ulm university universität
uulm

Ulm University | 89069 Ulm | Germany

Faculty of Engineering and Computer Science
Institute of Database and Information Systems

Exceptions and Exception Handling in Business Process Management Systems- Analysis and Classification

Bachelor´s Thesis at Ulm University

Submitted by:
Serife Öztemür
serife.oeztemuer@uni-ulm.de

Reviewer and Supervisor:
Prof. Dr. Manfred Reichert

2015

Version September 14, 2015

Abstract

During the last decade, business process management technologies have become increasingly important. The rapid technical growth causes large dynamic changes to process complexity and increases the amounts of variants. Business process executions are vulnerable to various exceptions. Mostly, the business analysts remain busy optimizing process modeling, rather than focusing on the classification of exceptions in order to optimize the processes. Contemporary exceptions are often given high priority in process modeling and are addressed in early analysis and design phases. Due to the fact that exceptions can be modeled either too rough or too precise, there is often a gap between optimal modeling and modeling all occurring exceptions.

Exceptions in business process management systems have to be specialized and merged into classifications. This details the types of exceptions affecting real world processes. This thesis has the aim to support optimized exception modeling in an early phase of business process development by analyzing the sources of exceptions and proposing classifications for exceptions. For this purpose, various process models, especially medical processes from the Ulm university clinic are critically evaluated.

Contents

1

Introduction

“There is no exception to the rule that every rule has an exception.”

James Thurber. (*08.12.1895- †02.11.1961)

1.1 Motivation

In recent years, as a result of increase in dynamics and complexity of business process models, companies require high and rapid adaptability and flexibility. In real-world processes, the durability and control of fixed workflow procedures, structures or unmanageable data processing are not sufficiently robust in the long term of execution for flexible circumstances. Usually unexpected circumstances, so called exceptions, occur not only in business processes but also in organizational processes in various institutions like production or design procedures or especially medical procedures in hospitals.

Regard the following simple real-life example: Depending on the disease stage during intestinal cancer treatment, there are three possible procedures: the operation, the chemotherapy or the radiotherapy, in which problem-free operations and procedures are required. For instance, unexpected complications like blood poisoning, incisional hernia or recurrence can occur as a result of the surgery. Due to particular exceptions, dysfunctional process instances can be the result if things go wrong, which have to be handled optimally. Unfortunately, many of such exceptions exist and if there is no effectively modeled exception handling, the execution time of whole process could be affected. Modeling of all occurred exceptions or imprecise modeling of exceptions causes methodical process errors and it is a difficult challenge to classify all of them in categories. Often, there exists a deviation between real process scenarios with widespread exceptional cases and the graphical

representation. Surely, process improvement is conceivable by categorizing the avoidable exceptions into types, which can be further optimized in the future. Potentially classification enables better process guidance in more predictable and assessable situations.

Often there is a lack of flexible adaptation and handling exceptions as well as runtime irregularities in business process management systems (BPMS). Allowing deviations or irregular activities in process modeling support robust execution.[1] Therefore, exceptions require flexible adaptation of dynamic business process as a basic prerequisite. Flexibility is becoming more and more important in BPMS. The more frequently a process has to be adapted due to changed circumstances, the less effective the traditional methods used for process improvement are. Figure 1 shows the Devil´s Quadrangle. It clearly describes the dilemma between the factors Quality, Time, Cost and Flexibility.[2]

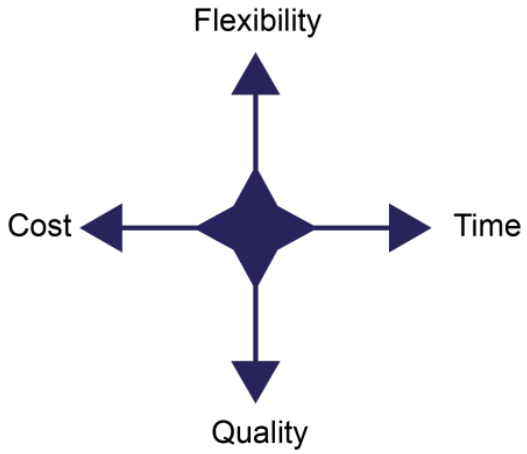


Figure 1. Devil´s quadrangle

These four dimensions describe process performance. As the factors are dependent on each other, it is necessary to take precautions not to completely disrupt the balance and coordination between the factors when optimizing one factor. The trade-off between each dimension should be dealt with in an optimal way.[2]

This inspires researchers to find pattern-based exception classification solutions for supporting optimized modeling and preventing unplanned occurrences with categorized exceptions. The famous Pareto Principle, the so called "*80 to 20 rule*" implies "*that 80% of all costs are caused by 20% of deviations and errors– the exceptions*".[3] (p. 14) For functional process design, it is necessary to have a

comprehension of functional exceptions. For this reason, the business process has to figure out how each type of functional exception should be handled. In functional modeling, the focus lies on the comprehension of essential procedures, details are composited.[3] Minimizing unnecessary events reduce costs, but to achieve this, it is essential to detect functional backgrounds while completing the process analysis, in order to eliminate dispensable expected and unexpected events.

Exception handling patterns supports flexible processes in this regard, reducing high costs. Categorization of exceptions merges similar types and helps to create automated exceptions handler, which can reduce process running, and helps avoiding interrupts or error time. High process quality can be achieved when the processes require performance faultlessly. Correct modeling is fundamentally important. In this context, exception classification and their handling methods are basic requirements for exception modeling in BPMS. Thus, time delays, dynamic process sequences or reset the process steps may not affect the practicability. Terminations and dynamic changes should be modeled as accurately as possible, in order to obtain the control over the implementation. For that reason, it is necessary to classify the type and occurrence probability of semantic exceptions.

1.2 Problem statement

As already mentioned, there are incentives for solving existing problems in the field of exception handling and flexible adaptability in business processes. In this section, some research questions help the readers to understand the overall intention and contribution of this thesis.

- **RQ 1:** *What are typical exceptions occurring every day? What are the most typical causes of exceptions?*

These two questions define the direction of this thesis. A lot of exceptions raised for similar reasons, might therefore be divided into types or might be summarized into one type to handle them together (i.e. in one go). To answer these two questions, we analyzed processes from healthcare, specifically from a women´s Hospital. Additionally we considered administration processes from the Campus Management System (the Communication and Information Center) of Ulm University.

- **RQ 2:** *How can the discovered exceptions be classified and categorized?*

This question deals with the main research problem of this thesis, which aims to classify exceptions in business processes. It is not possible to classify all types of exceptions, thus this thesis will analyze only expected exceptions.

- **RQ 3:** *How can the detected exception source be handled? What are most frequently occurring exception handling patterns?*

Expected exceptions in the processes need to be handled. An analysis is conducted to determine the exception handling patterns from various real-world domains. A view of the relation between exception sources and exception handling patterns are given in charts.

1.3 Objectives of the thesis

The general aim of this thesis is to analyze the typical exceptions occurring during business process execution, as well as their handling methods in business process management systems, and how the exceptions can be classified. The objective here is to define as many exceptions in the process models in concrete terms to have respective treatment options. For handling, it is important to know the source of these exceptions, how they relate to each other, how they can be classified and what the possible problem-solving approaches are. This includes an evaluation of how they fit into applications and a classification of the exceptions present in business process management systems.

For more than a decade, exceptions in process models have been studied in detail to understand their types and sources. Exceptions that cannot be anticipated are selected, based on certain criteria and are divided into categories. In addition, properties of the exceptions are utilized to support the proper classification. In particular, various exception-handling techniques are considered in order to identify the patterns and elements in workflows business processes causing errors. Since these patterns provide an operative support for capturing the exceptions, the most common exception patterns are analyzed and presented. Furthermore to deal with unexpected exceptions the importance of ad-hoc changes is emphasized to determine to which exceptions ad-hoc changes can usefully applied. Finally, based on often-occurring reasons of exceptions the classes and possible solution approaches are evaluated.

1.4 Outline of the thesis

This thesis is divided into five sections:

Section 1 provides an overview of the thesis consisting of scope, the motivation, problem statement, objectives and outline of the thesis.

Section 2 describes the research procedure methodology and provides an overview on data collection. By using a case study, exceptional situations from the medical process scenario are considered.

In Section 3, a general overview of the fundamentals of BPMS and exceptions is given, followed by an explanation of exception sources and their handling methods.

Section 4 describes the basic exception sources and the exception handling patterns.

Section 5 provides an analysis of exceptions sources and their exception handling patterns in BPMS. Several real-world processes and examples were examined in order to detect the exception sources and to analyze the possible handling methods.

Section 6 summarizes the main results, and gives an outlook regarding flexible handling.

2

Methodology

"Research methods shape the language we use to describe the world, and language shapes how we think about the world."

Benbasat and Weber 1996[4](p. 392)

In this section, literature study, work related methods and techniques used in the context of data acquisition and documentation of information are represented before the empirical part will be described in Section 4. The aim of this section is to give better clarity on data sources and data collection. Afterwards, an example from medical process scenario will provide an overview of a process model with exceptions.

2.1 Procedure of methodology

Figure 2 shows the process model for methodology of this thesis in sub-processes illustrated in *Signavio* using BPMN 2.0. A sub-process provides a detailed description of a sequence of the parent process and can encapsulate the complexity. *Signavio* is a process editor tool from the Signavio GmbH that supports most popular process modeling languages[5]. Different modeling languages are BPMN[1], EPC[2], BPEL[3] and YAWL[4] and DMN[5].

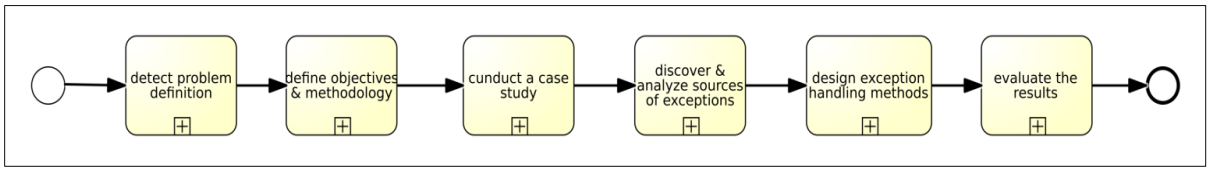


Figure 2. Process model for methodology of this thesis

[1] Business Process Model and Notation
[2] Event-driven Process Chain
[3] Business Process Execution Language is XML-based
[4] Yet Another Workflow Language
[5] Decision Model and Notation

Furthermore, the first and second sub-processes are used to define the problem statements, the aims, as well as the methodology of this thesis.
In order to get an overview at the beginning, the fundamental notions are explained. Unfortunately, there is no common definition of the term exception, for this reason various definitions from researchers are compared to clarify the minor differences. Additionally, the differences between the terms exception and error are explained.
In the third sub-process, a specific example of a case study is selected in the BPMN2.0 modeling language to introduce processes with existing exceptions.
As a further step in sub-process four “discover and analyze sources of exceptions”, the general information and data about sources of exceptions are captured and analyzed. The basic exception patterns are represented with their sources and effects. Finally, in the last sub-process, the results are evaluated and presented in detail.

2.2 Selection criteria

In regards to the classification of exceptions in business process models, exception handling is supported. This thesis is focused on analyzing of exceptions in process models, which means criteria for the analysis and setting priorities is selected. Analysts examine processes in the organizational- and operational views with different tools. These views are based on various models like the information model, the function model, the organizational models and the process model. The process model includes tasks and procedures spanning several organizations describing the operational structure.[6] We exclusively consider process models in the operational view. This means organization models and other model types are excluded. In addition, the information models are not taken into consideration. In the following section, a list of process models is presented from different sources for gathering and measuring data about exceptions. Moreover, the exceptions are not restricted to specific modeling languages. Mostly, our process sources are in BPMN2.0 and EPC notations and some are defined by textual descriptions and spreadsheets of the scenarios.

2.3 Data sources and data collection

In the following, to answer the RQ´s, collections of process reports are selected from several domains. In Table 1. **Data sources for identifying exceptions**, the process model sources for identifying exceptions are listed. On the one hand, internal process reports and models from the DBIS department of Ulm University are investigated. The core and sub-processes from the university clinic in Ulm, especially from women´s clinic inpatient and outpatient chemotherapy procedures are used.

On the other hand, the administration processes of the Campus Management System created from the Institute of DBIS in Ulm University are analyzed. The results are assessed using different statistical methods. Though most of the process notations are vary between BPMN2.0 and EPC, even UML was used in process reports. Hereafter data of process reports are listed.

Source	Publication Title and Reference	Domain	Name of Process Scenarios	Number of Process Models
1	Prozessentwurf am Beispiel eines Ablaufs aus dem OP- Bereich [7]	Healthcare	Processes in the surgical field	26 EPC
2	Prozessentwurf für den Ablauf einer statiomären Chemotherapie[8]	Healthcare	chemotherapy processes	15 EPC
3	Prozessentwurf für den Ablauf einer ambulanten Chemotherapie [6]	Healthcare	outpatient chemotherapy processes	7 textual and graphical presentations
4	Prozessentwurf für den Ablauf einer radiologischen Untersuchung [9]	Healthcare	radiological examination	9 UML
5	Prozessentwurf eines Ablaufs im Labor [10]	Healthcare	Laboratory procedures	7 UML

6	Klinische Prozesse von Caroline Streuer[5][6]	Healthcare	administrative admission, clinical admission, planning and performing diagnostic examination, preparing and performing surgery, postoperative treatment at ICU, inpatient care including preparation for fist chemo cycle, performing first chemo cycle, create epicrisis	32 BPMN2.0
7	KIZ[5][7]	Administration	Campus Management System processes from Communication and Information Center Ulm University	14 BPMN2.0

Table 1. Data sources for identifying exceptions

Source 1 comprises relevant sub-processes of surgical procedures like laboratory, radiological investigation, ordering medicines or patient documentation. Typically procedures are cooperated between two organizational units and different roles in the hospital.[7]

Source 2 consists of the entire process of the chemotherapy to prevent the development of tumors and ulcers, which cannot be treated surgically. The modeled EPC´s are textual described and supplemented with tables.[8]

Source 3 is represented in UML and includes preliminary textual descriptions. Here, the core processes with sub-process tasks are used for the implementation of outpatient chemotherapy.[6]

Source 4 includes one core and residual sub-process in the field of radiological examinations. Process procedures are mainly textual descriptions and UML models.[9]

[6] http://academic.signavio.com
[7] http://academic.signavio.com

Source 5 also can be divided into one core and residual sub-process. The processes give an insight into the procedure of laboratory investigation. Furthermore the sub-processes are significant part of inpatient and outpatient surgery and chemotherapy.[10]

Source 6 consists of processes of clinical area, especially from surgical chemotherapy. These processes are modeled in BPMN2.0 and some expected exceptions are handled as process flows or illustrated with events.

Source 7 consists of various Campus Management System processes from the Communication and Information Center of Ulm University. In these processes expected exception are illustrated in BPMN2.0.

Besides the process modeling tool Signavio, the submitted processes (from Source 1 to Source 6) are additionally designed with the ARIS Toolset[8] or with Bonapart, both are software for presentation and optimization of internal processes and structures.[6] The other sources are modeled using Signavio with BPMN2.0. Further examples for process modeling tools are the SAP-NetWeaver BPM[9] or the AristaFlow[10].

In the context of surgical procedure, the analyzed areas of relevant sub-processes are: ordering medicines, admission and discharge of patients, carrying out laboratory tests and radiology, counseling, writing physician´s letters and epicrises creation.
The documentation of process models of activities, tasks, e.g. and their exceptions in a tabular form of presentation was used. The analyzed sub-processes are documented separately in Excel sheets within their exception sources and handling techniques. The analysis results are also documented in tabular spreadsheets.

[8] Architecture of Integrated Information Systems by August Wilhelm Scheer
used for business process modeling and analyzing of organizational structures
[9] http://www.sap.com/germany/index.html
[10] http://www.aristaflow.com/

2.4 Case study: example of a medical process scenario

A case study

"(…) examines a phenomenon in its natural setting, employing multiple methods of data collection to gather information from one or a few entities (people, groups, or organizations)."

Benbasat et al. 1987[11] (p. 370)

In this section, according to this description by Benbasat et al.[11] (p. 370) we consider one example of a process model in healthcare.

Before we start with showing a process example, it is crucial to understand the basic elements of the process modeling notation BPMN2.0. BPMN has over 100 symbols[12] in its current version and is getting more and more complex with every iteration. With the support of the of the illustrated process model (see Figure 3) the basic symbols are explained.

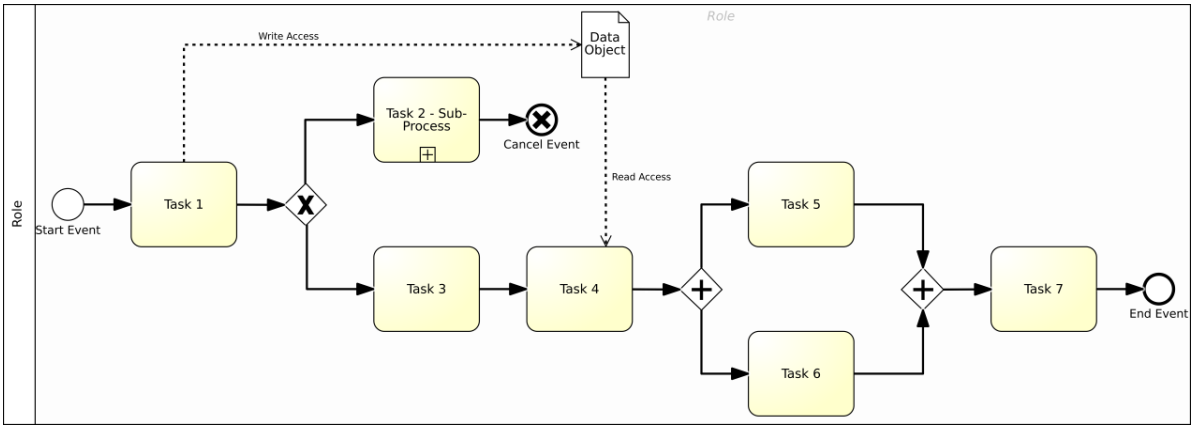


Figure 3. Process model example

Business processes include *events* and *activities*. Activities answer the important question "What needs to be done when and by whom?" and are individual work units represented as tasks. They "*describe unit of work that may be performed by humans or software application, or a combination thereof*."[12] (p. 64) They have an execution period and can consist of one singular step or represent a set of activities as sub-processes as in Task 2.[12]. An event represent "*things that happen instantaneously (e.g. an invoice has been received)*" and enables activities.[12] (p. 64) Events require

input or output, like starting or ending time specifications or arrival of messages from other instances to end the process.
In this context, the question to roles or organizational units (Org.Units) are expressed in lanes and pools. Firstly, a process is usually triggered by a *start event*. The transactions with full arrow-head shows the sequence flow. Typically the sequence flow gives the process direction point and procedures are read from left to right but can also modeled from up to down. After writing resp. creating a data object in the activity Task 1 an XOR gateway, is represented with crossed rhombus. This is representing an “either/or” option, which means Task 2 and Task 3 are mutually exclusive. After selecting the sub-process Task 2 a *cancel event* is occurring, this means the process is abruptly terminated. After selecting task 3, in Task 4 the same written data object is now invoked. Subsequently a parallel gateway, the AND type, is represented with plus crossed rhombus. The Tasks 5 and 6 are executed concurrent respectively parallel. Thereby, it must be considered that task 7 can be invoked when both activities has finished. Finally, an *end event* terminates the process. Nevertheless, the activity nodes, event nodes and control nodes are basic ones.

2.4.1 Example process of inpatient perform surgery

For a better imagination of exceptions in processes, an example process scenario is considered from the domain healthcare. In Figure 4 an overview for perform impatient surgery is given.
Once the process is triggered when the patient´s gynecologist confirms an ovarian respectively breast cancer the patient will refer to the hospital. At the hospital, relevant investigations are carried out and decided the urgency of surgery.
On the day of the operation, at around 6:30 in the morning the nurses for the intervention in his patient’s room are preparing the patient. Half an hour later, the patient is transported to the operating theatre and handed over to the surgical nurse to start the preparation for the surgery. An anesthesiologist provides the patient and (its duties include) puts a central catheter, administered the anesthetic and intubated the patient. From 7:30 the preparation of the surgical team is starting.
After completion of all preparatory work, the surgeon is starting the operation. Depending on the severity and complexity of the intervention, it can take from four up to ten hours.

In the best case of surgery, a removed tissue sample during surgery is sent to histology to be examined and interpreted by a pathologist. In order to obtain an insight into the malignancy of the tumor and thereby the disease state of the patient.
After the surgery, the patient is handed over to the post-operative care at the intensive care unit. [13] In the worst case of surgery, there are expected exceptions like complication during the surgery, patient die or failures of the surgeon. An error event catches the complication and throws an error event about the surgery. The patient die is involved in a sub-process, and after successful execution the process is terminated by a terminate event and if a surgeon perform a careless surgeon than an assistant surgeon have to check the situation.
In Figure 5, the sub-process “Perform Surgery-Patient dies” is triggered in the core process (Figure 4) and is handling an exception. This simple sub-process consists of three AND-Gateway connected tasks to issue death certificate, inform relatives and families physician and complete and archive patient record. The terminated sub-process triggers the termination of the core process.

As we can see, the three semantic exceptions are handled in a different modeling way. Events impress the question “What happened” within their sources and causes and impacts in the process branching and execution. Typical causes of events are messages, time durations, task related conditions, failures or errors.[3] Events are required for exception handling, which are explained precisely in subsequent section. Gateway influences the sequence of the process and gives a possibility for decisions of execution ways. They can be interfaces to sub-processes or organization Units. There exists AND-Gateway, OR-Gateway and XOR-Gateway and other Complex-Gateways, which can be event-based. Gateways illustrate branches and combinations in the control flow of a process model. During the exception handling, they can influence the process sequence.

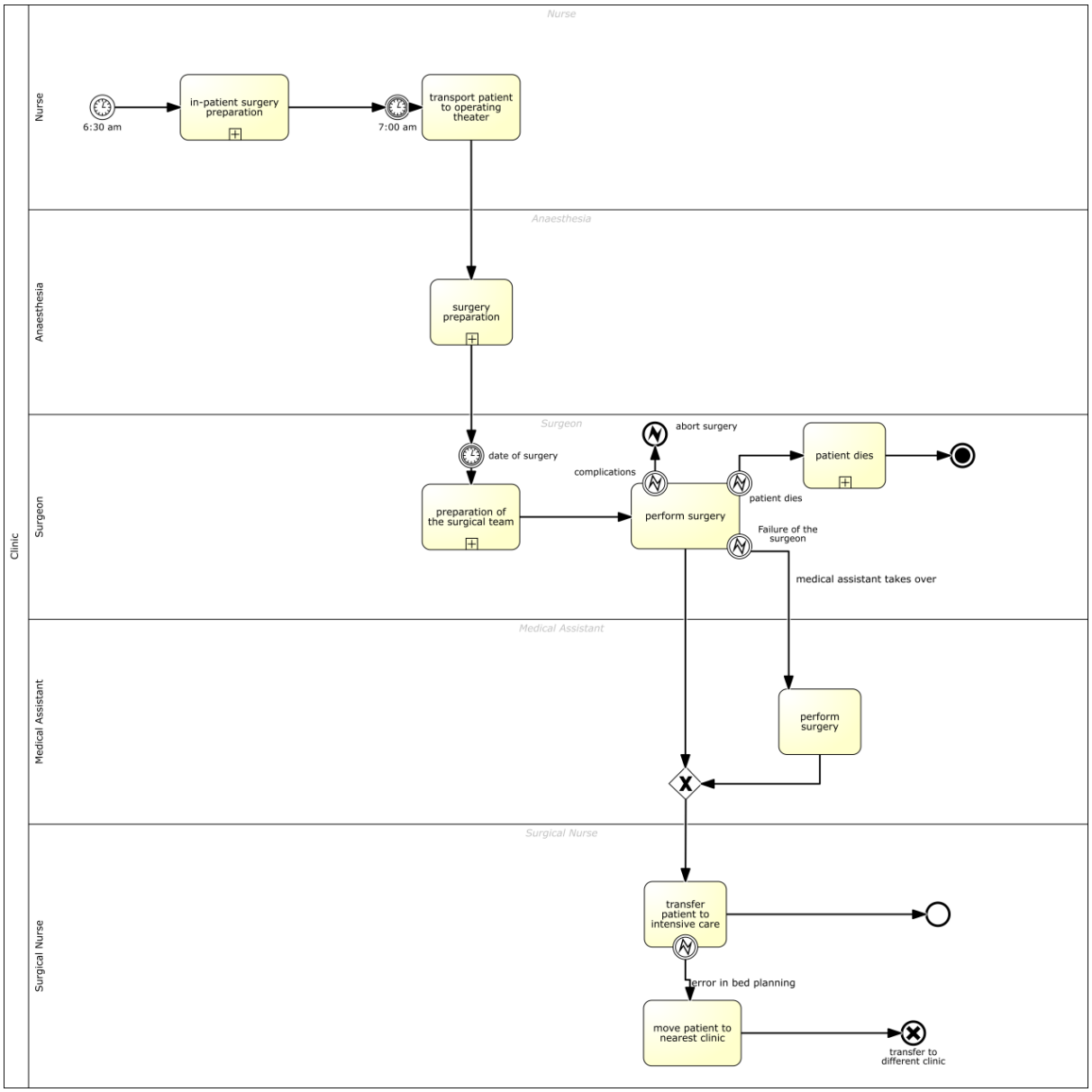


Figure 4. Perform surgery

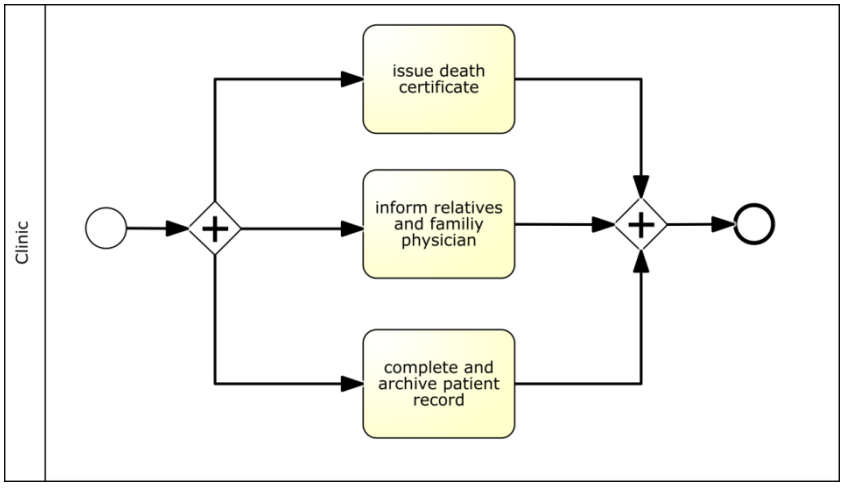


Figure 5. Sub-process perform surgery: patient dies

3

Fundamentals

Previously, Adam Smith (1723 - 1790) asserted that labor activities in production processes are more than necessary. Smith was one of the most influential pioneers for division and processing of labor activities in the 18th century during the beginning of BPM evolution.[14] In the early 20th century Frederick W. Taylor (1856 - 1915), who had proposed a set of principles, which known as Scientific Management, improved the high degree of labor division with an approach for standardization of business processes and allocation of management roles and employees.[15] The basic idea was to minimize the working time by the completion of subtasks.

To enable an orientation within the topic, this section introduces principle terminologies used in business process and business process management systems and deals with exceptions and their sources. At the outset, various definition approaches of the notions error and exception are investigated. In particular, the basic ingredients for the modeling language BPMN2.0 are explained. Hereafter, this section concludes with a discussion of exception handling methods.

3.1 Fundamentals of BPMS

Before the concept of Business Process Management Systems can be analyzed, it is important to define the term Business Process (BP). In general, each business and company is setting objectives to define success. In order to achieve these objectives, all activities and tasks between organization units have to be coordinated with each other. Automatic events can be activated as well as trigger activities. The term business process summarizes all these activities and tasks of production or manufacturing.

Bernhard Westfechtel uses an easily comprehensible definition:" *The business process comprises all activities carried out in an enterprise, including e.g. staffing,*

financing, production, marketing, etc."[16] Also, a process may consists of many actors, physical objects (e.g. materials, products), immaterial objects (e.g. electronic records) and of outcomes. Thus, in BP events, activities, actors or decisions are in coherent relation, which ultimately accomplishing a business goal.[12]

To ensure a faultless workflow in multiple BP, it is essential to manage, coordinate and optimize processes. Business process management (BPM) is dealing with this issue. BPM is included in all branches and not only in companies. To ensure zero-defect production or in education institutions, government institutions also in hospitals, where managed processes are required, there you can find BPM. The core task of BPM is continuous quality improvement, thereby minimizing costs and process runtime.[12] It is the unique discipline for efficiently improving the operational accomplishments of organization structures, but emphasizes also tools to observe, analyze, redesign or execute.[12] Techniques and concepts like process models for representations of activities or process instances, which represent particular situations, are characterizing BPM.[17] In turn BPs may be very complex, and not every activity can be automated. Therefore, dynamic process workflows systems are nowadays improved and supported by IT-Systems, which are known as Business Process Management Systems (BPMS).

A definition of BPMS is given by M. Weske[17]: "*Business process management system is a generic software system that is driven by explicit process representations to coordinate the enactment of business processes.*" Hence, there are several BPMSs with diversity methods in notations. The process modeling tools support the improvement of design and representation of business processes.

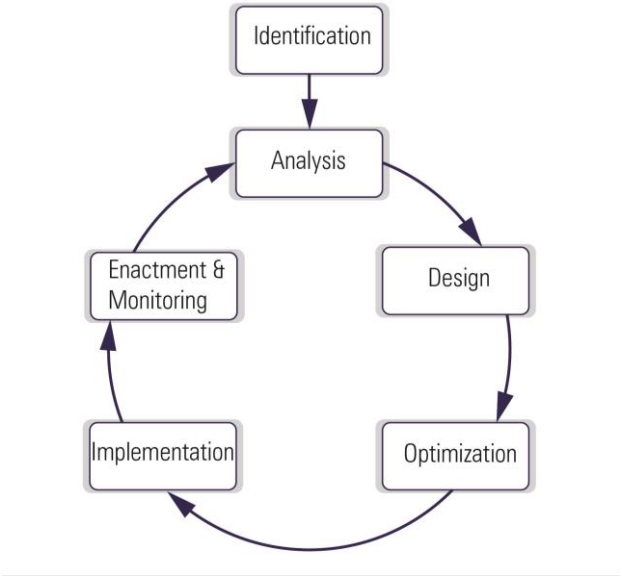


Figure 6. Business process lifecycle [18]

The BP-Lifecycle (see Figure 6) is a fundamental importance for the improvement of BPs. The aim of this lifecycle is the continuous optimization and improvement of services or products. It is consist of connected phases describing categories of BPM. After identification other repetitive phases are analysis, design, optimization, implementation and enactment, monitoring and usually evaluation.[17] To achieve the aim of the lifecycle, it is essential to phrase process oriented questions prior to identifying and analyzing relevant information. Alternatively, Tom DeMarco, a developer of structured analysis, mentioned:"*You can´t control what you can´t measure*". This is an important statement, which means before process modeling it is crucial to define process performance measures.[12] For instance, cycle time, quality rate and mostly error rate are common measures. Requirements identification or -validation can be detected in the analysis phase.[12] The objectives within analysis phase are to get an overview as complete as possible about overall processes, and to identify the participating systems alongside the interfaces. In the design phase, models of arduous processes are usually represented graphically, with internal repetitions of identification or verification. The analysis and design phases are core conditions for successful implementation, because inadvertent procedures and errors can be identified. Therefore, they should iteratively be optimized prior to implementation.[17]

The model level phase is separated in sub-steps e.g. roles and alternative paths. A successful transition from model level to process design requires a correct syntactic

and semantic view. After technical and semantic process design considerations, the transition between these is supplemented by functional exceptions and error handlings, alongside unusual alternative paths. Moreover, the design phase is well suited for system splitting, adding error handling and alternative paths. In addition, it supports an easier understanding of textual procedures via graphical representation of processes. In the implementation phase, processes are automated from as-is to to-be processes.[12]

The lifecycle (cf. Fig. 6) is a iterative process, which repeats after completion. Repetition is a precondition for accurate process execution, as inaccurate data, both inputs and outputs are controlled. The BPM life cycle is essential for the continuous improvement and consistent development of business processes. [12]

However, in particular phases of lifecycle´s and after process implementation´s, error and exception can occur like unexpected interruptions, data loss or other effects canceling processes. The gradual operations during the lifecycle enable to find out rough as well as special exceptions before implementation.[17] In this context, BPMS takes an important role in selecting and using suitable tools for particular lifecycle phases. In real world scenarios, unexpected or unlikely situations often preclude execution of procedures. For a better classification and treatment of exceptions, it is necessary to detect the sources of these extraordinary events and their possible handling methods.[19]

3.2 Fundamentals of exceptions and errors

Extraordinary circumstances in BPs can terminate or crash running processes. Hence, the comprehension of the notions exception and error is important for the understanding of the entire thesis. Depending on the context, there is a subtle difference between the notions exception and errors. Hence, it becomes essential to investigate the notions exception, and compare it with the term error. Some exception definitions are given in the following:

For M. J. Adams exception "*is simply an event that is considered to be a deviation from the expected control flow or was unaccounted for in the original process flow.*"[1]

Strong and Miller have early recognized that exceptions are "*cases that computer systems cannot process correctly without manual intervention*."[20] An exception can occur by unexpected contingencies, unsuited data or during sub-processes tasks.[20]

Exceptions are widespread in various ways: in common processes, they are generated from externalities or internalities, from interruptions or from erroneous input and output data tasks. Also, they are generated whenever there is a discrepancy between as-is processes and modeled processes.[21]

Correspondingly, M. Reichert explains, that "*exceptions are occurring events whose additional execution in WF instances is non-predictable, e.g. as in the WF models is deposited, suitable*".[19] (p. 20 sec. 2.1.2.1) According to this definition exceptions cover: spontaneous deviations of the process participants from the planned procedure, external events as a result of which a process cannot be successfully completed, errors in the execution of process steps, unavailability of resources or, last but not least, errors in software-/ hardware components. Thus, exceptions are frequently appearing events, which are not always foreseeably or predictable.[19]

Due to the plurality of definitions given above, it is difficult to establish one general definition for exceptions. Nevertheless, an exception can be described as a clearly determinable event that arises during a process execution. Finally, exceptions can be divided into one of two kinds: "expected" and "unexpected" exceptions (see Figure 7).

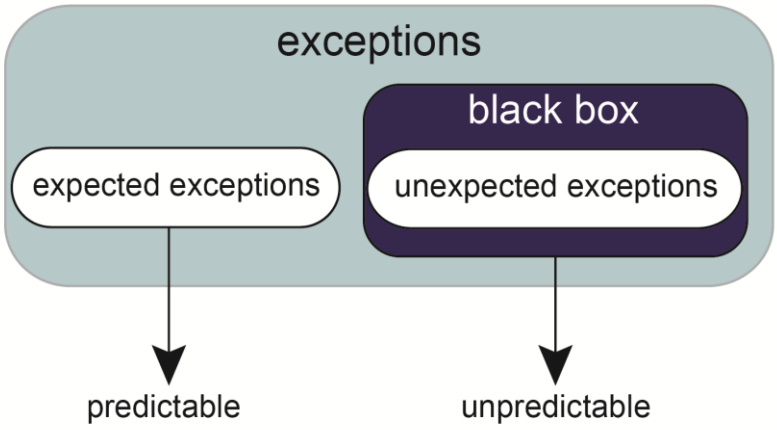


Figure 7. Expected and unexpected exceptions

However, in several cases, these two types are overlapping with each other, making an accurate distinction is very difficult. Commonly expected exceptions are

predictable and can be scheduled in a certain way. In contrast, unexpected exceptions are unpredictable and should be handled in a flexible and alternative way for example in an ad-hoc technique.[22] In healthcare, data loss or emergency patients are anticipated exceptions, whilst a building damage is an irregular appearance, and therefore unexpected. In simple terms, unexpected exceptions are considered as a Black Box as illustrated in Figure 7. They are not an integral part of process models due to the fact that until their appearance they are usually “non-existent”. If throughout the activity execution an unexpected exception occurs, then it should to be treated for current and future process models. As a result, the spontaneously appeared unexpected exception can be assumed and planned and no longer resides in the Black Box. It became an expected exception.

To summarize, an expected exception is a situation differing greatly from the norm, but still predictable and for that reason you can take measures to reduce the time-delays and enhance the quality of the process. In contrast, an unexpected exception appears spontaneously and unforeseen. Hence, it cannot be planned for in advance.

During the execution time BPs can be hampered by other difficulties, e.g. exceptions like processing defects and system crashes. These particular exceptions are called errors.[19] Error is a special kind and mostly the worst. Further examples of errors are coding or implementation failures, data transmission failures, or non-executable procedures like deadlocks. Errors can strongly affect the quality and performance of processes and systems, and should not be categorized as possible occurrences, because they have to be resolved immediately. Unfortunately, errors cannot always be treated transparently within the processes. For example, if the process was situated in transition between two events during the occurring of an error, the activity has to be restored or restarted.[19] Errors are divided into external and internal errors. Power outages or defective system components cause external errors. External ones cannot be processed clearly; it is often necessary to process by a restart and in order to ensure consistency. Contrariwise, internal errors are components of BPMS in which erroneous information has to be examined in tasks. [19]

Another differentiation of exception is deviation. Typically, deviations are unexpected and represent unusual occurring to the normal execution of the process. They can have different causes like timeouts, insert or delete of activities and the adaptability is

therefore much harder. However, deviations require a flexible exception handling like ad- hoc techniques.[19]

In practice, an analysis is always required to ensure superior control over management and process execution, and to give correct decisions for unforeseen appearances. Therefore, the analysis of semantic exceptions before modeling process is significant. Error and exception handling should be documented correctly, maybe textual in processes as well as in specifications for immediate solving. Relating to exception handling and classification, it is important to recognize the causes and types of expected exceptions.[19] Understandably, the exceptions and their sources must be detected first, in order to allow for an accurate process modeling and error-free process execution. Accordingly, the next sections deal with exception sources and their handling methods.

4

Exception sources and exception patterns

The following section describes the typical anticipated exception sources and the exception handling patterns. First exception sources are presented in detail. Furthermore, the exception handling patterns are characterized.

4.1 Exception sources

The handling of an exception is determined by its source. To abstract from concrete situations, it needs to be known which predictable exceptions are frequently occurring to identify their source. In fact, "*it is only possible to specify handlers for expected types of exception*."[22](p.5) Referring to this, this section gives a review about potential excepted exceptions sources based on RQ1 and RQ2 (see section 1). The characterization of all exceptional occurrences is difficult, therefore unexpected events are "*grouped into classes which are related by similarities that they possess in terms of conditions under which they might arise*."[22] (p. 5)

In particular, exception sources can be classified into five types[23]:

- External events
- Activity failures
    - technical
    - semantical
- Deadline expiration
- Resource unavailability
- Constraint violation

These typical sources provide a basis for handling mechanisms of exceptions, and are explained in more detail below:

- **External events**

Internal exceptions are foreseen and directly related to process management issues, like, e.g. the inability to find a component, or the non-compliance of procedure sequences or missing deadlines.[24] External exceptions are not predictable and normally triggered by the user.[23] Moreover, external exceptions are not suited in normal process sequences. In general, discrepancies between real-world and computer-modeled processes trigger externalities. They can be detected by extraordinary signals from outside the process instances or activity interruptions (e.g. a fire alarm being set off during surgery). The consequences are process break downs, cancelations of activities, or alternate measures.[22]

- **Activity failures:**

For many reasons, critical exceptions can appear during an activity execution for many several reasons. These may be divided generically in semantic and technical ones. Technical exceptions are errors resulting from the implementation or from technology, and are mostly caused by system failures, or for instance by activity breakdowns.[23] Technical exceptions are, e.g., system- related failures, an open data activity a file server is breakdown or hardware-/ software failures. Semantic errors occur in terms of unforeseen situations and are commonly caused during the activity execution. The latter can lead to mistakes and have to be handled directly.[23]

- **Deadline expirations:**

Deadline "*expirations constitute another source of exceptions whose handling might require certain action not covered by the normal flow of control.*" [23] (p. 129) Commonly, in some activities prescribed deadlines are required. Nevertheless, if during activity run-time fixed deadlines are neglected, then an exception will be raised. For example if the regular deadline for sending blood samples to laboratory tests is neglected, then the laboratory test could not be done on the same day. "*Usually the deadline indicates when the work item should be completed, although deadlines for commencement are also possible. In general with a deadline, it is also*

useful to specify at design time what should be done if the deadline is reached and the work item has not been completed." [23] (p. 5)

- **Resource unavailability**

If an activity cannot access one or more data resources during the execution then it will be impossible for the activity to proceed. Some of the problems that can possibly occur are[22]:

1. At distribution time, when a required resource cannot be found, or
2. At time after allocation, when *"the resource is no longer able to undertake or complete"*[22] (p. 5) the activity.

"*Although the occurrence of these issues can be automatically detected, they often cannot be resolved within in the context of the executing process and may involve some form of escalation or manual intervention. For this reason, they are ideally suited to resolution via exception handling.*"[22] (p. 5) Resource unavailability includes also the unavailability of human resources[23]. For example: If the surgeon is sick or unavailable then the surgery cannot be performed except a deputy is assigned.

- **Constraint violations**

Constraint violations relate to previous sources, and are defined as "*violations of constraints over data, resources, or process model elements (e.g.) activities*"[23] (p.13) Constraints require continuous observance for secure activity executions. They exclusively occur during the execution of activities, i.e., either before nor thereafter. In most cases, the implementation for dealing with constraint violations is similar to those dealing with external triggers.[22]

On the types of determined exceptions depends the various handling ways. Finally, section 4.2 deals with the exception handling methods in order to handle anticipated exception sources.

4.2 Exception handling methods

Suitable solution finding is a central requirement to handle with exceptional problems. Therefore, many possibilities for treating an exception exist. Nevertheless, the general handling strategy can be structured based on:[22]

- source and type of the detected exception
- how the detected exception can be handled
- how the other activities e.g. work items of the case will be dealt with
- what recovery step will be used to solve its effects

It is well known, that major state-of-the-art programming languages provide handling mechanisms for exceptions. For example, Java deals with exceptions by using catching and throwing principles called try-catch-throw-functions. According to this, there are also distinctive events in BP modeling for “catching” and “throwing” exceptions. The important parameters for exception handlers are causes, predictability, and context.[19] Therefore plurality of events like timers, messages, events or cancel events supports to visualize exception sources. Most process modeling tools allow direct throwing and dealing with exceptions.

In this section, the main catch and throw events are shown and described briefly in order to obtain an overview of the analyzed constructs afterwards. Furthermore, this section picks up the questions of RQ3 and describes basic exception handling patterns. As a last resort exception, patterns deal with various causes of exceptional circumstances, and the subsequent measures that need to be initiated.

To begin with, Figure 8[25] gives a general overview of some event notations in BPMN2.0 for exception catching and throwing.

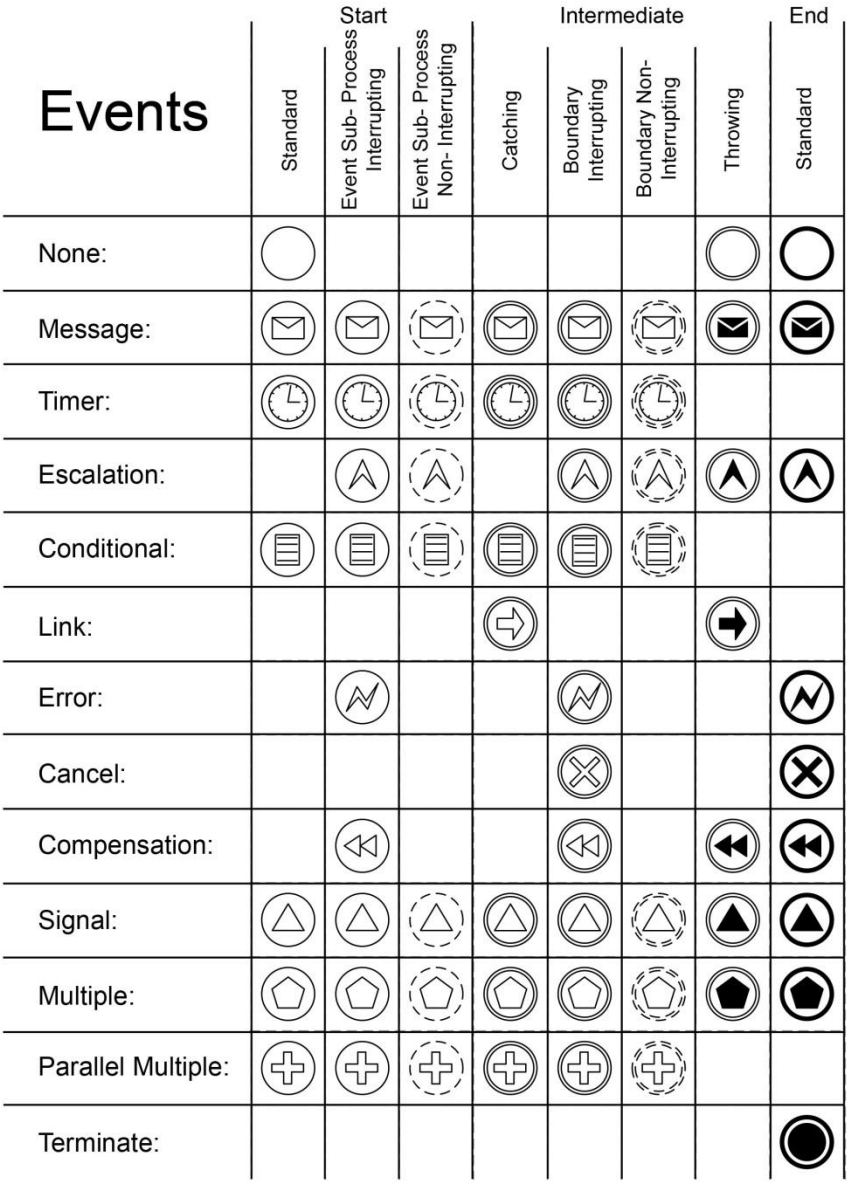


Figure 8. Overview of events in BPMN [25]

The basic event types in BPMN are start-, intermediate- and end-events. Foremost, these events influence the process execution and therefore they have to be modeled in order to increase the stability of the processes. While catching-events are specified triggers and can affect the course of the process, the throwing events are active triggers i.e. cancelations, errors or failures, messages, time delays etc.

For modeling unpredicted occurrences predominantly the error- or multiple events are used. Elsewise, BPMN2.0 does not specify which type of error is caught. Error catching is seriously modeled only with intermediate-events. Moreover, intermediate events are attached to activities for covering cases compared to if-then-else

functions. This may be exemplified: assume that task A attaches an intermediate message-event. Then, several possible cases can be executed:

1. If message-event occurs whilst A is being processed, then A will be immediately canceled, message exception is caught, and subsequent activities are executed.
2. If message-event does not occur whilst A is being processed, then the usual sequence will proceed.
3. If message-event occurs after A is completed, then it will be ignored.

Using the correct catching and throwing event for typical anticipated exceptions is important for the modeling and handling of exceptions. Furthermore, the knowledge and understanding of event notations is necessary to use the suitable handling patterns. The events in Figure 8 are summarized in the following Table 2 [25]:

Event Name	**Explanation**
None:	Start points or state changes can be represented by undefined- events.
Message:	Message events are used to receive and send messages or information.
Timer:	Start points or activities depend on time by using time intervals or time-outs.
Escalation:	The responsibility rises to a higher level.
Conditional:	The process proceeds if some conditions are fulfilled, or to reacts changed conditions like business rules.
Link:	Two link events can be used to permit alternative to a sequence flow.
Error:	This event shows a catching or throwing error state or exception.
Cancel:	Cancel-events react and triggers to transactions and terminate activities.
Compensation:	Handles or triggers compensations.
Signal:	A signal can be carried out several times between two processes.
Multiple:	Catches and throws several (summarized) events. Multiple is often used in catching exception and has XOR semantic.
Multiple Parallel	Catches parallel events and has an AND semantic character.
Terminate	Terminate event triggers the immediate termination of a process.

Table 2. Overview of events in BPMN and their explanations

On the whole, exception sources pose specific requirements to handling mechanism. Initially, exceptions must be detected immediately. Then finding out the type of exception needs to be identified.[22] After that, "*for anticipated exceptions, standard exceptions handlers can be defined. That is usually not possible for the unanticipated ones.*"[26] (p. 416) Hence, exception handlers solve expected situations by analyzing and documenting pre- and post-conditions from similar circumstances. Despite the patterns have been developed to treat common exceptions and failures correctly. These instructions provide guidance for the modeling aspects of exceptions.

Exception handling is separated into exlets, ad hoc changes and patterns. The Exlets and Ad hoc changes are for flexible handling in exceptions services. The main exception patterns for anticipated exception are described in detail in the following sections.

4.3 Exception patterns

Starting with the simple question: "What are patterns?" it is recalled to have an abstract level for offering a wide range of applications areas. Patterns describe schematic solutions for a category of related problems. Generally, patterns are described by their name, problem statement or application area. Moreover, descriptions and restrictions are discussed. In some cases some examples of patterns are given as well.

Together with exceptions, the exception patterns used to handle of expected circumstances. They designate corrective measures to avoid the consequences of exceptions and are, therefore, necessary for continuous and robust process execution.[22] Moreover, patterns assist to remove failures or circulations, which can reduce complexity in process modeling. Another benefit is the reusability of proven solutions. It is generally known that a large number of Workflow Patterns are exist. General BP patterns are further divided in control flow patterns, data-flow patterns, resources and the exception patterns, which constitute solutions for frequently performed or recurring work processes and problems.[27]

Exception patterns facilitate dealing with exceptional sources. For this reason handling possibilities are considered. In this section, main exception patterns are described using various business modeling tools, in particular BPMN2.0.

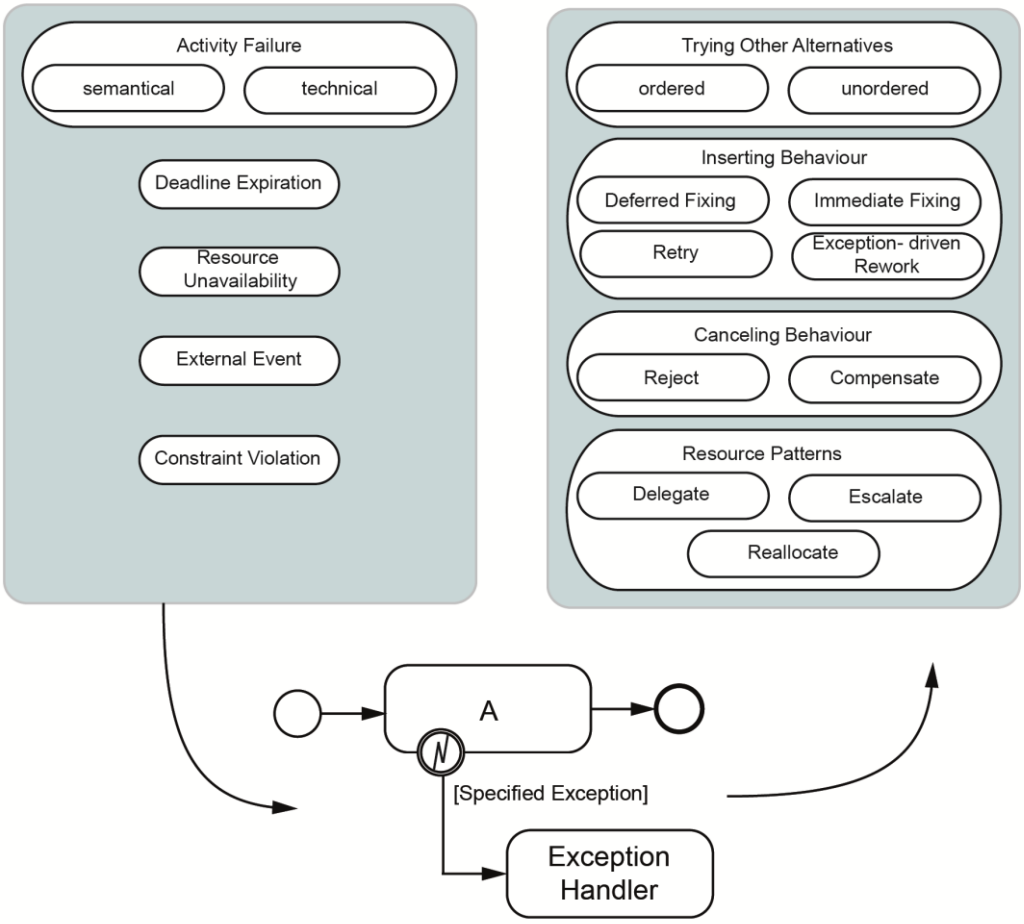


Figure 9. Exception sources and exception handlers[23]

Figure 9 [23] illustrates the relations between detected sources and their exception handlers. During the run time special handler's deal with various predicted exceptions. These handlers are prescribed procedures for solving the problems arising from extraordinary events.[22] In BP modeling, three categories for exception handling patterns exist, which deal with anticipated exceptions. In the following, a detail explanation is given in tables:[23]

- Trying alternatives patterns
    - Ordered alternatives pattern
    - Unordered alternatives pattern
- Adding behavior patterns
    - Immediate fixing
    - Deferred fixing

    - Retry
    - Exception-driven rework
- Canceling behavior patterns
    - Reject
    - Compensate

4.3.1 Trying alternatives

One possible way of dealing with exceptions during activity run-time is to try alternatives. If during ordinary processes some exceptional events prevent the execution of an activity, then alternate activities should be tried, either in sequence or non-sequence, until no more exceptions exist.[23]

These *trying alternatives* patterns can be grouped, either in *ordered alternatives* pattern or in *unordered alternatives* pattern.[23] Both alternatives patterns can be compared with the if-then-else construct in common programming languages: they have Boolean condition, means if alternative is false, then the next alternative will be tried, but if it is true first alternative is successfully completed.[28] Tables 3 and 4 summarize the characteristic of both patterns.

Pattern Name:	**Ordered alternatives pattern**
Description:	Multiple ways to proceed a process activity exist a strict expiry sequence.[28] The exceptions are part of the normal process and alternatives have to be provided when the normal sequence fails.[28]
Implementation:	This pattern is used whenever the order of the alternative ways is predefined. The exception handler calls the first alternative, A, which by itself uses another catching of exception that calls alternative B, this including another catching of exception that calls alternative X and so on. After all alternatives are processed the handler will terminate by throwing NoMoreAlternativesException.[28]

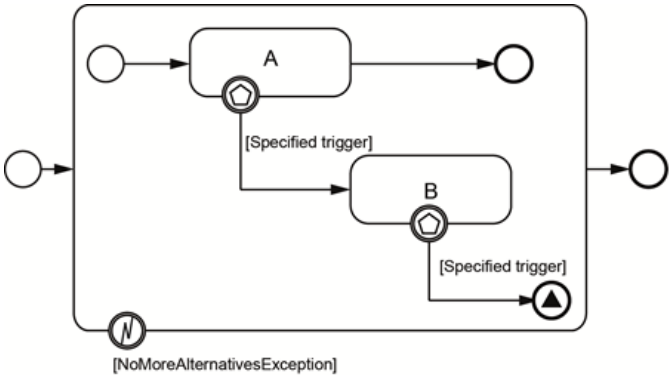
Example:	 Figure 10. Ordered alternatives[23] "*By default, activity A is executed. If A fails, activitiy B will be alternatively executed. If B also fails no more alternatives exist and a NoMoreAlternativesException is thrown and then propagated to the higher level sphere.*"[23] (p. 134)
Problems/ Restrictions:	- Appropriate knowledge about the order of exceptional cases required - Problem of never-ending alternatives is possible - Limited flexibility

Table 3. Ordered alternatives pattern

Pattern Name:	**Unordered alternatives pattern**
Description:	Unordered alternatives are used when decisions are made during the run-time[23] and the testable alternatives are irregularly free selectable.[28]
Implementation:	This pattern is used whenever there are various options and the order of alternatives is unknown.[28]

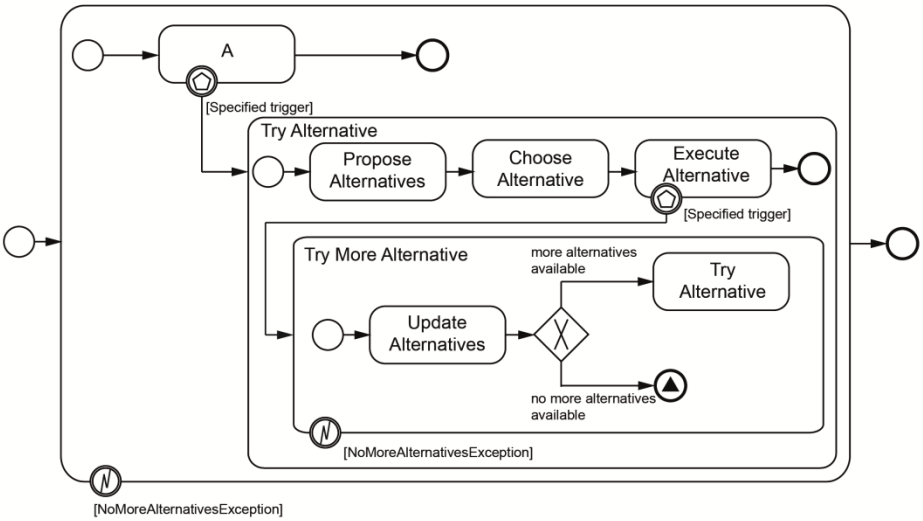
Example:	 Figure 11. Unordered alternatives[23] Figure 12 outlines what happens if the normal state activity A fails. Then "*a set of alternatives will be proposed from which one alternative has to be chosen*."[23] (p. 135) If the first alternative fails, then another alternative will be chosen. Finally, if all have failed NoMoreAlternativeExceptions is thrown.
Problems/ Restrictions:	- Can lead to an infinite loop - Can lead to a quite complex illustration of processes, because each alternative contains substructures[28] - Limited flexibility

Table 4. Unordered alternatives pattern

4.3.2 Adding behavior

Another possibility to handle exceptions is by processing additional activities, called the *Adding Behavior*.[23] This exception pattern contains *immediate fixing*, *deferred fixing*, *retrying* and *rework* patterns.

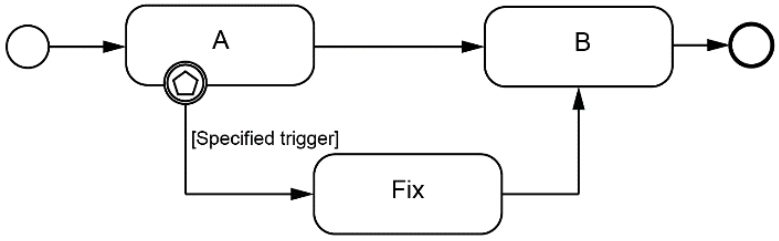
Pattern Name:	**Immediate fixing pattern**
Description:	Here, the normal sequence is interrupted, and the caught exception is immediately handled by performing an additional activity.[23]
Implementation:	This pattern is implemented by setting a trigger for exceptions. If the specified trigger is activated, then the normal sequence is supplemented by additional paths and activities for direct handling.
Example:	 Figure 12. Immediate fixing[23] Figure 12 depicts a particular triggered event while the normal flow is proceeding. Then, the "*normal flow will be interrupted and the process will continue with the exceptional flow."* [23](p.136)
Problems/ Restrictions:	- Applicable only to high probability of exceptions - Limited flexibility - Only successful on suitable level of the calling hierarchy

Table 5. Immediate fixing pattern

Pattern Name:	**Deferred fixing pattern**
Description:	In deferred fixing, the normal sequence is proceeding, while the exception is noticed, but handled afterwards.[23]

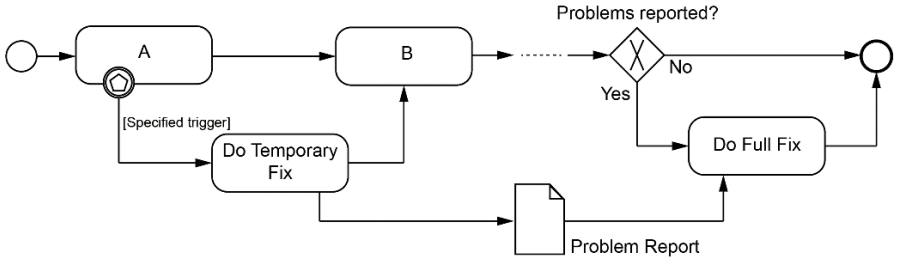
Implementation:	If the exception information needs only to be registered or not dealt immediately, then the handler can save the exception for later or exceptions are dealt in another sub-processes.[28]
Example:	 Figure 13. Deferred fixing[23] The process model (Figure 13) shows a handler with a trigger being activated during activity A. Initial fix is executed and a report is filed afterwards a gateway asks for reported problems. If there were any, full fixing is performed.[28]
Problems/ Restrictions:	- Handles exceptions temporarily - Limited flexibility

Table 6. Deferred fixing pattern

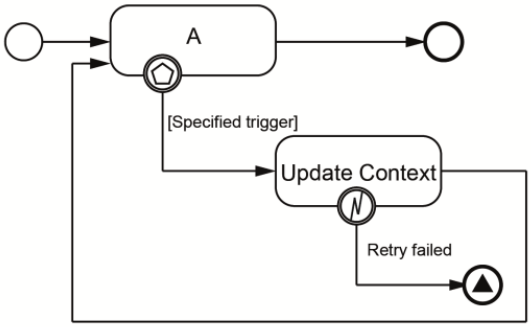
Pattern Name:	**Retry pattern**
Description:	This pattern can be used, if a activity should be reattempted in a moment after an exceptions is occurred.[23]
Implementation:	If activity A is triggering an exception, then Update Context is tested. During the context-update a further exception can be caught and handled in other possibilities. If retry is unsuccessful, then sequence can terminate.
Example:	

Figure 14. Retry[23]

	If the secretary writes minutes incorrectly, the documenting of minutes should be tried again. If email delivery failure exists, then sending email should send again.
Problems/ Restrictions:	- Limited flexibility - Infinite loop when retry is not checked.

Table 7. Retry pattern

Pattern Name:	**Exception-driven rework pattern**
Description:	Some exceptional situations may be solved by repetition of further activity at all times.[23] Usually this pattern represents the general retry.[28]
Implementation:	This pattern is used in the same situations like retry, but dealing with exceptions is time-independent, and the repetition of work is executed at later process points.
Example:	If a resource is unavailable for activity success but obtaining the resource will take time, and in the meantime, other activities can proceed. Then this activity can be retried to in another time.
Problems/ Restrictions:	- Considers externally triggered exceptions - Aftereffects can occur (like depended decision of other activities from the issue)

Table 8. Rework pattern

4.3.3 Canceling behavior

A further option of dealing with exceptions is the *canceling behavior patterns*. Either the *reject* pattern, which breaks the executions of sub-processes or the process itself, or the *compensate* pattern, which reverse finished activities or parts of processes.[23]

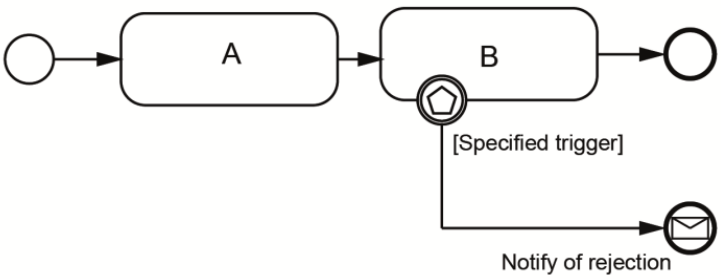
Pattern Name:	**Reject pattern**
Description:	This pattern generates an access barrier either to activities, or process parts.[28]
Implementation:	Normally an end-event is thrown a detected exception is remarked.[28]
Example:	
	Figure 15. Reject[23]
	If medical examination is performed, but the patient is not available, then the process is terminated and non-availability of the patient is indicated.
Problems/ Restrictions:	- Repetitions are not possible - Limited flexibility

Table 9. Reject pattern

Pattern Name:	**Compensate pattern**
Description:	In some cases, part of the process or finished activities must be repeated. This pattern defines activities, which have to be repeated and replaces activities for compensations.[28]
Implementation:	This pattern is used, when the result of an activity is unknown or unforeseeable. It is implemented as a sub-process with double borderlines and an attached compensation-event that undoes activities.[28]

Example:

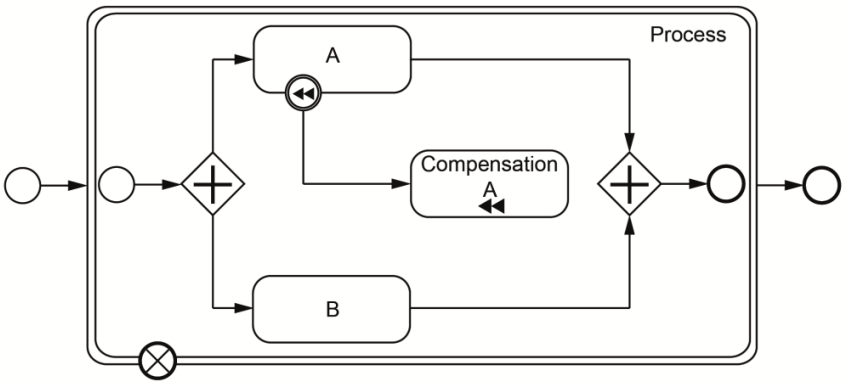


Figure 16. Compensate[23]

	In some cases, if fixed deadlines for activity termination exist but time delays occur, then all reservation or medical appointments have to be compensated.
Problems/ Restrictions:	- Problematic when all activities need to be undone at once - Can lead to complex process models - Limited flexibility

Table 10. Compensate pattern

In conclusion, all aforementioned exception patterns are practically applicable for behavioral changes at process models.[23] In addition, further pattern exist:[27]

- Resource patterns
  - Delegate
  - Escalate
  - Reallocate
- Flexible work item handling patterns
- Handling exceptions with exlets

A detailed review of flexible work item handling patterns and exlets are beyond the scope of this thesis.

4.3.4 Resource patterns

Extensions made to work distribution patterns are called the *Resource Patterns*. These resource patterns are further divided in *creation patterns*, *push patterns*, *pull patterns*, *auto-start patterns*, *visibility patterns*, *multiple resource* patterns and *detour* patterns.

Altogether, 43 resource pattern exist, but, in the context of exception handling only selected *detour* patterns will be considered.[29] *Resource Patterns* are resource-related exception handlers that give solution for exception handling during the activity run-time.[23] In other words, they are more independent from behavioral changes compared to the above-mentioned patterns. If an exception is detected during the activities execution´s, then its handling depends "*on the current state of execution of the work item*".[22] (p. 7) In some cases, an immediate action is required to solve the problems during an activity´s executions. The resource-related patterns comprise the delegation, escalation, reallocation and deallocation exceptions.[23] (p. 140) Thereby the understanding of work item lifecycle is required. The work item lifecycle offers "*the basis for determining what options exist for handling a work item in a given state when an exception is detected*."[22] (p. 6) A work item- , activities or tasks, can pass through the states: *offered*, *withdrawn*, *allocated*, *started*, *failed,* or *completed*. Figure 17 depicts these states.

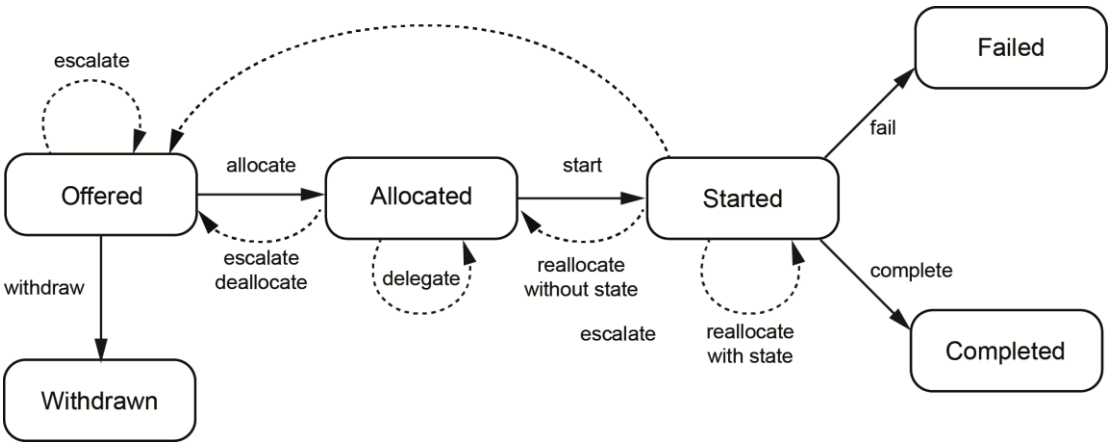


Figure 17. Resource-related exception handler[23]

The implementation of these patterns depends on the particular cases and the current state of the activities. The actual and target state can be understood in the

Figure 17 by following of the arrows. The arrows present actions and the squares present the possible state before or after dealing with exceptions. Moreover, the details about the *Resource Patterns* are explained.

Pattern Name:	**Delegation pattern**
Description:	A resource assigns an older work item to another resource.[23] The opposite would be reallocating a work item.
Implementation:	• Actual state: Allocated • Target state: Allocated
Example:	If a person (resource) R has been assigned to an activity, but R is not available, then R is replaced by a deputy (i.e., an alternative resource).
Problems/ Restrictions:	- Limited flexibility

Table 11. Delegation pattern

Pattern Name:	**Escalation pattern**
Description:	A possibility to handle a dead or delayed activity is to offer or allocate it to one or more resources.[23] The treatment is more intensified.
Implementation:	• Actual state: Allocated, Offered, Started • Target state: Allocated, Offered
Example:	If an activity is out-of-time, then dealing with it will be extended transfer another resource.
Problems/ Restrictions:	- Limited flexibility

Table 12. Escalation pattern

Pattern Name:	**Reallocation pattern**
Description:	The reallocation pattern can be divided in *Stateful Reallocation*

	and *Stateless Reallocation*. If the resource reassigns an activity to another resource, but the current state is retained, then reallocation is stateful. Otherwise, in stateless reallocation, the activity is restarted and the current state changes.[23]
Implementation:	• Actual state: Started • Target state: Started, Allocated
Example:	If termination deadline of activity has passed, then a higher responsible person reallocated, to complete the activity.
Problems/ Restrictions:	- Limited flexibility

Table 13. Reallocaation Pattern

Pattern Name:	**Deallocation Pattern**
Description:	Deallocation is used, if a resource "*makes a previously allocated work item available, i.e. the work item can be offered to other resources*."[23] (p.141)
Implementation:	• Actual state: Allocated • Target state: Offered
Example:	If the treatment results of patients need to be deallocated to other hospitals for further treatments.
Problems/ Restrictions:	- Limited flexibility

Table 14. Dellocation pattern

Other possible work item states are the *Completed* and *Failed* states, both included in the Figure 17. An activity is completed, if the standard state is fully terminated. Yet, if the current state is restricted to finish, maybe because of failures, then the activity fails.[23] The implementation of resource-related pattern depends on semantic exceptions and on individual situations. The illustration of exceptions can be done resorting to previously mentioned patterns, events, or sequence flows.

5

Analysis

“The curiosity is always the top priority of a problem to be solved.”

Galileo Galilei (*15.02.1564- †08.01.1642)

This section provides an analysis of exceptions sources and their exception handling patterns in BPMS. Several real-world processes and examples were examined in order to detect the exception sources and to analyze the possible handling methods. The evaluation procedure of the examined processes is presented in order to address the RQ1, RQ2 and RQ3. Finally, the results of the analysis are illustrated with the help of charts.

5.1 Techniques and methods

This subsection describes which techniques and methods are used to identify the exceptions. In real-world processes there are unlimited exception cases, which can occur during the process run-time. Obviously, the evaluation of all occurring exceptions sources would go beyond the scope of this thesis. Therefore typical sources of anticipated exceptions and as well as their handling through patterns are analyzed.

High amounts of processes in various BP modeling languages are investigated in order to determine the frequencies of the occurrence of various sources and their corresponding handling patterns. Therefore exceptions sources from different domains - modeled with different BPMS tools - are analyzed and categorized. The data sources from the different domains for identifying exceptions were shown in Table 1 (see section 2). In total 110 process models obtained from analyzing healthcare and administration domains were considered. Five core-processes were excluded from evaluation due to the fact that they give a general overview of sub-processes. As a result, 105 sub-processes were examined in more detail.

The documentation of process models of activities, tasks, e.g. and their exceptions in a tabular form of presentation was used. The analyzed sub-processes are documented separately in Excel sheets within their exception sources and handling techniques. The sheets summarize the results of the analysis and assist in the evaluation of data sets. Several criteria were defined to identify the exceptions and handling patterns. The exception sources were divided into internal and external causes. In addition as basis for further studies, organization units, roles referred to activities, event and gateways were detected. The resulting tables can be found into the attached CD-ROM. On the basis of these data sets, frequency analysis were performed and illustrated. The results are described in the following section 5.2.

5.2 Classification and results

As has already been mentioned, the exceptions sources can be divided into anticipated and unanticipated exceptions. In this study, only the anticipated exceptions were handled. Therefore, to get an overview before presenting the results, Figure 18 visualizes the exception sources and classifies the sources into types. Reichert and Weber[23] observe in their book that expected exceptions can categorized in five potential sources: Activity Failure, Deadline Expiration, Resource Unavailability, External Event and Constraint Validation.

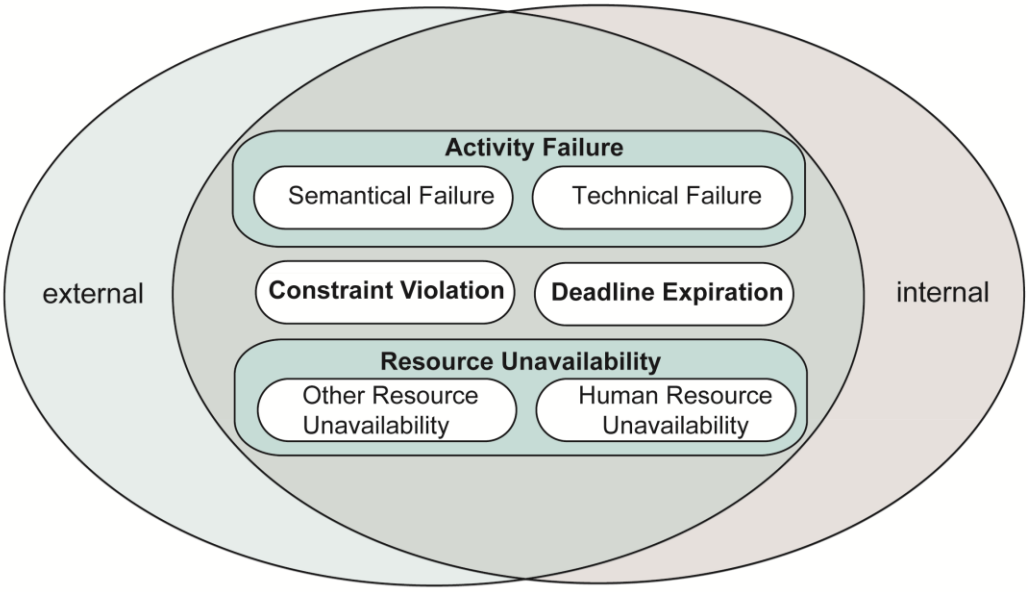


Figure 18. Exception sources and classification

The data analysis based on these sources. In general, to determine external events, a discussion with process designers is required. Therefore, this exception type is taken out of consideration. The real-world processes were investigated and allotted to the exception sources shown in Figure 18. It shows that every anticipated exception source can be triggered internally or externally. In addition to the separation of Activity failure into semantic and technical exceptions, the Resource Unavailability was divided into Human – and Other Resources Unavailability. For example, absent employees belong to Human Resources and missing data or item to Other Resources.

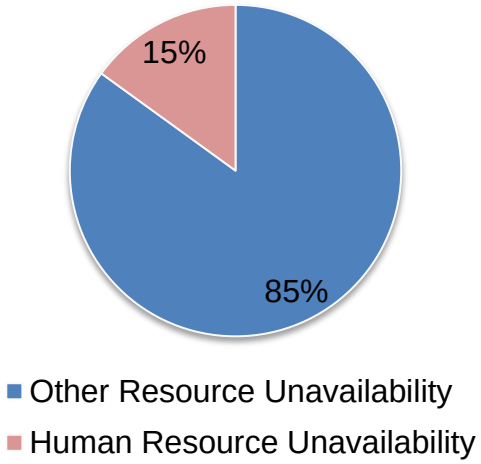


Figure 19. Percentage of other resource- and human resource unavailability

Figure 19 shows the percentage frequencies of other resources - and human resources unavailability in both analyzed domains. 85 % of resource unavailability are caused by other resource unavailability. Although human resource unavailability plays an important role in healthcare processes, only 15 % of the resource unavailability consists of human resource unavailability. The main reason for this result could be that automated processes replace nowadays activity executions.

In order to answer the RQ1 the number of occurrences of the various exceptions were determined and illustrated in Figure 20 regardless of the modeling language and the domain. Activity failure occurs most frequently in these real-world processes with 337 times. For that reason, activity failure can be defined as the core-exception type. Resource unavailability also plays a significant role within the exception types. The lowest number of occurrences was shown for deadline expiration. The reason for this may be the fact that in most of the investigated processes the time deadlines are not modeled. Hence, it is very difficult to detect all deadline expirations

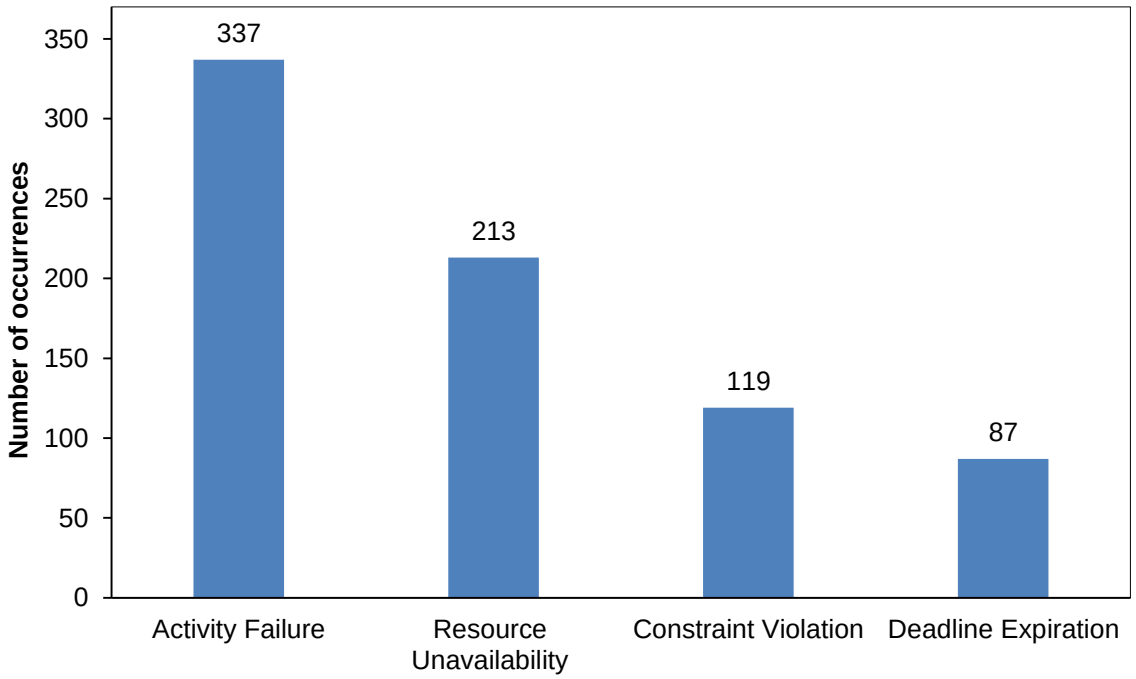


Figure 20. Number of occurrences of exception sources

A subdivision of activity failure into technical failures and semantic failures is illustrated in Figure 21. Activity failures consist of 83% semantic failures. This seems to be reasonable as a high number of semantic failures are frequently caused by incorrect or faulty executions of tasks in the investigated domains. Whereas only 17 % of activity failures caused by technical failures. In contrast to semantic failures, technical failures can iteratively be optimized after the first occurrence.

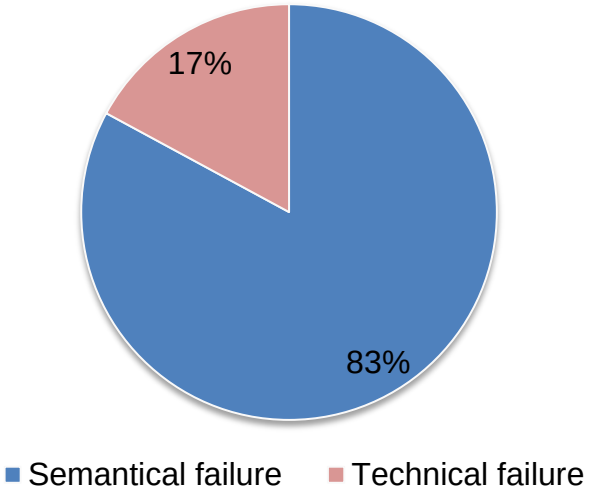


Figure 21. Activity failure- semantical and technical failures

In the next step to answer the RQ3: "*What are most frequently occurring exception handling patterns?*", the classified sources of exceptions were matched into basic exception handling patterns. Figure 22 shows the general occurrence number of exception patterns.

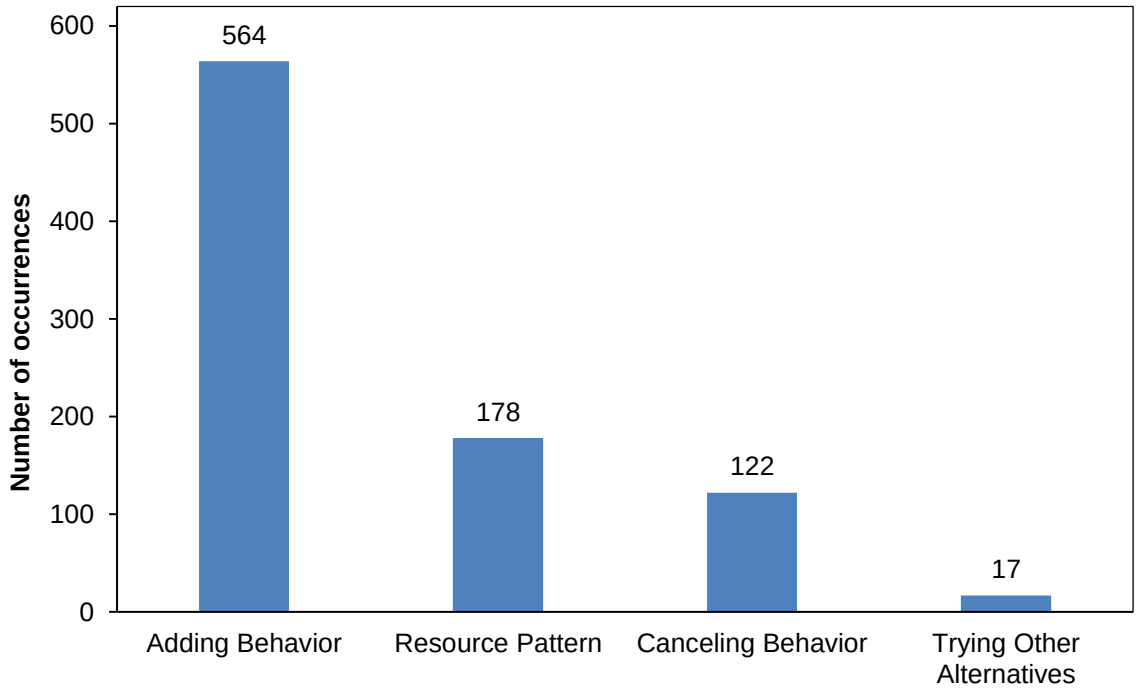


Figure 22. Number of occurrences of exception handling patterns

In some cases, exceptional sources could be handled by more than one pattern. Thus, the occurrence number of the possible matched patterns is 944. Overall, it can be noted that the adding behavior patterns occur most frequently with 564 times. This indicates that most of the exceptions can be treated by additional activities like *immediate fixing, deferred fixing, retrying and rework patterns*. It is more or less obvious why there is a high occurrence of adding behavior patterns, as this seems to be a usual reaction to handle exceptions during process execution. Resource pattern occurred in second place with 178 times. This is due to the fact that in the healthcare domain the handling of human processes need changes with regard to the resource aspect. Another interesting observation is that canceling behavior is less common (122 times) because it can cause the cancelation of the whole process or undoing the completed process. The trying alternatives show the lowest number of occurrences. The reason is that trying alternative patterns require knowledge about all possible alternatives.

5.3 Detailed considerations

In this section, the evaluation results are analyzed in detail. To deal with RQ3, the question is how each exception source can be specified by the exception handling patterns. Therefore, the relation between each exception source and their possible handling ways are illustrated Figure 23 and 24. The exception sources activity failure and were investigated as representative examples.

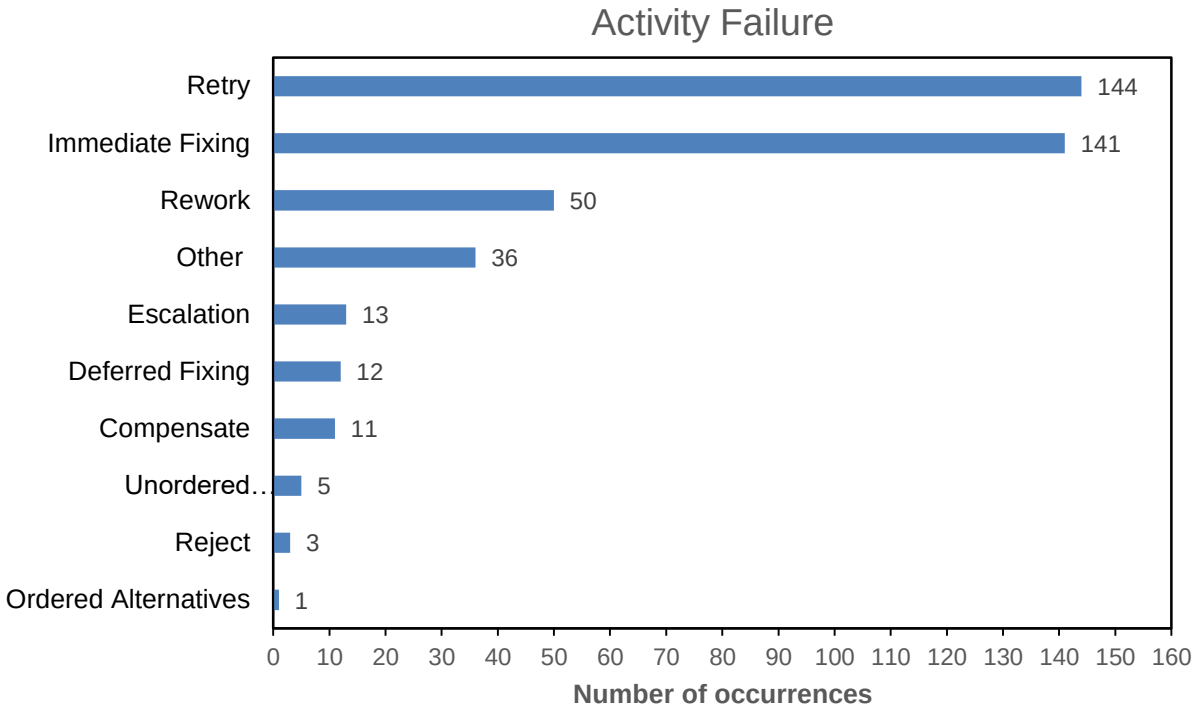


Figure 23. Activity failure handled with each exception pattern

It has to be mentioned that more than one exception handling pattern can allocated to particular exception source. For 337 detected activity failures 416 possible handling patterns were allocated, that means at least 79 sources could be dealt in two or more ways. For healthcare and administration domain can be noted that the patterns retry and immediate fixing occurs most frequently in activity failures. It makes sense because immediate fixing and retry can appear together. In the real world, exceptions are directly tried to solve or after fixing retried again. As the third most commonly occurring pattern is rework. Rework handles exceptions in a similar way, but it is less limited and independent of time. To summarize, the type activity failures was usually handled by using adding behavior patterns.

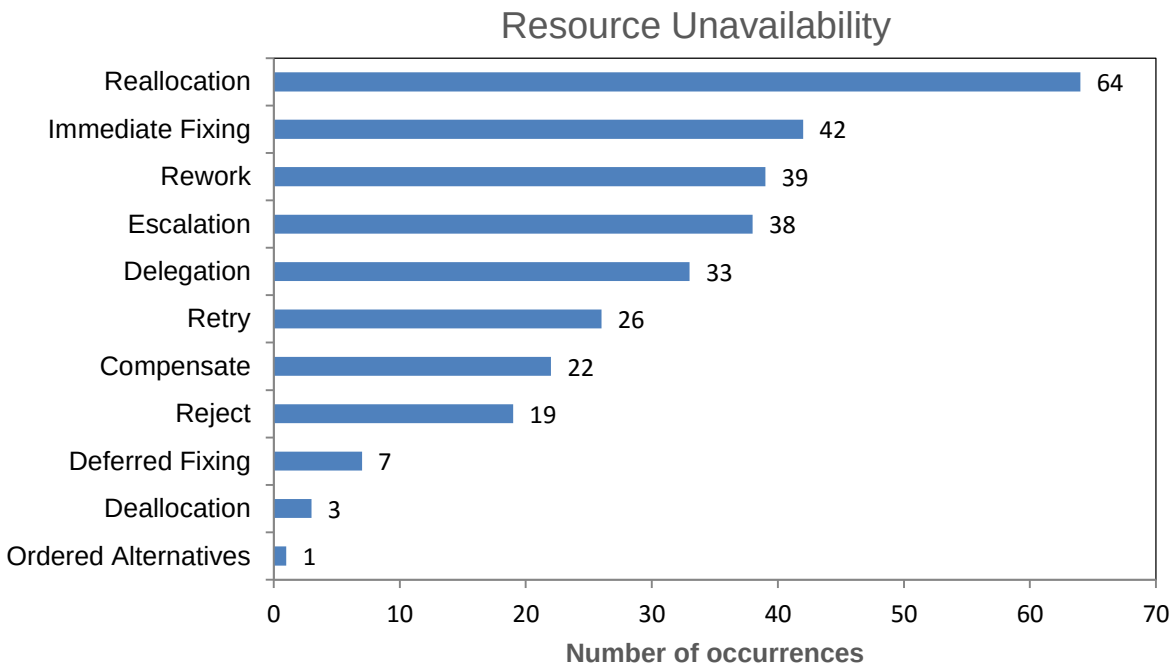


Figure 24. Resource unavailability handled with each exception pattern

Figure 24 shows the exception type resources unavailability and the corresponding possible handling patterns. In comparison to activity failure, in resource unavailability reallocation is the most used pattern. The reason for this is that the lack of resources cannot always be handled immediately. Nevertheless, the first priority in processes is the direct treatment. Therefore, immediate fixing (42) comes on the second place. In

summary, the resource-related patterns (Delegation, Escalation, Reallocation and Deallocation described in section 4.2) make in sum 138. Adding behavior patterns occur this time in second place with 114.

6

Conclusion and Outlook

This thesis investigated the exception sources and exception handling in BPMSs in real-world processes. In summary, a considerable effort was spent in identifying the exception sources. In total, 110 processes from two different domains were analyzed and approximately 1500 possible expected exceptions were detected. The exceptions were categorized into sources and were specified by basic exception handling patterns.

It has been showed that activity failures occur most commonly in the investigated domains. They consist mainly of semantic failures. The reason for this is that in investigated healthcare- and administration processes, exceptions are frequently caused by incorrect or faulty execution of tasks.

Subsequently, the classified sources of exceptions were matched into basic exception handling patterns. The adding behavior patterns appear most frequently. It seems to be a common reaction to handle exceptions during process execution. This results also corresponds to the findings of Lerner et al. [28]. They have also analyzed processes from different domains- medical and digital government domains. Accordingly, they have also showed that immediate fixing belonging to adding behavior occurs most frequently in both domains.

Furthermore, the dependence between exception sources and the related handling patterns was investigated. Activity failures were usually handled by using adding behavior patterns. Contrary to activity failures, resources unavailability were most commonly treated by resource-related patterns. The reason for this is that the lack of resources cannot always be handled immediately.

In conclusion, these basic patterns are still not flexible enough to handle suddenly appearing occurrences immediately. Many strictly structured processes, that are no longer current, need a long procedure for modification of the process structure. For

handling of unanticipated exceptions, adaptability and flexibility are required in order for run-capable processes. Emphasizing the importance of flexibility, additional flexible exception handling ways were developed. The wide range of flexible handling can be separately considered in further intense evaluations. The evaluated data can be used as basis for the studies.

List of figures

List of tables

Bibliography

[1] M. J. Adams, “Facilitating Dynamic Flexibility and Exception Handling for Workflows,” Queensland University of Technology, Brisbane, Australia, 2007.

[2] S. Sadiq, P. Soffer, and H. Völzer, “12th International Conference: BPM 2014.” Springer Verlag, Haifa, Israel, pp. 151 – 160, 2014.

[3] T. Allweyer, *BPMN 2 .0 - Bussiness Process Management and Notation: Einführung in den Standard für die Geschäftsprozessmodellierung*, 2. ed. Books on Demand GmbH, 2008.

[4] I. Benbasat and R. Weber, “Research Commentary: Rethinking ‘Diversity’ in Information System Research.” Australia, pp. 389 – 399, 1996.

[5] “Signavio GmbH,” *05.08.2015*. [Online]. Available: http://www.signavio.com/de/.

[6] B. Schultheiß, J. Meyer, R. Mangold, and M. Reichert, “Prozessentwurf für den Ablauf einer ambulanten Chemotherapie.” p. 29, 1996.

[7] B. Schultheiß, J. Meyer, R. Mangold, and M. Reichert, “Prozessentwurf am Beispiel eines Ablaufs aus dem OP-Bereich: Ergebnisse einer Analyse an der Universitätsfrauenklinik Ulm,” p. 116, 1996.

[8] B. Schultheiß, J. Meyer, and R. Mangold, “Prozessentwurf für den Ablauf einer stationären Chemotherapie.” 1996.

[9] I. Konyen, B. Schultheiß, and M. Reichert, “Prozessentwurf für den Ablauf einer radiologischen Untersuchung.” 1996.

[10] I. Konyen, B. Schultheiß, and M. Reichert, “Prozessentwurf eines Ablaufs im Labor.” 1996.

[11] I. Benbasat, D. K. Goldstein, and M. Mead, “The Case Research Strategy in Studies of Information Systems,” *MIS Q.*, vol. 11, no. 3, pp. 369–386, 1987.

[12] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*. Springer Berlin Heidelberg, 2013.

[13] J. Meyer, “Anforderungen an zukünftige Workflow-Management-Systeme: Flexibilisierung, Ausnahmebehandlung und Dynamisierung - Erörterung am Beispiel medinzinisch-organisatorischer Abläufe,” 1996.

[14] W. M. van der Aalst, “A decade of business process management conferences: personal reflections on a developing discipline,” in *Business Process Management*, Springer Berlin Heidelberg, 2012, pp. 1–16.

[15] E. P. Kelly, “The One Best Way: Frederick Winslow Taylor and the Enigma of Efficiency.,” *Academy of Management Perspectives*, vol. 11, no. 3. pp. 101–104, 1997.

[16] B. Westfechtel, *Models and tools for managing development processes*, 1646th ed. Springer Verlag Berlin - Heidelberg, 1999.

[17] M. Weske, *Business Process Management: Concepts, Languages, Architectures*, 2nd ed. Springer Verlag Berlin - Heidelberg, 2012.

[18] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of business process management*. Springer Verlag Berlin - Heidelberg, 2013.

[19] M. Reichert, “Dynamische Ablaufänderungen in Workflow-Management-Systemen,” Ulm University, Institute of Database and Information Systems, 2000.

[20] P. J. Kammer, G. A. Bolcer, R. N. Taylor, A. S. Hitomi, and M. Bergman, “Techniques for Supporting Dynamic and Adaptive Workflow,” *Computer Supported Cooperative Work (CSCW)*. pp. 269–292, 2000.

[21] M. Reichert and D. Peter, “ADEPT flex – Supporting Dynamic Changes of Workflows Without Loosing Control,” *J. Intell. Inf. Syst.*, no. 2, pp. 93–129, 1998.

[22] N. Russell, W. Van Der Aalst, and A. Ter Hofstede, “Exception Handling Patterns in Process- Aware Information Systems,” *BPM Cent. Rep. BPM-06-04*, pp. 288–302, 2006.

[23] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[24] A. Fortis, A. Cicortas, and V. Iordan, “Considerations on Resource Usage in Exceptions and Failures in Workflows,” *Ann. Univ. Tibiscus, Comp. Sci.*, 2009.

[25] “BPMN 2.0 - Business Process Model and Notation,” *04.08.2015*, 2011. [Online]. Available: http://bpmb.de.

[26] F. Daniel, K. Barakaoui, and S. Dustdar, *Business Process Management Workshops: BPM 2011 International Workshops*. Clermont- Ferrand, France: Springer Heidelberg Dordrecht London New York, 2011.

[27] W. M. P. Van der Aalst, a. H. M. Ter Hofstede, B. Kiepuszewski, and a. P. Barros, “Workflow patterns,” *Distrib. Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[28] B. S. Lerner, S. Christov, L. J. Osterweil, R. Bendraou, U. Kannengiesser, and A. Wise, “Exception Handling Patterns for Process Modeling,” *IEEE Trans. Softw. Eng.*, vol. 36, no. 2, pp. 162–183, 2010.

[29] W. M. P. Van Der van der Aalst and M. Pesic, “Towards a Reference Model for Work Distribution in Workflow Management Systems,” *First Int. Work. Bus. Process Ref. Model.*, pp. 76–82, 2005.

Acknowledgment

First of all, my sincere thanks go to Professor Dr. Manfred Reichert for his continuous support, guidance as supervisor and for being motivating and optimistic mentor, especially for the helpful advices and for the interesting discussions.

Moreover, I take this opportunity to express the profound gratitude from my deep heart to my family for their love and continuous support.
Güven Manay and Dr. Rahime Gök-Manay are acknowledged for proof-reading of this thesis.
Special thanks my boyfriend Mehmet Dinc for his personal and mental support.

At the end of my thesis I would like to thank all those people who made this thesis possible and an unforgettable experience for me.

Name: Serife Öztemür Matrikelnummer: 748114

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den . . . . . .

Serife Öztemür