist of five patterns:

1. Sequence pattern
2. Parallel Split
3. Synchronization
4. Exclusive Choice
5. Simple Merge

The following figure shows a process to prepare a hot drink either coffee or tea. This example will be used in explaining the few coming patterns.

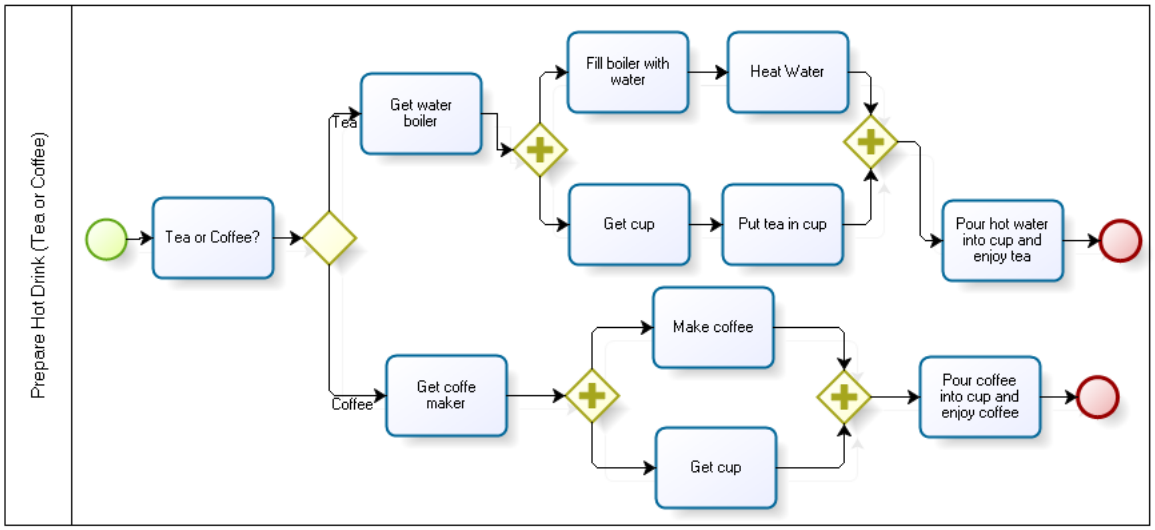

Figure 5.1: Prepare Coffee/Tea Process

5.1.1 Sequence pattern

If two activities are connected together with a control flow edge that has no conditions on it then we have a sequence pattern. This pattern is the fundamental building block for any process model. It is not uncommon to find it in almost every process model. It implies that whenever the first activity is completed the second should start, and as well whenever the second activity starts this implies that the first was executed till completion.

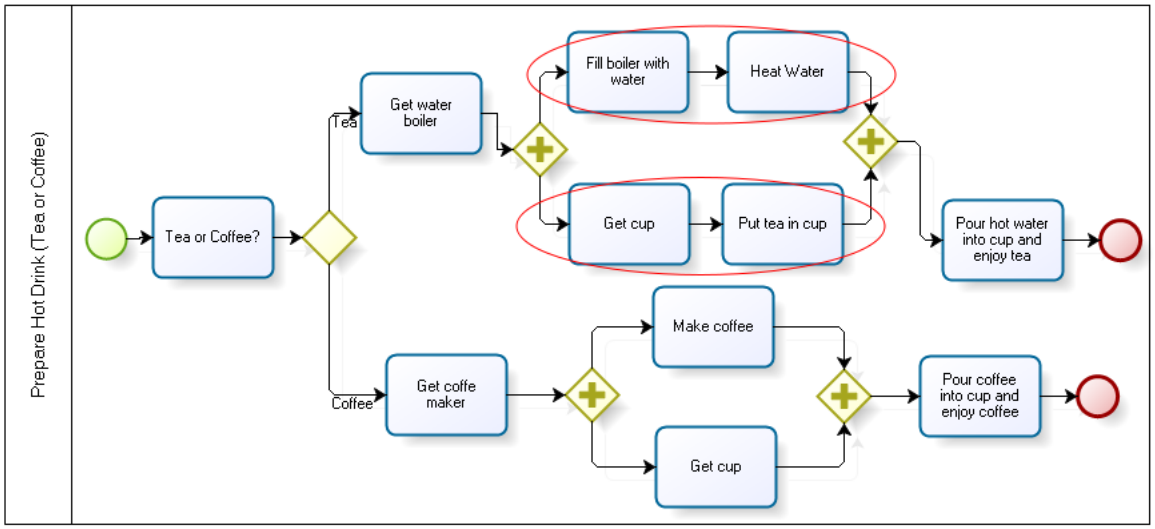

Figure 5.2: Sequence Pattern

As shown in the figure, in the "Prepare Hot Drink" process we can find the

sequence pattern more than once highlights in red ellipses. For example "Heat Water" can only start after "Fill boiler with Water" is completed. Same for the other sequence pattern, where "Put tea in cup" can only start after "Get cup" is completed.

5.1.2 Parallel Split (Fork)

The effect of the Fork pattern is that it makes the flow of control diverges into two or more parallel branches, each of which executes concurrently. Thus a single thread of execution will be split into two or more threads.

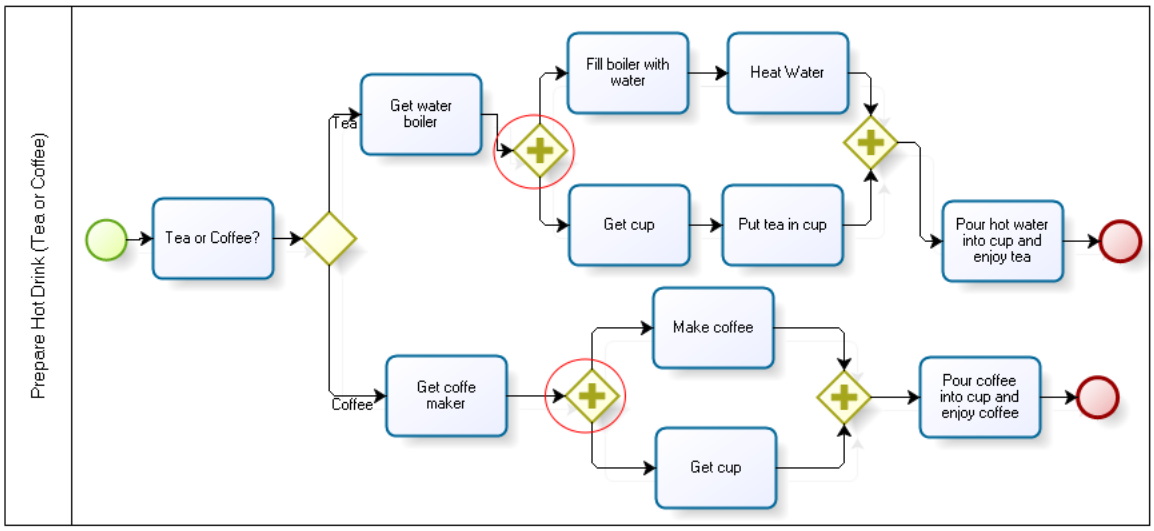

Figure 5.3: Parallel Split Pattern

As shown in the figure, while "Make coffee" is being executed "Get cup" can be executed in parallel. Same with the other Parallel Split pattern where "get cup" and "Put tea in cup" are being executed at the same time "Fill boiler with water" and "Heat Water".

5.1.3 Synchronization (Join, AND-join)

The effect of the Join is that it converges two or more branches into a single thread of control. In fact the Join is the complement of the Fork pattern. Note that if a Fork is used then the different branches may or may not be joined later. A Join waits until all of its input branches are enabled before it passes on the control of flow to its single output branch.

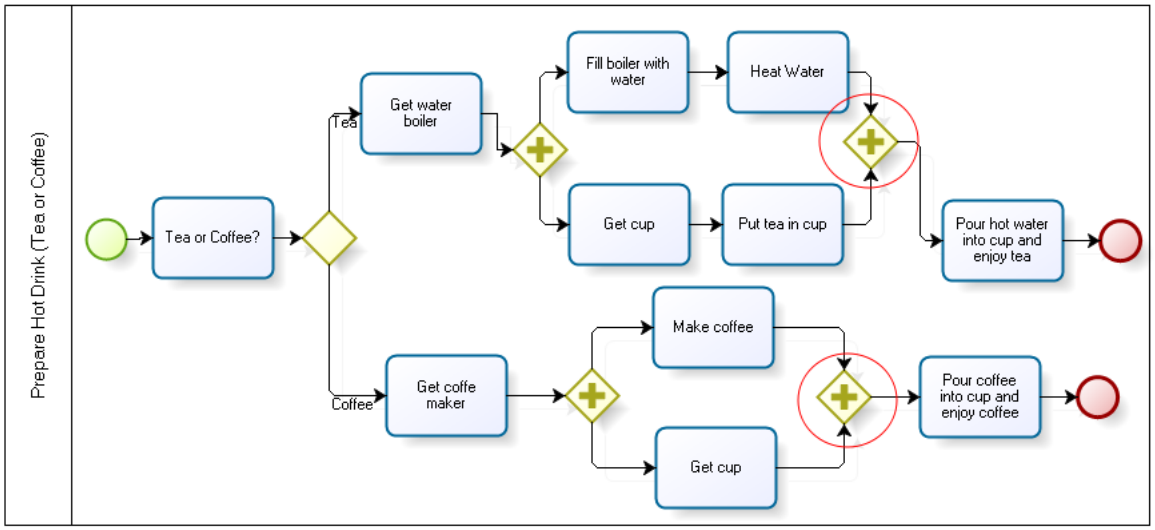

Figure 5.4: Join Pattern

As shown in the figure, a Join converges two or more branches into a single thread of control. For example, after "Heat water" and "Put tea in cup" are both completed then the Join converges these two threads of control into one and "Pour hot water into cup and enjoy tea" starts. Same thing happens for the other Join, only after "Make coffee" and "Get cup" are both completed then "Pour coffee into cup..." can start.

5.1.4 Exclusive Choice (Decision)

When one thread of control is followed by two or more branches and based on some condition only one of these branches will be followed then we have a Decision pattern. The condition on which the decision is based upon can take different forms. The condition can be the outcome of a preceding activity, the result of some user decision, or the value of a specific data element in the process engine.

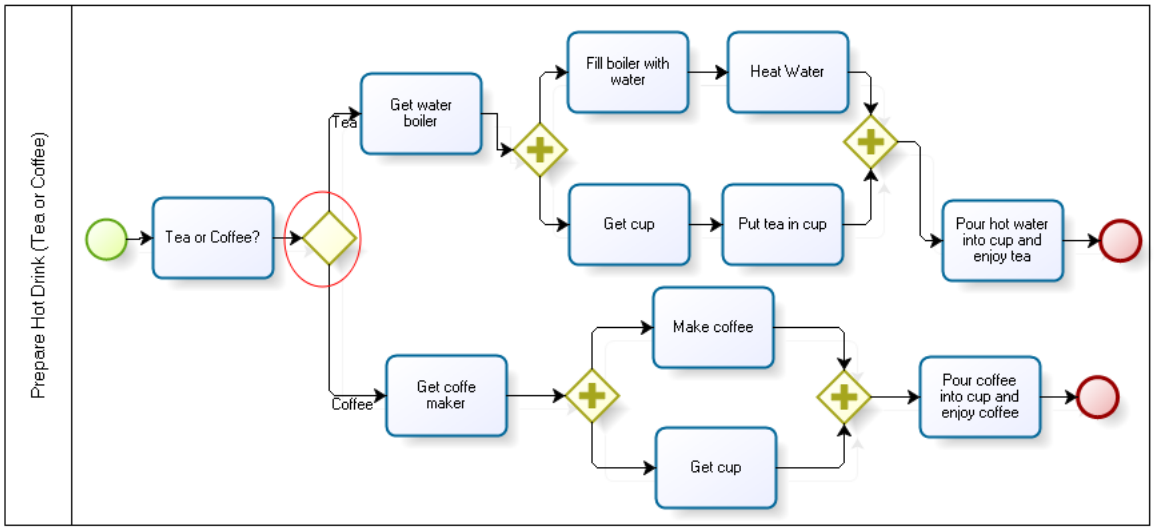

Figure 5.5: Decision Pattern

As shown in Fig.5.5, based on the outcome of the preceding activity either one of the two following branches will be taken, that is to say, either preparing tea or coffee.

5.1.5 Simple Merge (XOR-Join, Asynchronous join, Merge)

A Merge allows for the convergence of two or more branches into one subsequent branch. In effect the Merge is the complement of the Decision pattern.

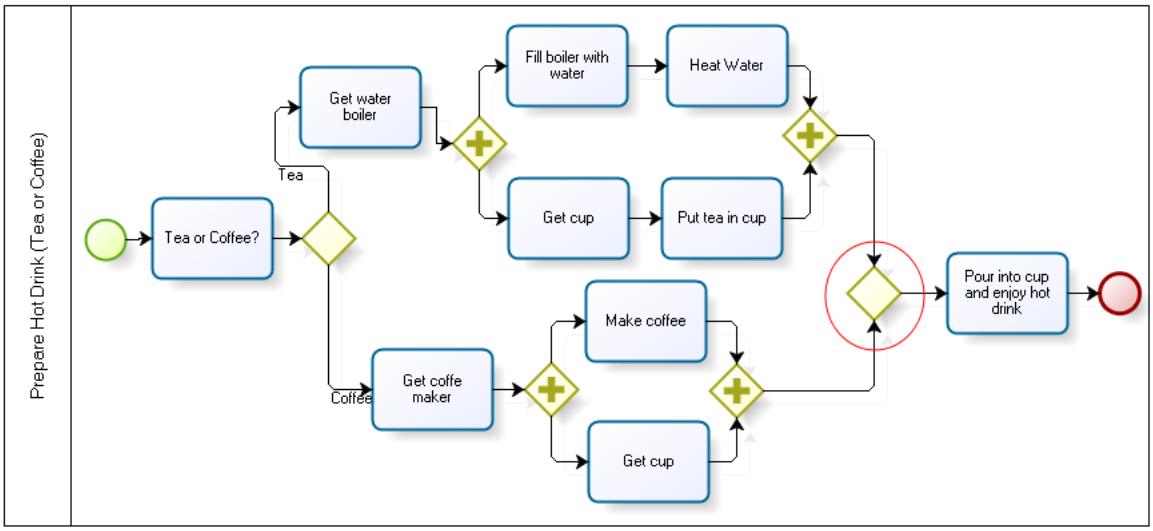

Figure 5.6: Merge Pattern

As shown in Fig. 5.6, a Merge allows for the two different branches (paths) to

converge. So whenever any one of the two branches before the Merge is enabled then the Merge will pass the control to the following activity. In this process, "Pour into cup and enjoy hot drink" will be enabled.

Join versus Merge

It is important to note the difference between Join and Merge. Both are used for converging different paths. As mentioned earlier, a Join waits until all of its input branches are enabled before it passes on the control of flow to its single output branch. On the other hand, Merge passes control to its single output branch, whenever any input branch is enabled. That is why another name for Merge is Asynchronous join.

Other than in special cases, typically Join is used with a Fork and a Merge is used with a Decision. Using a Merge with a Fork leads to the part after the Merge being executed more than once. On the other hand using Join with a Decision will lead to a deadlock.

5.1.6 Summary

Figure 5.7 summarizes the 5 basic control flow patterns with an example.

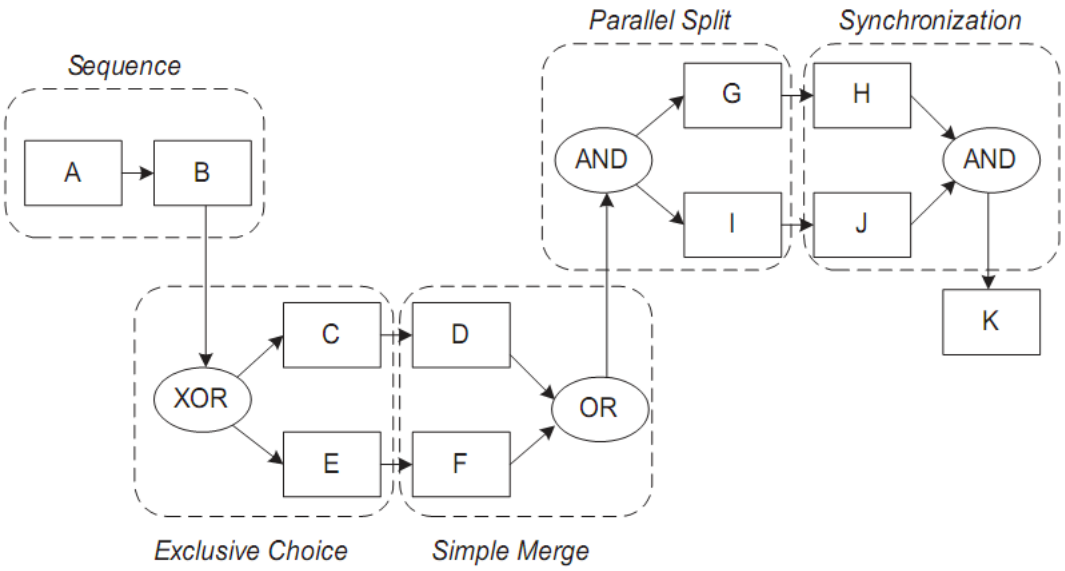

Figure 5.7: Graphical representation of basic control flow patterns [47]

5.2 Advanced Branching and Synchronization Patterns

5.2.1 Multi-Choice

With a Mutli-Choice a branch is diverged into two or more branches. Multi-Choice gets its input and based on some condition it enables one or more of the outgoing branches.

Fork versus **Decision** versus **Multi-Choice**

Fork enables all of its output branches; a Decision (exclusive decision) enables only one; and Multi-Choice is something in-between. Based on some conditions it can enable one, all or some combination of the output branches. The conditions are evaluated at runtime based on the execution information and a decision is made.

In BPMN this pattern can be represented in 3 different ways. By using implicit split and having conditions on the outgoing arcs. Another way is to use an OR-split. Finally it can be represented by a complex gateway.

The multi-choice pattern is also known by other names such as conditional routing, selection, OR-split, multiple choice.

Figure 5.8 presents the process of emergency call.

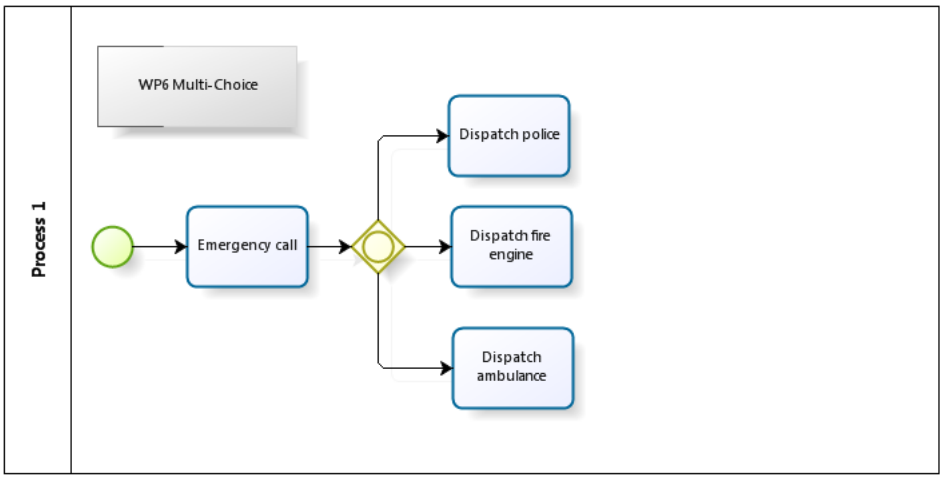

Figure 5.8: Multi-choice pattern (based on an example from [80])

In the example and depending on the nature of the emergency call, one or more of the three activities shown after is started immediately. The three activities are dispatch police, dispatch fire engine and dispatch ambulance.

5.2.2 Structured Synchronizing Merge

Structured synchronizing merge is the complement of the Multi-Choice. Structured Synchronizing Merge (simply known as Synchronizer) converges two or more

incoming branches into one. The synchronizer waits for every branch that has been activated by the previous Multi-Choice before passing the control to its outgoing branch.

Structured synchronizing merge pattern can also be known as "synchronizing join" or synchronizer.

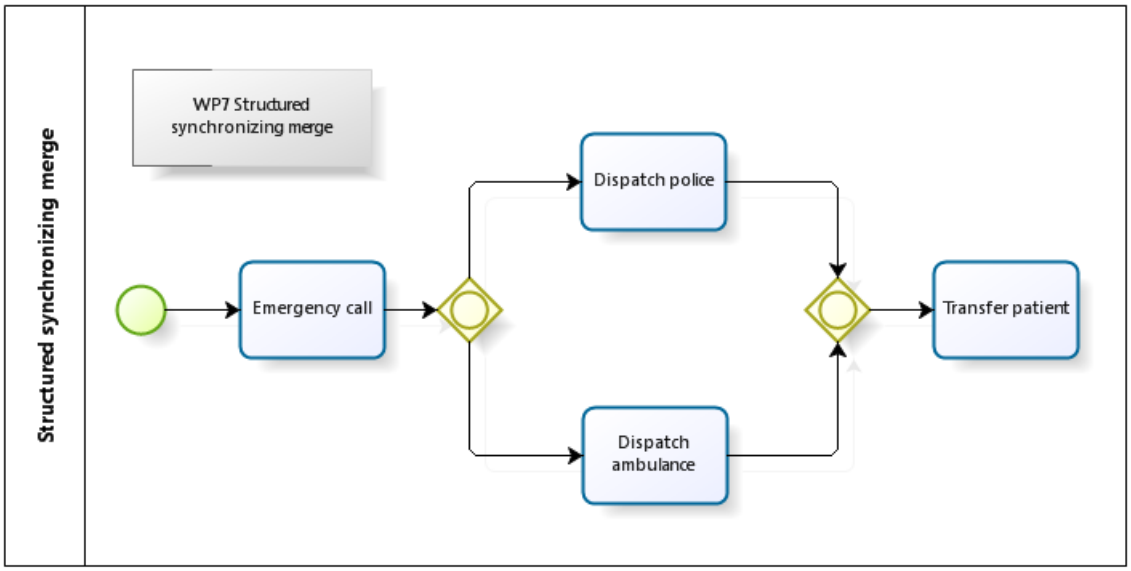

Figure 5.9: Structured synchronizing merge (based on an example from [80])

Figure Example

Figure 5.9 shows a process with a structured synchronizing merge pattern. In this example of emergency call and depending on the type of the emergency, either or both of the dispatch police and dispatch ambulance activities are initiated simultaneously. When all emergency vehicles arrive at the accident, the transfer-patient activity starts.

5.2.3 Multi Merge

A multi merge converges two or more incoming branches, where for each active incoming branch the outgoing branch will be activated. In this case there is no synchronization since for each incoming branch it will continue uninterrupted into the merged branch.

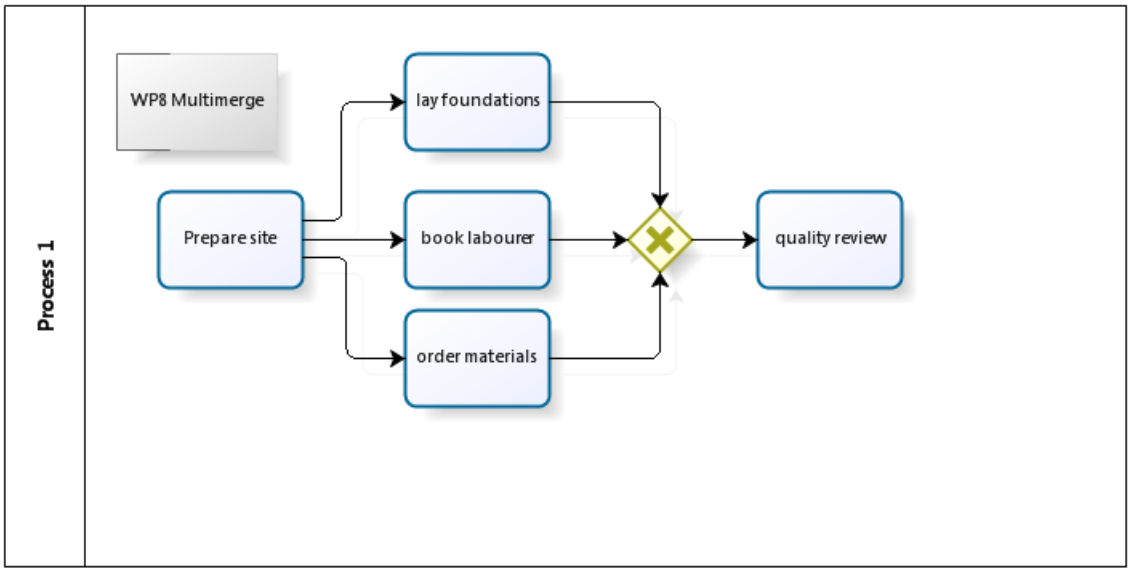

Figure 5.10: Multi merge pattern (based on an example from [80])

In the example shown and after preparing the site, we have three parallel activities "lay foundations", "order materials" and "book labourer" activities. A quality review activity is executed after each one of the parallel activities completes. This means that quality review will be executed three different times.

5.2.4 Structured Discriminator (1-out-of-M-join)

A structured discriminator converges two or more branches into one outgoing branch. Once the first incoming branch is active the discriminator passes control to the outgoing branch. When other incoming branches are enabled, these does not result in the activation of the outgoing branch until all incoming branches are enabled, then the discriminator is reseted.

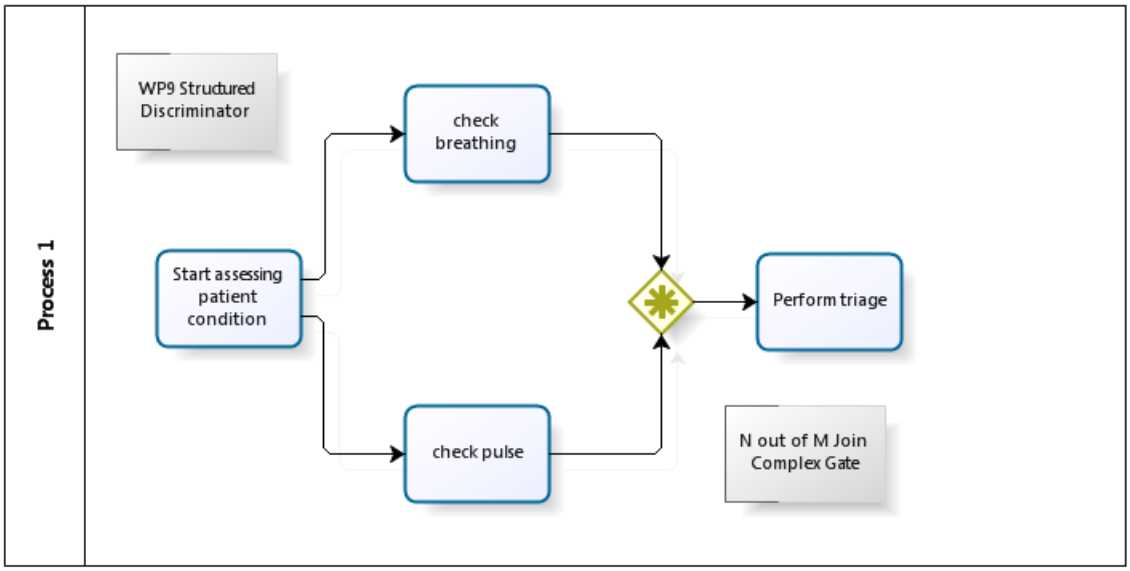

Figure 5.11: Structured Discriminator pattern (based on an example from [80])

In the example shown in Fig. 5.11 when dealing with a cardiac arrest, we need to perform "check breathing" and "check pulse" activities at the same time (i.e. in parallel). Whenever one of them is completed the "perform triage" activity starts. Whenever the other activity is completed it is then ignored and the "perform triage" activity is not repeated.

5.3 Structural Patterns

5.3.1 Arbitrary Cycles

An arbitrary cycle is a point in the process where one or more activities can be repeated several times. It is important to notice that for this pattern there can be more than one entry and/or exit points.

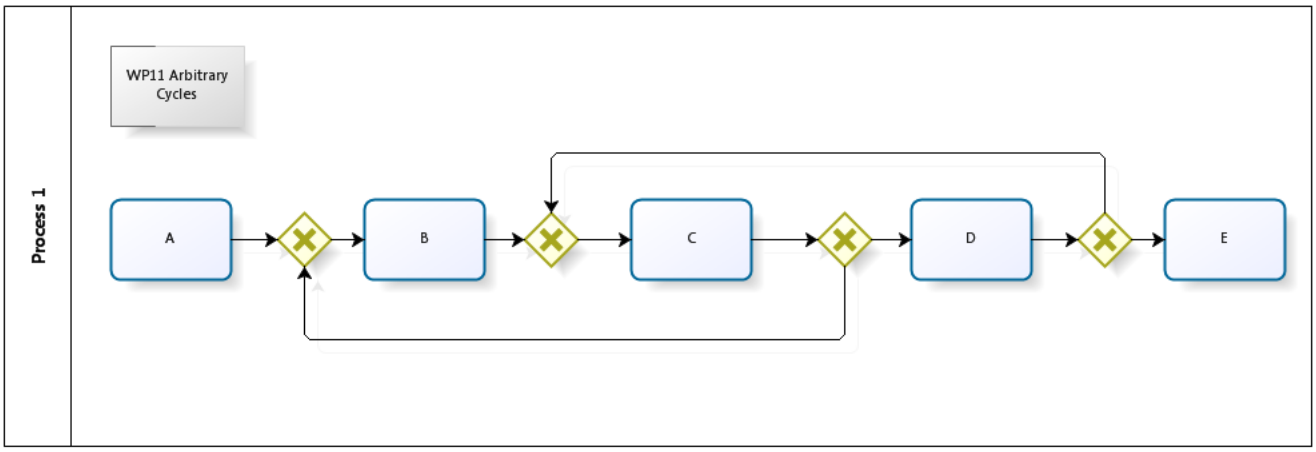

Figure 5.12: Arbitrary cycles pattern

In the example shown in the figure, we can notice two arbitrary cycles interleaved leading to more than one entry and more than one exit points.

5.3.2 Implicit Termination

Implicit termination pattern means that a process should terminate whenever there is nothing more that can be processed and that the process is not in a deadlock.

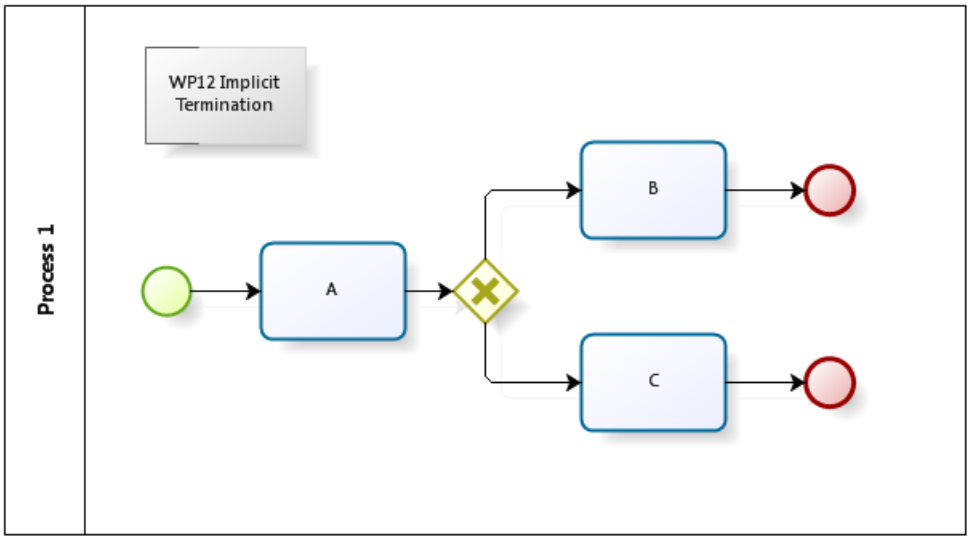

Figure 5.13: Implicit Termination pattern

In the example shown in the figure, after activity A is completed either activity B or C should start. But after B or C is completed nothing else is required to be processed so the process terminates.

5.4 Multiple Instance Patterns

Patterns described in this section are multiple instance patterns. A multiple instance pattern exists whenever there is an activity in a business process that can be running and active more than once. That is to say there are multiple activity instances within the same process instance for the same activity. This situation is similar to programming where having multiple threads running for the same definition. The difference between the patterns explained here is the need for synchronization when the activity instances are completed and when the number of the activity instances is determined.

5.4.1 Multiple Instances without Synchronization

In multiple instances without synchronization pattern the process instance can start multiple instances from the same activity such that there will be multiple activity instances at the same time. These instances will run concurrently and are independent from each other. In this pattern the multiple instances does not need to be synchronized when they are completed.

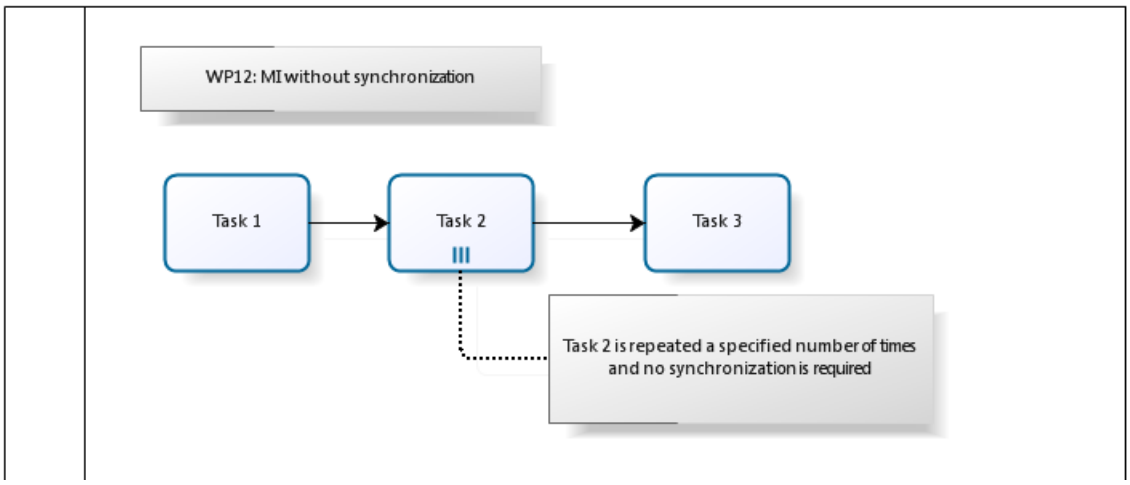

Figure 5.14: Multiple Instances without synchronization

5.4.2 Multiple Instances with a priori Design-Time Knowledge

In multiple instances with a priori design-time knowledge, there is a need for synchronization and the number of instances is known in advance at the design-time.

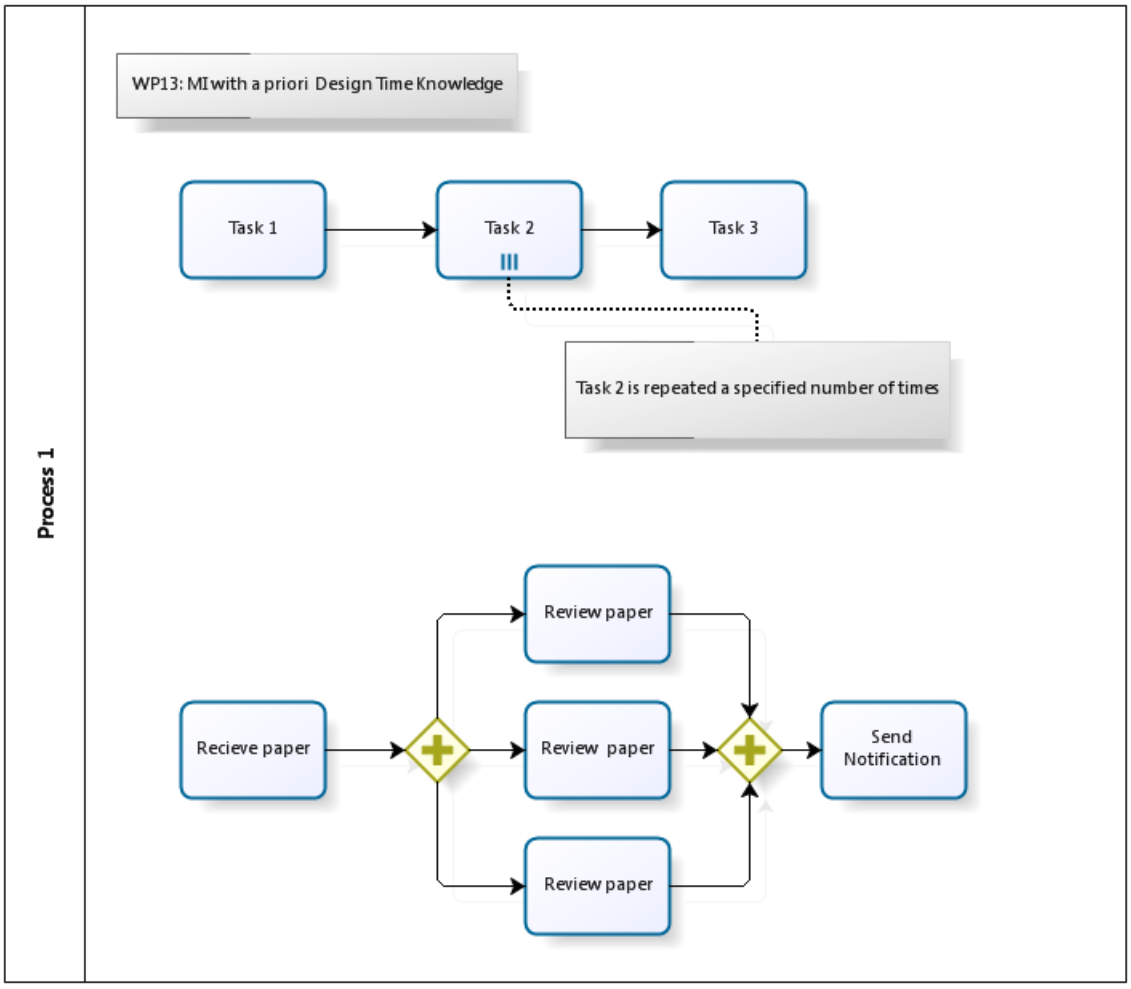

Figure 5.15: Multiple Instances with a priori Design-Time Knowledge

In the example of peer reviewing, it is determined that for each paper there will be three different reviewers. So the review paper is a multiple instance with the number three. In effect the execution of this process is equivalent to the process shown in Fig. 5.15 where there are three separate "review paper" activities.

5.4.3 Multiple Instances with a priori Run-Time Knowledge

In multiple instances with a priori run-time knowledge, there is a need for synchronization and the number of instances is known in advance at the run-time.

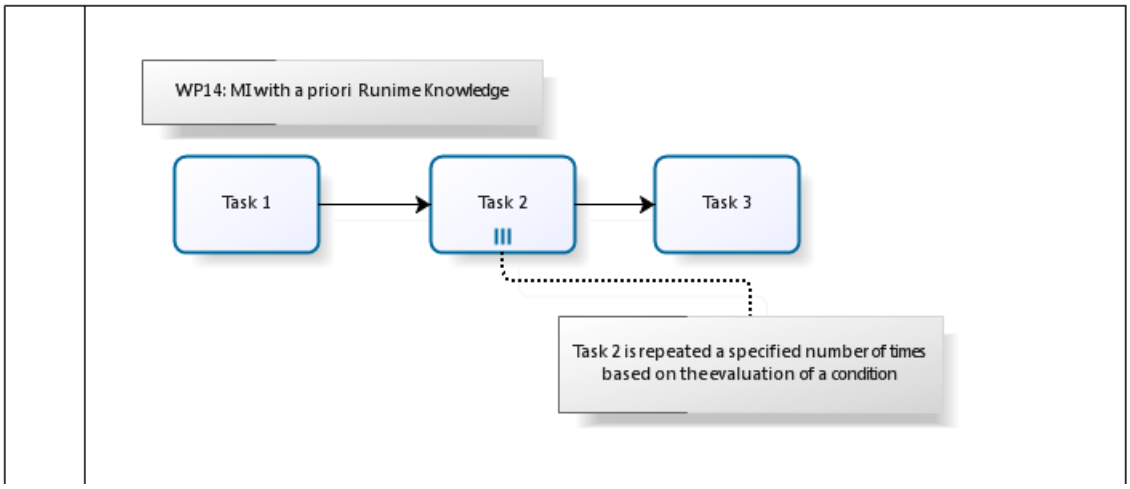

Figure 5.16: Multiple Instances with a priori Run-Time Knowledge

5.4.4 Multiple instances without a priori Run-Time Knowledge

In multiple instances without a priori run-time knowledge, there is a need for synchronization and the number of instances is not even known in advance at the run-time.

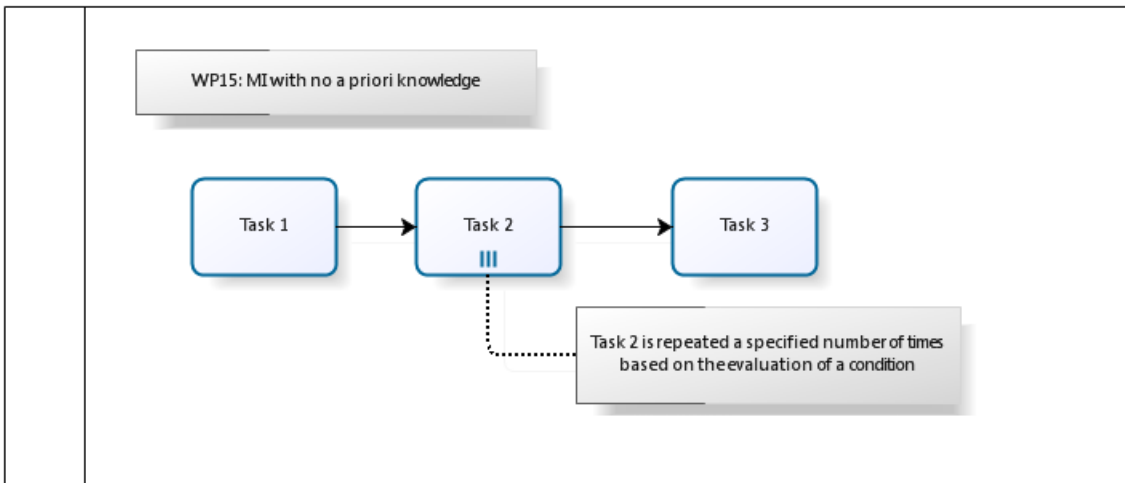

Figure 5.17: Multiple Instances without a priori Run-Time Knowledge

5.4.5 Comparing multiple instances patterns

When there exist multiple instances from the same activity within the same process instance either we do not need to synchronize the multiple instances or we need to synchronize them. If there is a need to synchronize the multiple instances then it

is required to know the number of the instances to be able to synchronize them. And the difference between the last three patterns is about when the number of instances is known. In one case when the number is know at the design time of the process, in another case when it is know in advance (a priori) at run-time (i.e. before any of the instances is started) while the final case is when it is not even known in advance at run-time.

To understand the difference between the previous 4 patterns, using feature model can be very helpful here. The following feature model summarizes the 4 different patterns.

Multiple Instance
- –Not Synchronized
- –Synchronized
  - —"Number of instances is known at"
    - —Design time
    - —Run time
    - —without runtime

Some notes on the previous feature model:

Not Synchronized and Synchronized are mutually exclusive.

"Number of instances is known at" is a mandatory feature.

The "Number of instances is known at" can be removed from the tree but it makes the tree more readable.

Also Design time, run time, without runtime are mutually exclusive.

5.5 State-based Patterns

5.5.1 Deferred Choice

A deferred choice pattern describes a point in the process model where there are several branches and only one will be selected based on interaction with the environment.

BPMN supports the deferred choice pattern through the use of event-based exclusive gateway. After the gateway there should be either a receive task or an intermediate event using message-based triggers (so the choice of which path to follow is based on the message received).

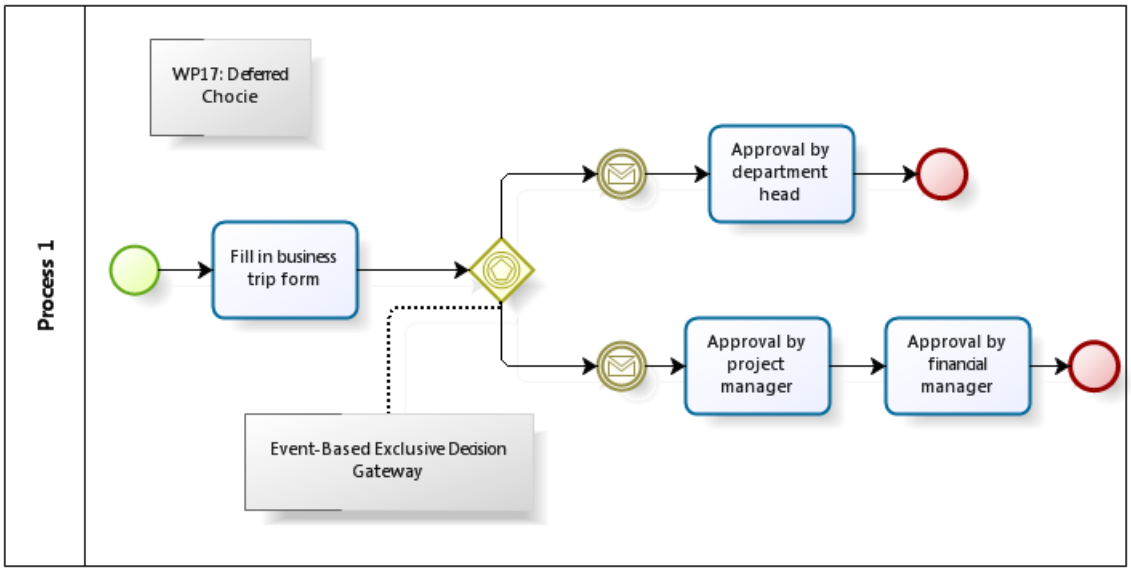

Figure 5.18: Deferred choice pattern

In the example shown in the Figure 5.18 (based on an example from Workflow Patterns[1]), a business trip must be approved before its booking. There are two options for an employee to get this approval. One way is by getting an approval from the department manager, the other is by getting the approval of both the project manager followed by the approval of the financial manager. Doing both is not possible and the decision is manual.

5.5.2 Interleaved Parallel Routing

In this pattern, there is a set of activities that need to be executed only once under the condition that the execution satisfy some partial order in addition to the restriciotn that no two activities can be executed concurrently.

A good example given by Russell et al.[80] is what is needed when dispatching an order. There are three different activities prepare invoice, pick goods and pack goods. The pratial order in this case is that pick goods must be executed before the pack goods. Prepare invoice can be executed at any tine.

Interleaved Parallel Routing is not supported in BPMN 1.0, since it only supports simple tasks via an ad-hoc process but no support for interleaving groups or sequences of tasks.

5.5.3 Milestone

A certain activity can be executed only if the process is in a certain state (named as a milestone). If we have a process with activities A, B and C. B can be executed

[1]Workflow Patterns, http://is.tm.tue.nl/research/patterns/patterns.htm

only if A has completed but before C starts.

An example for this pattern [80] is when "enroll student" activity can be started only if "open enrollment" has completed and "close off" did not start yet.

Milestone is not supported in BPMN 1.0 since there is no support for states.

5.6 Cancellation Patterns

Patterns in this section are about the concept of cancellation, which means withdrawing a running activity or process case. Cancellation concept is very useful in the case of exception handling.

5.6.1 Cancel Activity

An activity that is about to start is canceled before it starts execution. If the activity has started then it is disabled and if possible, the currently running instance of that activity is halted and removed.

BPMN supports the cancel activity pattern through compensation handlers attached to activities. This is achieved by attaching error type triggers to the boundary of the activity to be cancelled as shown in figure 5.19.

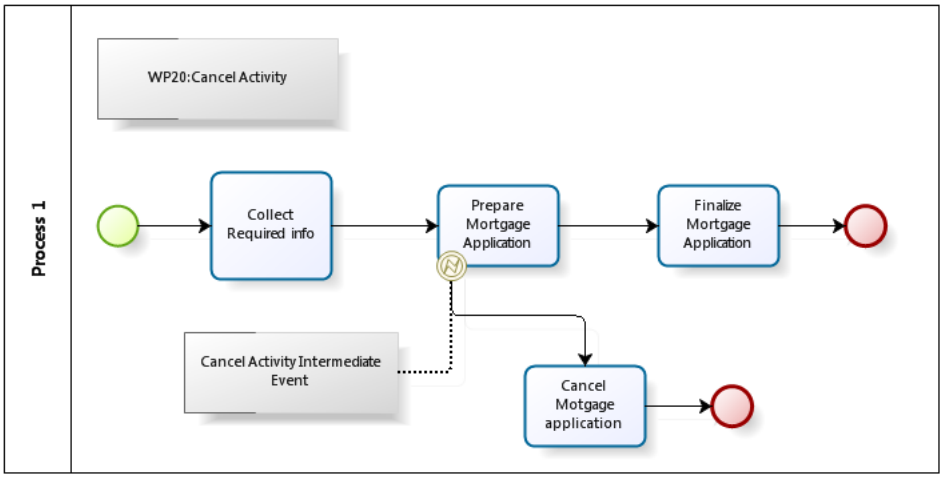

Figure 5.19: Cancel Activity pattern (based on an example from [80])

Figure 5.19 depicts a business process model for processing a mortgage. The applicant can decide not to purchase the house and cancel his/her application while the "Prepare Mortgage Application" activity is about to start or has started but no yet completed. But if we need to allow for the purchaser to be able to cancel the application for mortgage at any time within the process then we will need to use cancel case pattern.

5.6.2 Cancel Case

In the cancel case pattern, a complete process instance is cancelled. By cancelling the whole process instance, all of its executing activities, in addition to those which may execute at some time in the future, and all its sub-processes are cancelled. In this pattern, the process instance status is recorded as completed unsuccessfully.

BPMN supports cancel case pattern through including the whole process in transaction (sub process) with an associated end event. The end event allows to end all the executing activities in the process instance (case).

Figure 5.20 shows a process model that contain a subprocess. The subprocess is for the creation of mortgage. The whole subprocess can be cancelled regardless at which step in the subprocess is being executed. Figure 5.21 shows the same process but with one difference the mortgage subprocess is expanded.

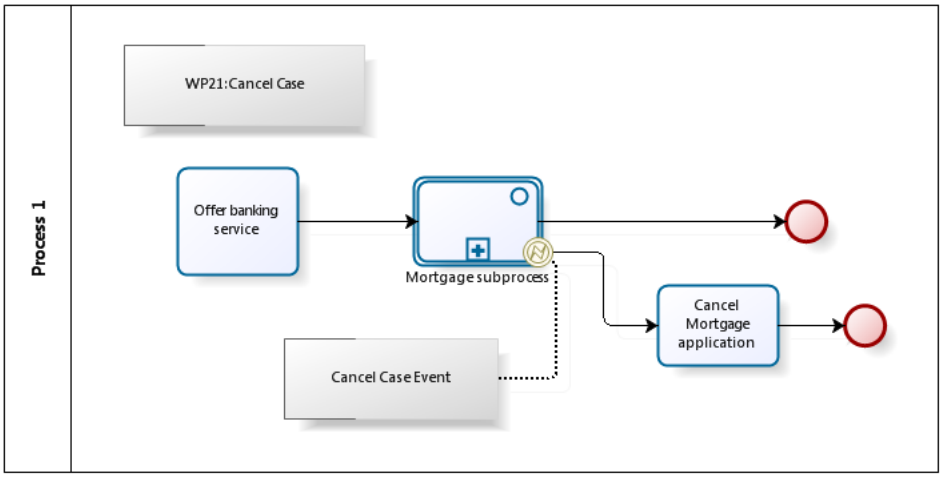

Figure 5.20: Cancel case pattern (based on an example from [80])

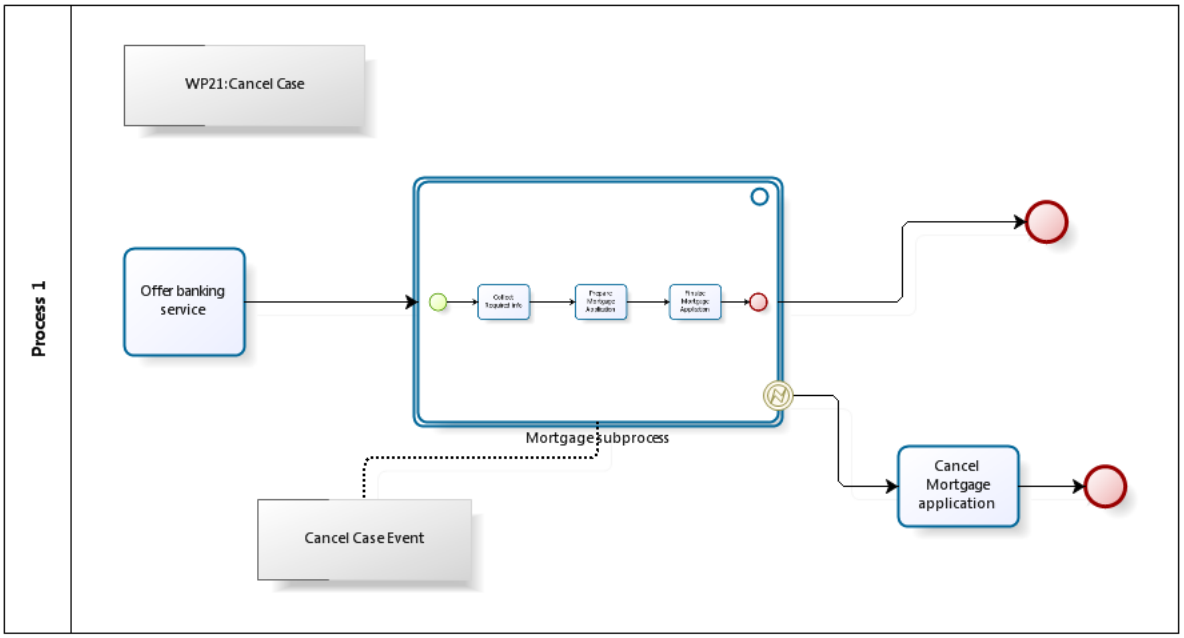

Figure 5.21: Cancel case pattern with subporcess expanded (based on an example from [80])

5.7 Summary

In this chapter we explained and presented examples for the control flow pattern catalog. We chose to present this catalog because control flow, although not the only perspective, is typically the most prevalent perspective when creating business process models. Process modeling tools should have a built-in support for different BPM patterns to ease the construction of BP models.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this thesis we presented an empirical research study that was conducted within an IT department with few thousand employees serving an organization with around 70,000 employees. This study was conducted over 9 months in two sites located in two different cities. Data collection phase included 2 on-site observations, 5 focus groups, and 18 interviews resulting in 23 hours of recordings, 181 pages of transcriptions, and 712 points identified. Based on this study we identified the challenges associated with Requirements Engineering in general and the creation and usage of business process models in particular from a practical point of view. We identified

- 10 key pain points ranked from highest to lowest priority. These are the challenges faced by Business Analysts in dealing with Requirements Engineering in practice.
- 5 tool features wish list from Business Analysts' points of view ranked from highest to lowest priority. These are the desired features for requirements management tools.
- 6 tool features wish list from developers' points of view ranked from highest to lowest priority. These are the desired features for requirements management tools.
- challenges associated with Business Process Modeling in practice.

In addition we examined the pain points and explored the connections between them, ultimately determining the cause/effect relationship that existed between most of the pain points then created a model to capture those relationships. In particular, the model explains why basic requirements determination needs time and effort which should not be reduced. If management attempts to stop or limit the required time and effort, these attempts will lead to less than optimal requirements specifications that, in turn, will create even greater negative consequences for the organization.

The study has shown that the studied organization applies business modeling, but uses a very lightweight version of the Line of Visibility Enterprise Modeling (LOVEM) methodology [72] and more advanced concepts are handled in textual description. That is to say, BAs see the value of BPM, but need simple ways to create models. Business process modeling patterns have been proposed to ease creating more complete models [95]. This is the reason why the thesis studies BPM patterns, by providing a survey and a taxonomy. We consider the survey to be the first survey of BPM pattern catalogs. Also the taxonomy presented is a first attempt to formulate a BPM pattern catalogs taxonomy. Further, the catalog with examples is a first step towards future work of preparing teaching material and creating empirical studies to show whether these patterns can actually help practitioners create better models.

6.2 Recommendations for Future Work

How BPM patterns can be made available in a repository [66] and also how process variants can be defined and transformed into concrete models [52, 77, 112, 39, 73, 10, 75] are still the subjects of ongoing research. Another problem that needs to be researched is the question of how semantic queries can be used to locate suitable patterns. An approach based on Pi-calculus and ontologies has been published by Markovic and Pereira [60].

From a practical point of view, it is desirable to integrate the use of patterns directly into the modeling tools. In [37], Gschwind et al. state that "despite the common belief in the importance of patterns, only limited support for using patterns in today's business process modeling tools can be found." A step towards integrating patterns into modeling tools should be a formalization of the patterns. An effort to achieve such formalization is documented in [100, 99, 98, 38] where the authors present an UML-based process meta-models that allow explicit representation of process patterns.

It is important to point out that empirical studies showing how these patterns can be used and how they are used in reality are still rare. Empirical studies can help to determine which patterns are more useful than others, and why. For example, it is a relevant question to ask which subset of micro patterns is the most useful for a given purpose [64]. Moreover, studies on the factors affecting the usage of BPM patterns and studies documenting the effects of BPM patterns on modeling time and model quality are yet to be seen.

Last, it is important to question which techniques can be used to improve and widen pattern adoption in industry. Having a taxonomy can be the starting point from which both researchers and business process modeling experts can collaborate.

Appendix A

Permissions Page

Most of the material presented in this thesis was taken from existing and submitted papers. The exploratory study explained in Chapter 3 and parts of Chapter 6 are based on [8], to which my personal contribution was approximately 40%. Parts of Section 3.3 is based on [17], to which my personal contribution was approximately 20%. Section 2.2, Chapter 4, and parts of Chapter 6 are based on [1], where I have selected the topic and developed the main classification and then both co-authors contributed at about 50% in the paper.

Mohamed AbdElRazik

As one of the authors of [8, 17], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Krzysztof Czarnecki

As one of the authors of [8, 17], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Daniel Berry

As one of the authors of [8, 17], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Michal Antkiewicz

As one of the authors of [1], I fully acknowledge the thesis author's statement above and give full permission to use any part of the papers we co-authored mentioned above. I also fully acknowledge that the thesis represents original research conducted by the thesis author.

Ralf Laue

References

[1] Mohamed AbdElRazik and Ralf Laue. A Survey of Business Process Modeling Patterns: Towards a Business Process Modeling Pattern Language. *Software: Practice and Experience journal - Special Issue on "Pattern Languages: Addressing the Challenges*, 2011. (submitted). 60

[2] Mohamed AbdelRazik, Janette Wong, Krzysztof Czarnecki, and Leho Nigul. Bridging the Business-IT divide using BPM: Challenges and opportunities. In *CASCON '09: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, pages 327–327, New York, NY, USA, 2009. ACM. 1

[3] Rafael Accorsi and Claus Wonnemann. Strong Non-Leak Guarantees for Workflow Models. In *ACM Symposium on Applied Computing (Enterprise Engineering Track)*, TaiChung, Taiwan, 2011. 32

[4] Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos. *Patterns for e-business: A Strategy for Reuse.* IBM Press, 2001. 7

[5] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau, K. Plösser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, and M Zeller. WS-BPEL Extension for People (BPEL4People), version 1.0, 2007. Technical report. 26

[6] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, New York, August 1977. 5

[7] Scott W. Ambler. *Process Patterns: Building Large-Scale Systems Using Object Technology.* Cambridge University Press, New York, NY, USA, 1998. 6

[8] Michal Antkiewicz, Mohamed AbdelRazik, Krzysztof Czarnecki, and Daniel Berry. The Requirements Engineering Practices and Tool Support at X. Technical report, University of Waterloo, January 13, 2010. 9, 60

[9] A. Awad, G. Decker, and N. Lohmann. Diagnosing and repairing data anomalies in process models. In *Business Process Management Workshops, BPM*

2009 International Workshops, Ulm, Germany, September 2009, Revised Papers, volume 43, pages 5–16. Springer, 2010. 35

[10] Thomas Baier, Emilian Pascalau, and Jan Mendling. On the Suitability of Aggregated and Configurable Business Process Models. In *Enterprise, Business-Process and Information Systems Modeling*, volume 50 of *Lecture Notes in Business Information Processing*, pages 108–119. Springer Berlin Heidelberg, 2010. 59

[11] A. Barros, M. Dumas, and A. ter Hofstede. Service Interaction Patterns: Towards a Reference Framework for Service-based Business Process Interconnection. Technical report, Queensland University of Technology, Brisbane, 2005. `http://www.workflowpatterns.com/documentation/documents/ServiceInteractionPatterns.pdf`. 8

[12] Alistair P. Barros, Marlon Dumas, and Arthur H. M. ter Hofstede. Service Interaction Patterns. In van der Aalst et al. [109], pages 302–318. 5, 8

[13] Oscar Barros. Business process patterns and frameworks: Reusing knowledge in process innovation. *Business Process Management Journal*, 13(1):47–69, 2007. 6

[14] Jörg Becker, Philipp Bergener, Michael Räckers, Burkhard Weiß, and Axel Winkelmann. Pattern-Based Semi-Automatic Analysis of Weaknesses in Semantic Business Process Models in the Banking Sector. In *18th European Conference on Information Systems (ECIS 2010)*, pages 1–12, 2010. Pretoria, South Africa. 35

[15] Jörg Becker, Burkhard Weiß, and Axel Winkelmann. Developing a Business Process Modeling Language for the Banking Sector - A Design Science Approach. In *15th Americas Conference on Information Systems (AMCIS 2009), San Francisco, USA*, pages 1–12, 2009. 31

[16] Jörg Becker, Lars Algermissen, Thorsten Falk, Daniel Pfeiffer, and Philippe Fuchs. Model Based Identification and Measurement of Reorganization Potential in Public Administrations - the PICTURE-Approach. In *Tenth Pacific Asia Conference on Information Systems (PACIS 2006), Kuala Lumpur, Malaysia*, pages 860–875, 2006. 31, 35

[17] Daniel M. Berry, Krzysztof Czarnecki, Michał Antkiewicz, and Mohamed AbdelRazik. Requirements Determination is Unstoppable: An Experience Report. In *2010 18th IEEE International Requirements Engineering Conference*, 09/2010 2010. 2, 17, 60

[18] Barry Boehm and Victor R. Basili. Software Defect Reduction Top 10 List. *Computer*, 34(1):135–137, 2001. 1

[19] B.W. Boehm. Software Engineering. *Computers, IEEE Transactions on*, C-25(12):1226 –1241, dec. 1976. 1

[20] Peter Bolstorff and Robert Rosenbaum. *Supply Chain Excellence: A Handbook for Dramatic Improvement Using the SCOR Model, second edition.* Amacom, New York, NY, USA, 2007. 33

[21] Egon Börger. Modeling Workflow Patterns from First Principles. In *Proceedings of the 26th international conference on Conceptual modeling*, ER'07, pages 1–20, Berlin, Heidelberg, 2007. Springer-Verlag. 23

[22] Christoph Bussler and Armin Haller, editors. *Business Process Management Workshops, BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005, Revised Selected Papers*, volume 3812 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006. 64, 71

[23] Silvana Castano and Maria Grazia Fugini. Rules and Patterns for Security in Workflow Systems. In *Proceedings of the IFIP TC11 WG 11.3 Twelfth International Working Conference on Database Security XII: Status and Prospects*, pages 59–74, Deventer, The Netherlands, The Netherlands, 1999. Kluwer, B.V. 32

[24] Thomas Curran, Gerhard Keller, and Andrew Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998. 33

[25] Thomas H. Davenport. *Process Innovation: Reengineering Work Through Information Technology.* Harvard Business School Press, Boston, MA, USA, 1993. 4

[26] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Property specification patterns for finite-state verification. In *FMSP '98: Proceedings of the second workshop on Formal methods in software practice*, pages 7–15. ACM Press, 1998. 7

[27] Amal Elgammal, Oktay Türetken, Willem-Jan van den Heuvel, and Mike P. Papazoglou. Root-cause analysis of design-time compliance violations on the basis of property patterns. In *Service-Oriented Computing - 8th International Conference, ICSOC 2010, San Francisco, CA, USA, December 7-10, 2010. Proceedings*, volume 6470 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2010. 7

[28] Hans-Erik Eriksson and Magnus Penker. *Business Modeling With UML: Business Patterns at Work.* John Wiley & Sons, Inc., New York, NY, USA, 1998. 6

[29] Peter Fettke, Peter Loos, and Jorg Zwicker. Business Process Reference Models: Survey and Classification. In Bussler and Haller [22], pages 469–483. 33

[30] Alexander Foerster, Gregor Engels, and Tim Schattkowsky. Activity Diagram Patterns for Modeling Quality Constraints in Business Processes. In Lionel Briand and Clay Williams, editors, *Model Driven Engineering Languages and Systems*, volume 3713 of *Lecture Notes in Computer Science*, pages 2–16. Springer Berlin / Heidelberg, 2005. 7

[31] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series)*. Addison-Wesley Professional, illustrated edition edition, January 1994. 5

[32] Jaap Gordijn, Hans Akkermans, and Hans van Vliet. Business Modelling Is Not Process Modelling. In *Conceptual Modeling for E-Business and the Web*, volume 1921 of *Lecture Notes in Computer Science*, pages 40–51. Springer, 2000. 7

[33] Volker Gruhn and Ralf Laue. Good and Bad Excuses for Unstructured Business Process Models. In *Proceedings of 12th European Conference on Pattern Languages of Programs (EuroPLoP 2007)*, 2007. 35

[34] Volker Gruhn and Ralf Laue. Reducing the Cognitive Complexity of Business Process Models. In *IEEE International Conference on Cognitive Informatics, Hong Kong 2009*, 2009. 34

[35] Volker Gruhn and Ralf Laue. A Heuristic Method for Detecting Problems in Business Process Models. *Business Process Management Journal*, 16(4), 2010. 34

[36] Volker Gruhn and Ralf Laue. Detecting Common Errors in Event-Driven Process Chains by Label Analysis. *Enterprise Modelling and Information Systems Architecture*, x(x), Jan 2011. 35

[37] Thomas Gschwind, Jana Koehler, and Janette Wong. Applying Patterns during Business Process Modeling. In Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors, *BPM*, volume 5240 of *Lecture Notes in Computer Science*, pages 4–19. Springer, 2008. 24, 59

[38] Mariele Hagen and Volker Gruhn. Process Patterns - a Means to Describe Processes in a Flexible Way. In *5th International Workshop on Software Process Simulation and Modeling (ProSim 2004)*, pages 32–39, 2004. 59

[39] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Capturing Variability in Business Process Models: The Provop Approach. *Journal of Software Maintenance*, 22(6-7):519–546, 2010. 59

[40] Oliver Holschke, Philipp Gelpke, Philipp Offermann, and Christian Schröpfer. Business Process Improvement by Applying Reference Process Models in SOA - a Scenario-based Analysis. In Martin Bichler, Thomas Hess, Helmut Krcmar, Ulrike Lechner, Florian Matthes, Arnold Picot, Benjamin Speitkamp, and Petra Wolf, editors, *Multikonferenz Wirtschaftsinformatik*. GITO-Verlag, Berlin, 2008. 36

[41] Pavel Hruby. *Model-Driven Design Using Business Patterns*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 6

[42] Hwai-Jung Hsu and Feng-Jian Wang. Delegation Patterns in Workflow Systems. In *The 7th International Workshop on Software Cybernetics (IWSC 2010), Seoul, Korea*, 2010. 26

[43] M. Jackson. Problems and requirements (software development). In *RE'95*, pages 2–9, Los Alamitos, CA, USA, 1995. IEEE Computer Society. 16

[44] Jürgen Jung and Jonas Sprenger. Muster für die Geschäftsprozessmodellierung. In Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, and Klaus Turowski, editors, *Integration, Informationslogistik und Architektur. Proceedings der DW2006*, number 90 in Lecture Notes in Informatics, pages 189–204, 2006. 6

[45] V. Kabilan. Contract Workflow Model Patterns Using BPMN. In T.A. Halpin, K. Siau, and J. Krogstie, editors, *Proceedings of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD'05), held in conjunctiun with the 17th Conference on Advanced Information Systems (CAiSE'05), Porto, Portugal, EU*, pages 557–568. FEUP, Porto, Portugal, EU, 2005. 30

[46] Maria Kavanagh. Towards A Pattern Language for Business Process Modeling. In *Proceedings of 9th European Conference on Pattern Languages of Programs (EuroPLoP 2004)*, 2004. 6

[47] B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows*. PhD thesis, Faculty of Information Technology, Queensland University of Technology, Australia, 2003. x, 23, 44

[48] Jae Ho Kim, Woojong Suh, and Heeseok Lee. Document-based workflow modeling: a case-based reasoning approach. *Expert Syst. Appl.*, 23(2):77–93, 2002. 30, 31

[49] Jana Koehler and Jussi Vanhatalo. Process Anti-Patterns: How to Avoid the Common Traps of Business Process Modeling, Part 2: Modeling data flow. *IBM WebSphere Developer Technical Journal*, Issue 10.4, April 4, 2007. 35

[50] Jana Koehler and Jussi Vanhatalo. Process Anti-Patterns: How to Avoid the Common Traps of Business Process Modeling, Part 1: Modeling control flow.

IBM WebSphere Developer Technical Journal, Issue 10.2, February 28, 2007. 34

[51] Stefan Kühne, Heiko Kern, Volker Gruhn, and Ralf Laue. Business process modeling with continuous validation. *Journal of Software Maintenance and Evolution, Research and Practice*, (22):547 – 566, 2010. 34

[52] Marcello La Rosa. *Managing Variability in Process-Aware Information Systems*. PhD thesis, Faculty of Science and Technology, Queensland University of Technology, Australia, 2009. 59

[53] Andreas Lanz, Barbara Weber, and Manfred Reichert. Workflow Time Patterns for Process-aware Information Systems. In *Enterprise, Business-Process, and Information Systems Modelling*, LNBIP 50, pages 94–107, Hammamet, Tunisia., Jun 2010. 11th International Workshop BPMDS and 15th International Conference EMMSAD at CAiSE 2010, Springer. 7

[54] Ralf Laue and Ahmed Awad. Visualization of Business Process Modeling Anti Patterns. In *Proceedings of the First International Workshop on Visual Formalisms for Patterns*, Electronic Communications of the EASST, 2009. 34, 36

[55] Barbara Staudt Lerner, Stefan Christov, Leon J. Osterweil, Reda Bendraou, Udo Kannengiesser, and Alexander Wise. Exception Handling Patterns for Process Modeling. *IEEE Transactions on Software Engineering*, pages 162–183, 2010. 27, 28

[56] Rong Liu and Akhil Kumar. An Analysis and Taxonomy of Unstructured Workflows. In *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 268–284. Springer, 2005. 34

[57] Jacques Lonchamp. Process Model Patterns For Collaborative Work. In *15th IFIP World Computer Congress - Telecooperation'98*, Vienna, Austria, 1998. 30

[58] Therani Madhusudan, J. Leon Zhao, and Byron Marshall. A case-based reasoning framework for workflow model management. *Data Knowl. Eng.*, 50:87–115, July 2004. 31

[59] Thomas W. Malone, Kevin Crowston, and George A. Herman. *Organizing Business Knowledge: The MIT Process Handbook*. MIT Press, Cambridge, MA, USA, 2003. 33

[60] Ivan Markovic and Alessandro Pereira. Towards a Formal Framework for Reuse in Business Process Modeling. In Arthur ter Hofstede, Boualem Benatallah, and Hye-Young Paik, editors, *Business Process Management Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 484–495. Springer Berlin / Heidelberg, 2008. 59

[61] Steffen Mazanek and Mark Minas. Business process models as a showcase for syntax-based assistance in diagram editors. In *MoDELS*, pages 322–336, 2009. 24

[62] Hema S. Meda, Anup Kumar Sen, and Amitava Bagchi. On detecting data flow errors in workflows. *J. Data and Information Quality*, 2:4:1–4:31, July 2010. 35

[63] Jan Mendling, Gustaf Neumann, and Markus Nüttgens. Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). In *Second GI-Workshop XML4BPM XML for Business Process Management*, 2005. 24

[64] Michael Muehlen and Jan Recker. How much language is enough? Theoretical and practical use of the business process modeling notation. *Advanced Information Systems Engineering*, pages 465–479, 2008. 59

[65] Kioumars Namiri and Nenad Stojanovic. Pattern-based design and validation of business process compliance. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *Lecture Notes in Computer Science*, pages 59–76. Springer Berlin / Heidelberg, 2007. 32

[66] A. Norta and P. Grefen. A Pattern Repository for Establishing Inter-Organizational Business Processes. BETA Working Paper Series, WP 175, Eindhoven University of Technology, Eindhoven, 2006. 59

[67] Sen'ichi Onoda, Yoshitomo Ikkai, Takashi Kobayashi, and Norihisa Komoda. Definition of deadlock patterns for business processes workflow models. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences-Volume 5*, page 5065. IEEE Computer Society, 1999. 34

[68] Alexander Osterwalder and Yves Pigneur. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers.* John Wiley & Sons, Inc., New York, NY, USA, 2010. 7

[69] Martyn A. Ould. *Business Processes : Modelling and Analysis for Re-Engineering and Improvement.* John Wiley and Sons, 1995. 29

[70] Marco Paludo, Robert Burnett, and Edgard Jamhour. Patterns leveraging analysis reuse of business processes. In William Frakes, editor, *Software Reuse: Advances in Software Reusability*, volume 1844 of *Lecture Notes in Computer Science*, pages 205–300. Springer Berlin / Heidelberg, 2000. 22, 29, 30

[71] Frank Puhlmann and Mathias Weske. Using the π-calculus for formalizing workflow patterns. In van der Aalst et al. [109], pages 153–168. 24

[72] IBM Redbooks. *Business Process Reengineering and Beyond.* IBM, 1995. 18, 59

[73] H.A. Reijers, R.S. Mans, and R.A. van der Toorn. Improved Model Management with Aggregated Business Process Models. *Data and Knowledge Engineering*, 68(2):221 – 243, 2009. 59

[74] Hajo A. Reijers and Selma Limam Mansar. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. *Omega, The International Journal of Management Science*, 33:283–306, 2005. 6, 36

[75] Iris Reinhartz-Berger, Pnina Soffer, and Arnon Sturm. A domain engineering approach to specifying and applying reference models. In Jörg Desel and Ulrich Frank, editors, *EMISA*, volume 75 of *LNI*, pages 50–63. GI, 2005. `http://is.haifa.ac.il/~spnina/publications/ReinhartzSofferSturm.pdf`. 59

[76] Motschnig Renate, Vinek Günther, and Philipp Randa. Specifying and analysing static and dynamic patterns of administrative processes. In *ECIS 2002, European Conference on Information Systems*, pages 862–871, Gdansk, Polen, 6 2002. 22, 29

[77] Marcello Rosa, Johannes Lux, Stefan Seidel, Marlon Dumas, and Arthur H. M. Hofstede. Questionnaire-driven configuration of reference process models. *Advanced Information Systems Engineering*, 4495:424–438, 2007. 59

[78] Tomislav Rozman, Gregor Polancic, and Romana Vajde Horvat. Analysis of most common process modeling mistakes in BPMN process models. In *2008 BPM and Workflow Handbook*, 2008. 34

[79] N. Russell and W.M.P. van der Aalst. Workflow Resource Patterns as a Tool to Support OASIS BPEL4People Standardization Efforts. *BPTrends*, 6(3):1–26, March 2008. 27

[80] Nick Russell, Arthur, Wil M. P. van der Aalst, and Natalya Mulyar. Workflow control-flow patterns: A revised view. Technical report, BPMcenter.org, 2006. `http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf`. x, xi, 23, 28, 45, 46, 47, 48, 54, 55, 56, 57

[81] Nick Russell, Arthur T. Hofstede, David Edmond, and Wil van der Aalst. Workflow data patterns. Technical report, Queensland University of Technology, Brisbane, 2004. 25, 28

[82] Nick Russell, Ter, David Edmond, and Wil van der Aalst. Workflow data patterns: Identification, representation and tool support. In Lois Delcambre, Christian Kop, Heinrich C. Mayr, John Mylopoulos, and Oscar Pastor, editors, *Proceedings of the 24th Int'l Conference on Conceptual Modeling (ER 2005)*, pages 353–368. Springer Verlag, Klagenfurt, Austria, October 2005. `http://www.workflowpatterns.com/documentation/documents/data_patternsER2005.pdf`. 25, 28

[83] Nick Russell, Arthur H.M. ter Hofstede, and David Edmond. Workflow resource patterns. 2004. 26, 28

[84] Nick Russell, Wil M. van der Aalst, Arthur H. M. Ter Hofstede, and David Edmond. Workflow resource patterns: Identification, representation and tool support. Technical report, 2005. `http://www.workflowpatterns.com/documentation/documents/ResourcePatternsCAiSE.pdf`. 26, 28

[85] Nick Russell, Wil M. P. van der Aalst, and Arthur Hofstede. Exception Handling Patterns in Process-Aware Information Systems. Technical report, 2006. 27, 28

[86] Nick Russell, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. Workflow exception patterns. In Eric Dubois and Klaus Pohl, editors, *CAiSE*, volume 4001 of *Lecture Notes in Computer Science*, pages 288–302. Springer, 2006. 27, 28

[87] Nick Russell and Wil M.P. van der Aalst. Evaluation of the BPEL4People and WS-HumanTask Extensions to WS-BPEL 2.0 using the Workflow Resource Patterns. BPM Center Report , Department of Technology Management, Eindhoven University of Technology GPO Box 513, NL5600 MB Eindhoven, The Netherlands, November 2007. 27

[88] David Schumm, Oktay Turetken, Natallia Kokash, Amal Elgammal, Frank Leymann, and Willem-Jan van den Heuvel. Business process compliance through reusable units of compliant processes. In Florian Daniel and Federico Facca, editors, *Current Trends in Web Engineering*, volume 6385 of *Lecture Notes in Computer Science*, pages 325–337. Springer Berlin / Heidelberg, 2010. 32

[89] Darius Silingas and Edita Mileviciene. Refactoring BPMN models. In *BPMN 2.0 Handbook*, 2010. 34

[90] Sergey Smirnov, Matthias Weidlich, Jan Mendling, and Mathias Weske. Action patterns in business process models. In *Proceedings of the 7th International Joint Conference on Service Oriented Computing*, Stockholm, November 2009. 7

[91] Glenn Smith. Improving process model quality to drive BPM project success. 2008. Online; last accessed April 2011. 34

[92] C. Stephenson and W. Bandara. Enhancing Best Practice in Public Health: Using Process Patterns for Business Process Management. In *Proceedings of the Fifteenth European Conference on Information Systems*, pages 2123–2134, 2007. 30

[93] Janis Stirna and Anne Persson. Anti-patterns as a means of focusing on critical quality aspects in enterprise modeling. In *Enterprise, Business-Process*

and Information Systems Modeling, 10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 8-9, 2009. Proceedings, volume 29 of *Lecture Notes in Business Information Processing*, pages 407–418. Springer, 2009. 7

[94] Sherry X. Sun, J. Leon Zhao, Jay F. Nunamaker, and Olivia R. Liu Sheng. Formulating the data-flow perspective for business process management. *Info. Sys. Research*, 17:374–391, December 2006. 35

[95] L. H. Thom, C. Iochpe, and M. U. Reichert. Workflow patterns for business process modeling. In *Proceedings of Workshops and Doctoral Consortium of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), Trondheim, Norway*, volume I, pages 349–358, Norway, June 2007. Tapir Academic Press. 29, 59

[96] Lucinéia Heloisa Thom, Jean Michel Lau, Cirano Iochpe, and Jan Mendling. Extending business process modeling tools with workflow pattern reuse. In Jorge Cardoso, José Cordeiro, and Joaquim Filipe, editors, *ICEIS (3)*, pages 447–452, 2007. x, 29

[97] Lucinéia H. Thom. *A Pattern-based Approach for Business Process Modeling*. PhD thesis, Universidade Federal do Rio Grande do Sul, 2006. 29, 30

[98] Hanh Nhi Tran, Bernard Coulette, and Bich Thuy Dong. Modeling process patterns and their application. In *ICSEA '07: Proceedings of the International Conference on Software Engineering Advances*, page 15, Washington, DC, USA, 2007. IEEE Computer Society. 59

[99] Hanh Nhi Tran, Bernard Coulette, and Dong Thi Bich Thuy. A UML-Based Process Meta-model Integrating a Rigorous Process Patterns Definition. In Jürgen Münch and Matias Vierimaa, editors, *PROFES*, volume 4034 of *Lecture Notes in Computer Science*, pages 429–434. Springer, 2006. 59

[100] Hanh Nhi Tran, Bernard Coulette, and Dong Thi Bich Thuy. Broadening the use of process patterns for modeling processes. In *SEKE*, pages 57–62. Knowledge Systems Institute Graduate School, 2007. 59

[101] Nikola Trčka, Wil M. Aalst, and Natalia Sidorova. Data-flow anti-patterns: Discovering data-flow errors in workflows. In *CAiSE '09: Proceedings of the 21st International Conference on Advanced Information Systems Engineering*, pages 425–439, Berlin, Heidelberg, 2009. Springer-Verlag. 35

[102] W. van der Aalst, K. van Hee, A. ter Hofstede, N. Sidorova, H. Verbeek, M. Voorhoeve, and M. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, pages 1–31, 2010. 10.1007/s00165-010-0161-4. 34

[103] W. M. P. van der Aalst. Don't go with the flow: web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003. 24

[104] W. M. P. van der Aalst. Pattern-Based Analysis of BPML (and WSCI). Technical report, Queensland University of Technology, Brisbane, 2005. 24

[105] W. M. P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language (Revised Version). Technical report, Queensland University of Technology, Brisbane, 2003. 24, 36

[106] Wil van der Aalst, Arjan Mooij, Christian Stahl, and Karsten Wolf. Service Interaction: Patterns, Formalization, and Analysis. In Marco Bernardo, Luca Padovani, and Gianluigi Zavattaro, editors, *Formal Methods for Web Services*, volume 5569 of *Lecture Notes in Computer Science*, pages 42–88. Springer Berlin / Heidelberg, 2009. 8

[107] Wil M. P. van der Aalst. Patterns and XPDL: A critical evaluation of the XML process definition language. Technical Report BPM-03-09, BPMcenter.org, 2003. 24

[108] Wil M. P. van der Aalst, Arthur, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, July 2003. `http://www.workflowpatterns.com/documentation/documents/wfs-pat-2002.pdf`. 23, 28, 39

[109] Wil M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera, editors. *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649, 2005. 62, 67

[110] Wil M. P. van der Aalst, Alexander Dreiling, Florian Gottschalk, Michael Rosemann, and Monique H. Jansen-Vullers. Configurable Process Models as a Basis for Reference Modeling. In Bussler and Haller [22], pages 512–518. 36

[111] B.F. van Dongen, J. Mendling, and Wil M. P. van der Aalst. Structural patterns for soundness of business process models. In *EDOC '06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, pages 116–128, Los Alamitos, USA, 2006. IEEE Computer Society. 34

[112] B. Weber and M.U. Reichert. Refactoring process models in large process repositories. In *Proceedings 20th Int'l Conf. on Advanced Information Systems Engineering (CAiSE'08)*, volume 5074 of *Lecture Notes in Computer Science*, pages 124–139, London, June 2008. Springer Verlag. 59

[113] B. Weber, S.B. Rinderle, and M.U. Reichert. Change support in process-aware information systems - a pattern-based analysis, November 2007. 8

[114] Stephen A. White. Process Modeling Notations and Workflow Patterns. On BPMN website, 2004. 24

[115] P. Wohed, W. van der Aalst, M. Dumas, A. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. In *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, chapter 12, pages 161–176–176. Springer Berlin / Heidelberg, 2006. 25

[116] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Pattern-Based Analysis of BPEL4WS. Technical report, Queensland University of Technology, Brisbane, 2002. 24

[117] Petia Wohed, Wil Aalst, Marlon Dumas, and Arthur Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In *Conceptual Modeling - ER 2003*, volume 2813 of *Lecture Notes in Computer Science*, pages 200–215. Springer Berlin / Heidelberg, 2003. 24

[118] Petia Wohed, Marlon Dumas, Arthur H. M. Ter Hofstede, and Nick Russell. Pattern-based Analysis of the Control-flow Perspective of UML Activity Diagrams. In *In Proceedings of the International Conference on Conceptual Modelling (ER*, pages 63–78. Springer Verlag, 2005. 24

[119] Petia Wohed, Nick Russell, Arthur H.M. ter Hofstede, Birger Andersson, and Wil M.P. van der Aalst. Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark. *Information and Software Technology*, 51(8):1187 – 1216, 2009. 23

[120] Christian Wolter, Michael Menzel, Andreas Schaad, Philip Miseldine, and Christoph Meinel. Model-driven business process security requirement specification. *Journal of Systems Architecture*, 55(4):211 – 223, 2009. 7

[121] Jian Yu, Tan Phan Manh, Jun Han, Yan Jin, Yanbo Han, and Jianwu Wang. Pattern based property specification and verification for service composition. In *WISE*, volume 4255 of *Lecture Notes in Computer Science*, pages 156–168. Springer, 2006. 7