Automated Generation of Business Process Models from Natural Language Input

Master Thesis

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE (M.Sc.)

IN INFORMATION SYSTEMS

AT THE SCHOOL OF BUSINESS AND ECONOMICS OF THE
HUMBOLDT-UNIVERSITÄT ZU BERLIN

submitted by
Fabian Friedrich
Matriculation number 509709

Supervisor: Prof. Dr. Jan Mendling
Associate Supervisor: Dr. Frank Puhlmann
Berlin, 29th November 2010

Acknowledgement

This part of my thesis is dedicated to express my appreciation and sincere gratitude to all the people who influenced and supported me in its creation.

First of all I want to thank my supervisor Prof. Jan Mendling for awaking my interest in BPM, for his great and supportive teaching style, and, of course, for supervising this thesis. In the last months he has not only supported my with scientific advice, help on structuring the thesis and valuable comments and ideas, but also with encouragement and friendship. Thank you! I also want to thank him to establish the contact to Frank Puhlmann and the inubit AG in one of his seminars in the first year of my master studies. I also want to express my gratitude to Frank who had the initial idea for the topic of this thesis, for his feedback and valuable suggestions, and for providing me with a challenging and inspiring work environment in the last two years.

Moreover, I want to thank my fellow student for their support and friendship throughout all of my master studies. I particularly want to thank Christopher, Henrik, Thomas, Johannes, Matthias, Roland, Robert, Daniel. I greatly appreciated your company. Without the several study groups, collaborations on seminar papers and collective exam preparations the time of study would have been rather dull and incomparably harder.

My special thanks go to Christopher, Christian, Henrik, Anne, Robert and Felix for proof reading my thesis and their valuable corrections and recommendations.

For helping me to collect all of the test data I want to thank Frank Puhlmann, Oliver Holschke, Jan Recker, Thomas Kohlborn, Dirk Breitkreuz, Hajo Reijers, Benjamin Fabian, Felix Elliger, Matthias Schrepfer, Nicolas Peters, and Christian Zimmer. They made a large part of this work possible.

Lastly, I want to thank my family for their support and understanding during times of high work load. Anne and Josephin thank you for your love and warmth, encouragement, patience, and for believing in me. I am looking forward to a bright and happy future together with you.

Abstract

A first step to enable effective business process management (BPM) is the design of appropriate conceptual models. These models are used to describe the roles and responsibilities of the employees in an organizational chart, the structure of data, e.g. in an UML class-diagram, and the flow within a process, e.g. as a BPMN-model. They provide the foundation for BPM-initiatives to increase operational performance, but apart from their importance it is also a time and resource intensive task to create such models. On the other hand, information on all the aforementioned subjects is usually available in a company within unstructured textual documents. To reduce the modeling efforts and to accelerate the realization of benefits from a BPM-initiative, we propose an approach for the automated generation of business process models from text documents. In order to create this approach, we analyzed several texts and the corresponding manually created models and derived transformation heuristics from the identified syntactic and semantic patterns. Furthermore, the approach was implemented in a research prototype and evaluated using a similarity metric based on the graph edit distance. Our evaluation has shown encouraging results. In average we were able to generate models which are 76% similar to those created manually by a human.

"Within a computer natural language is unnatural."

Alan Perlis

Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction

This section provides an introduction to this master thesis. After a discussion of the motivational aspects in section 1.1, the contribution to the body of knowledge of Information Systems research is highlighted in section 1.2. Section 1.3 will enlighten the contribution in the perspective of design science. Then, the introduction concludes by providing an outlook on the structure of this thesis in section 1.4.

1.1. Motivation

Business process management is a discipline which seeks to increase the adaptability and value creation potential of companies by holistically analyzing and optimizing their processes regardless of departmental boundaries. But, in order to be able to analyze a process and to tap the full efficiency potential a thorough understanding of the process is required first. The necessary level of understanding can be achieved by creating formal models of business processes. Such business process models act as a means to document and analyze the underlying process and can serve as a blue print for subsequent implementation and automation activities. Several formal or semi-formal graphical notations are available to map a process to its graphical representation containing the performed tasks, involved actors, required data and documents, and business rules, which describes the logical and temporal aspects of the process execution [102].

The knowledge needed to construct process models has to be extracted from the participating actors [49]. However, these actors are usually not qualified to create formal models [35]. For this reason, modeling experts are required to iteratively formalize and validate the models in collaboration with the domain experts. But, this traditional procedure of extracting process models through interviews, meetings, or workshops [110] tends to be cost- and time-intensive due to the informal setting and ambiguity or misunderstandings between the involved participants [105]. Therefore, the initial elicitation of conceptual models presents a knowledge acquisition bottleneck [46]. According to Herbst [49] the acquisition of the as-is model in a workflow project requires 60% of the total time spent.

Thus, substantial savings are possible by providing automation support to a business analyst.

Having said that, we also have to consider that information about the internal processes of an organization is often already present in the form of informal textual specifications. Such textual documents can be policies (e.g., for travel or expenses), reports, forms, manuals, knowledge management systems, and e-mail messages. Content management professionals estimated that 85% of the information in companies is stored in such an unstructured way, mostly as text documents [6]. Moreover, the amount of unstructured textual information is growing at a much faster rate than traditional structured data [135]. It seems reasonable to assume that these texts are potential sources of information for the construction of conceptual models.

Thus, in order to enable domain experts to create formal models simply through a textual description and to leverage the information potential of already existing text documents an automatic transformation technique is needed. The discipline concerned with this type of analysis is Natural Language Processing (NLP). It enables the automated analysis of the syntax and semantics of natural language texts. By applying state-of-the-art NLP techniques to business process descriptions we are able to automatically create formal process models. This will make it possible to list the mentioned time- and resource-savings potentials and to enable a quicker realization of BPM-projects and their benefits. Therefore, the goal of this thesis is to investigate possibilities for such an automated transformation technique, to describe a prototypical implementation, and to evaluate it.

1.2. Research Contribution

As stated in the previous section, the goal of this thesis is the development of a system which is able to generate conceptual models needed for a BPM-initiative from text. Using our approach a system analyst is relieved from the time-intensive modeling task. By pursuing this research goal, we evaluated related works in the field of BPM and NLP, developed a novel transformation approach, and created and evaluated it. Therefore, this

thesis provide the following contributions:

Literature Review we collected several works dealing with the problem of automated process model generation and evaluated their strengths and weaknesses.

Categorization of Issues Based on these findings we developed a theoretical framework which categorizes important issues which are relevant for a transformation approach.

Novel Transformation Approach We transfered our theoretical framework into practice and developed a novel transformation approach. It is the first which does not rely on a pure syntax parse of sentences, but uses grammatical relations — Stanford Dependencies. Furthermore, it is the first which effectively combines four well known NLP tools, namely the Stanford Syntax Parser, FrameNet, WordNet and an anaphora resolution algorithm. Moreover it covers a wider spectrum of modeling constructs then other approaches and includes novel solutions for problems like activities which are spread over several sentences.

Prototype This approach was implemented and tested in our research prototype.

Comprehensive Test Data Set In order to evaluate our approach a comprehensive test data set containing 47 process descriptions and manually created models covering various domains was collected. Additionally, all of the texts and models are included in this thesis, which enables other researchers to verify our approach and to build upon this test data set. Thereby, we also addressed a shortcoming of other research conducted in this area which hardly ever incorporated the underlying test data.

Evaluation Approach To assess the accuracy of our transformation approach we created an evaluation methodology using graph edit distance and reported all results.

1.3. Research Methodology

To develop the transformation procedure, we followed a four step approach.

Test Data Collection The first step was to collect initial test data sets, consisting of natural language process specifications and a corresponding process model. They provided us with insights on how humans are translating a textual description to a process model. Occasionally, the process model was not provided as a BPMN diagram in which case we applied a transformation procedure as described in section 4.2.2.

Analysis of Syntactic and Semantic Patterns By analyzing the linguistic patterns we were able to map text constructions to their corresponding modeling fragments. This also revealed issues regarding the quality of the syntax parse and the text itself. We systematically categorized these issues and developed a conceptual solution strategy.

Transformation Rules Derivation To effectively mitigate the detected issues we derived appropriate transformation heuristics and refined them iteratively. The rules were then implemented in our research prototype to assess the output.

Evaluation We defined, e.g., a similarity metric to compare the models generated by our transformation procedure to those manually created by humans. We applied these metrics to evaluate the performance of our transformation procedure. The gathered information could then be used in another iteration to refine or discover new patterns and rules and to improve the performance regarding the test data set.

The structure of this methodology is illustrated in figure 1. By following this methodology, the contributions of the research conducted in this thesis follows the paradigm of *design science*, as defined by Hevner [50]. In his paper, Hevner states that Information Systems Research is characterized by behavioral sciences and design science. While behavioral science "seeks to develop and verify theories that explain or predict human or organizational behavior", design science tries to "extend the boundaries of human and organizational capabilities by creating new and innovative artifacts" [50]. The transfor-

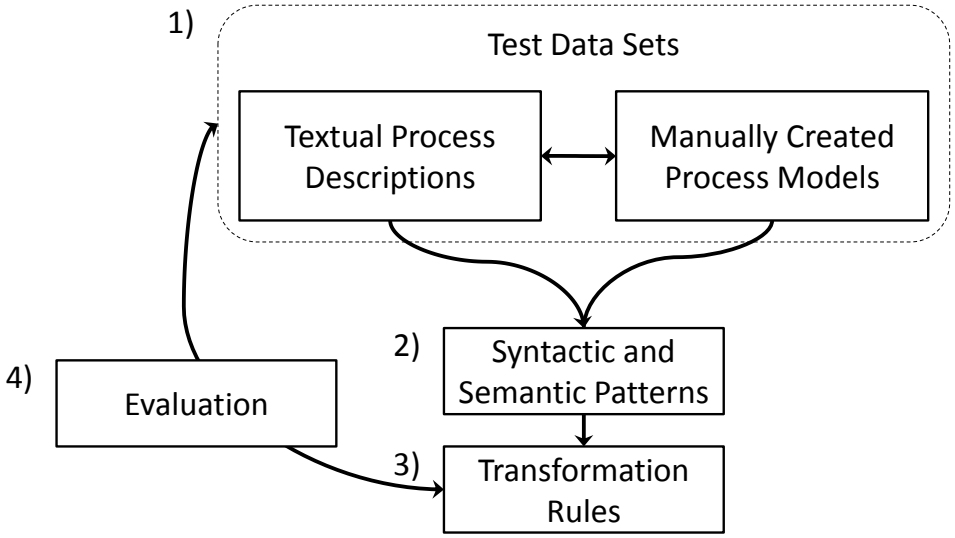

Figure 1: Research methodology underlying this thesis.

mation approach which is developed and presented in the context of this thesis can be classified as such an artifact as it provides a model, the transformation method and an instantiation in form of a research prototype. In order to achieve an easier understanding of what can be considered effective design science research, Hevner proposed seven research guidelines which should be addressed to attain complete research.

Guideline 1: Design as an Artifact "Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation."[50] As the aim of our research is to analyze and describe an approach to transform natural language text into process models and to build and evaluate a corresponding prototype, this guideline can be deemed to be fulfilled.

Guideline 2: Problem Relevance "The objective of design-science research is to develop technology-based solutions to important and relevant business problems."[50] As outlined in section 1.1 a full automation of the as-is design process could reduce the resource requirements of a workflow project by up to 60%. Therefore, the business relevance of our research can be derived from the financial and temporal

implications of a successful implementation.

Guideline 3: Design Evaluation "The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods." [50] As mentioned before, an evaluation, which also captures quality and utility aspects, will be conducted in section 4. We will employ different analytical methods to compare the models generated by our approach with those manually created by a human modeler.

Guideline 4: Research Contributions "Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies." The contributions of this thesis were outlined in section 1.2. Furthermore, all natural language texts and the resulting models which were used for deriving our transformation approach and to conduct the final evaluation are fully listed in the Appendices, so a verification by external parties is easily possible.

Guideline 5: Research Rigor "Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact." [50]. Hevner argues that rigor in a design-science context is often achieved by mathematical formalism and by making effective use of the existing knowledge base. Our approach builds on existing research by making use of, e.g., the *Stanford Parser*, *WordNet* and *FrameNet* [3] in the area of Natural Language Processing (NLP). Furthermore, we are relying on BPM standards like *BPMN 2.0*, foundations of graph theory, and are taking existing publications on the topic of model generation from text into account.

Guideline 6: Design as a Search Process "The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment." [50] Due to the nature of natural language, it is not possible

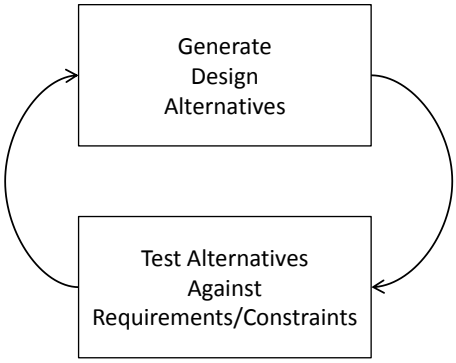

Figure 2: The Generate/Test Cycle [50].

to analyze the whole problem space and find an optimal solution. Because of this characteristic, the problem of parsing and interpreting natural language text is a "wicked" problem, according to the terminology of Hevner. Therefore, the design of our transformation system can be regarded as a search process, where heuristics are designed, tested for their suitability and then further refined. This fits into the generate/test cycle (see figure 2) depicted in [50], which was described in [116].

Guideline 7: Communication of Research "Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences." [50] As a copy of this Master Thesis will remain in the library of the Humboldt-Universität zu Berlin it is publicly available for all types of audiences. Furthermore publications of the most important findings to a wider audience on workshops and/or conferences is planned.

By relating the elements of this thesis to the aforementioned guidelines for information systems research, we demonstrated that the conducted research extends the body of knowledge and conforms to accepted research standards in the discipline of information systems.

1.4. Structure of this Thesis

Section 2 provides background information on the usage, techniques, and goals BPM and Natural Language Processing (NLP) and will introduce their basic concepts and the components we build upon. For BPM we focus on the aspects of process modeling and how to create a valid and formally correct model. In the area of NLP we will explain the workings of the Stanford Parser which enables the analysis of the syntactic structures of a sentence, the resolution of anaphoric references and the analysis of the semantics of a word or phrase with the help of WordNet and FrameNet. Afterward, related articles which describe approaches similar to the one presented in this thesis are explained and differences are highlighted.

In section 3 the details of the transformation procedure are developed. Firstly, the structure and identified issues of documents describing process models are explained and the requirements imposed on a system capable to transform process descriptions into process models are highlighted in section 3.1.

Based on these issues, we propose a two step analysis. We star with the analysis of each sentence using its syntax parse in section 3.3 and continue by analyzing the information contained in the whole text in section 3.4. Then, we show how process models can be generated from the collected information in section 3.5 and how the user can be enabled to quickly adapt important parts of the model to his preferences in section 3.6.

After explaining our transformation approach, we apply it to several examples in section 4. First, our test data set containing 47 process descriptions and their corresponding process models is introduced in section 4.1. Afterward, we define a similarity metric based on the graph edit distance in section 4.2 and show the results of the comparison of models generated by our approach to those created by a human modeler in order to evaluate the accuracy of our approach.

The thesis will conclude in section 5 by outlining perspectives for further research and by highlighting the limitations of our work.

2. Background

This thesis covers an interdisciplinary topic. It relies on concepts and research in the areas of Conceptual Modeling and BPM in particular, as well as NLP. First we will give a rough overview about the topics covered by BPM and will present the "Business Process Model and Notation" (BPMN) standard. Furthermore, the fundamental concepts of quality of process models are introduced and we will highlight how these can be transfered to our approach in order to create correct models.

Afterward, Natural Language Processing is introduced and three specific problems out of its domain are illustrated. These three problems are syntax parsing of sentences, anaphora resolution techniques, and the semantic analysis. During these descriptions, we will demonstrate the capabilities and sketch the internal workings of the four tools we employ for our approach, the Stanford Parser, WordNet, FrameNet, and an Anaphora Resolution technique which is an adapted version of Hobb's algorithm.

2.1. Business Process Management

Business Process Management (BPM) is a management discipline and was defined as:

> "Supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information." [124]

BPM is concerned with the documentation, automation, optimization, and integration of processes.[4]. To achieve these goals companies follow the BPM life-cycle (see figure 3). The first phase of the life-cycle is concerned with the analysis and definition of the current processes within the organization. The next step is the creation of a formal process model which represents these processes [119]. The formal models serve as a basis for the implementation phase. It includes the provision of the appropriate hardware infrastructure, the configuration and implementation of the associated systems and also the training of the involved employees. Finally, the process can be enacted and the process

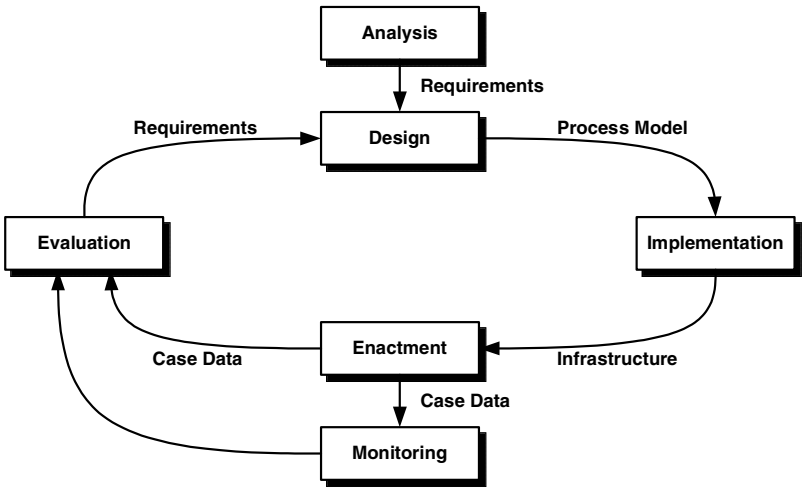

Figure 3: The Business Process Management Life-cycle [81].

is executed with full information system support. While the process is executed, data about the handled cases can be collected. This case data in turn can be used to monitor the performance of the executed process, to trigger alerts when certain key performance indicators are not met and to decide on the necessary counteractions. In the evaluation phase, the collected case data can be analyzed to derive new requirements, which enable further process improvements in the next iteration of the BPM life-cycle.

Companies have to put considerable effort into the design, implementation, execution, monitoring, and evaluation of processes and the corresponding data. Thus *Business Process Management Systems* (BPMS) are often employed to provide the necessary software support for those activities. In his book, Weske defined BPMS as "a generic software system that is driven by explicit process representations to coordinate the enactment of business processes". Such explicit representations are conceptual models, which are the result of the design phase as depicted in the BPM life-cycle. In the area of BPM different kinds of conceptual models are required, including representations of functions, data, organization, system landscapes, and most importantly for our perspective process models [110]. Such process models provide an abstracted representation of several business

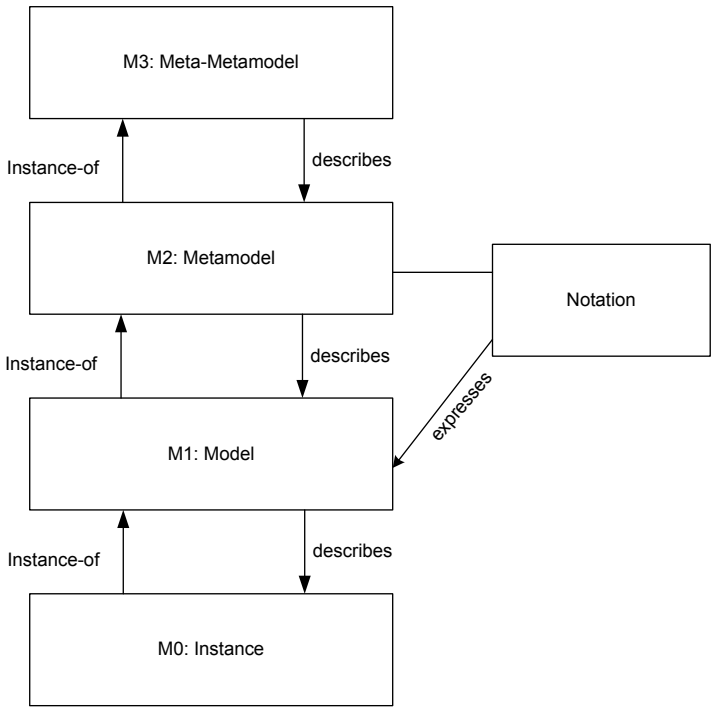

Figure 4: Levels of Abstraction [134].

process instances which are the interactions within a company conducted to create value. It includes activities conducted by humans and/or software systems and is able to show interdependencies. To visualize process models, different notations can be used, like EPC [20, page 258ff], YAWL [125], BPMN [87], UML Activity Diagrams [88], or Petri Nets [92]. These languages define the metamodel — the rules and structure the model has to follow (see figure 4).

The process of crafting and refining such conceptual models in general was characterized in by Frederiks [35]. He distinguishes four phases: elicitation, modeling, verification, and validation. The results of the elicitation phase is an informal specification. In our case this can be a process description in natural language. Next, this informal specification is transformed into a formal representation during the modeling phase. This formal representation is then verified and checked for conformance to its metamodel. As it is

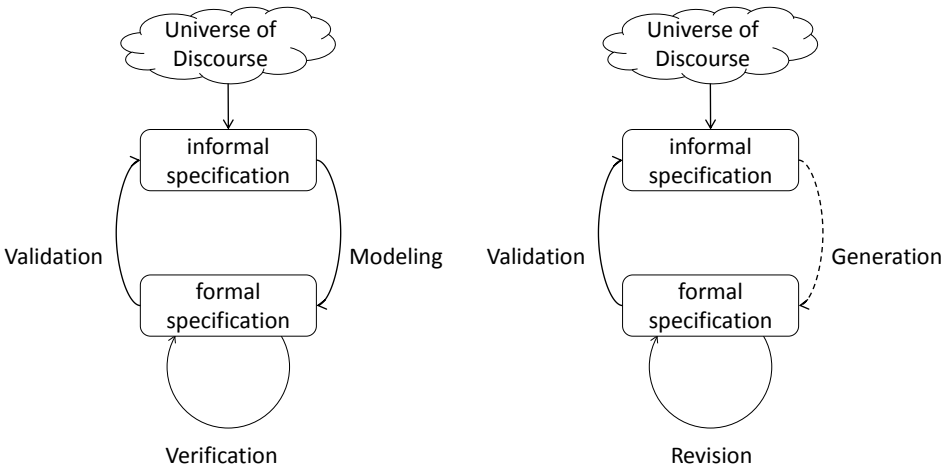

Figure 5: The process of information modeling adapted from [35] (left) and revised including generation capabilities (right).

our goal to automate the modeling process, we can directly generate a preliminary formal specification from the informal specification. During this generation procedure we can ensure that only constructs which are valid regarding the metamodel are created. Therefore, instead of verifying the generated model, it has to be revised and corrected to account for possible errors in the transformation procedure. The original process taken from Frederiks [35] and an adapted version which reflects improvements of introducing an automated transformation approach are shown in figure 5.

2.1.1. Business Process Model and Notation 2.0

For the approach developed in this thesis we decided to use BPMN 2.0 as language or metamodel for the resulting process models. The main reason for this decision is that BPMN is an official standard supported by the Object Management Group (OMG) for process modeling. Furthermore, it provides rich expression capabilities and is widespread. This is also reflected within our test data set (see section 4.1), where the majority of the contained process models were directly provided in BPMN. We used the most

recent version — BPMN 2.0 —, which was last revised in June 2010[1] [87]. With the introduction of BPMN 2.0, each element of a BPMN process diagram has clearly defined semantics. Furthermore, it is based on several former modeling techniques like "UML, IDEF, ebXML, LOVeM, and Event-driven Process Chains" [103] and seeks to combine their strengths.

From a graph theoretic point of view, a BPMN business process model can be regarded as a directed graph with the nodes N and edges E $\subseteq$ N x N connecting those nodes [45]. Thereby, an edge always connects two nodes, starts at a node which we define as source and ends at a node which we define as target. During the descriptions of our transformation procedure in section 3 we will utilize these definitions.

BPMN distinguishes between four types of elements.

- Flow Objects (Activities, Events and Gateways)
- Swimlanes (Pools and Lanes)
- Artifacts (e.g. Data Objects, Text Annotations or Groups)
- Connecting Objects (Sequence Flows, Message Flows and Associations)

While the first three types of elements are nodes, the latter ones are edges. In figure 6 we are displaying a subset of BPMN Flow Objects, Swimlanes and Artifacts. All of these elements are considered potential generation results of our transformation approach.

"A Task is an atomic Activity within a Process Flow. A Task is used when the work in the Process cannot be broken down to a finer level of detail." [87, page 160]. Whenever a group of activities is combined or reusage of a process fragment is intended, a Subprocess can be used instead of a Task. A Subprocess can be collapsed and expanded (hence the plus sign at the bottom) to either show or hide its details. Gateways play an important role as they enable the process flow to be split and joined. An exclusive

[1] `http://www.bpmn.org/`; last accessed 2010-11-05

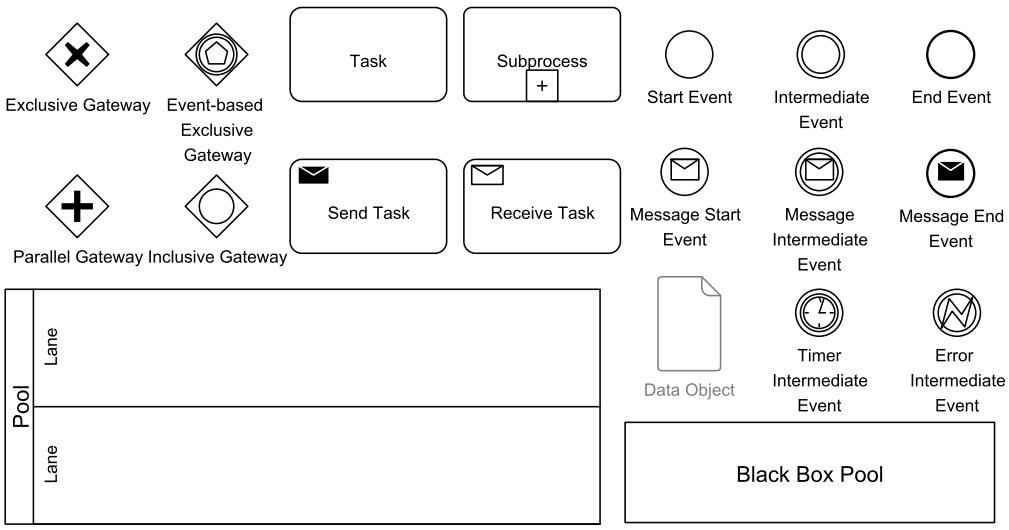

Figure 6: Considered subset of BPMN nodes.

Gateway (or XOR Gateway) is used to model a decision. Out of all its incoming or outgoing edges only one path will be selected. An event-based exclusive Gateway shows the same behavior, but requires that all of its successors are events. The semantics of a parallel Gateway are different as it will activate all of its outgoing paths or requires that all of its incoming paths are activated. Thus it can be used to model concurrent behaviors. The inclusive Gateway (also OR Gateway) can activate 1–n of its in-/outgoing edges or requires them to be activated in order to proceed. It is thus more versatile, but also more complex [80]. Event nodes can be used to denote the start or end of a process. Additionally, intermediate Events can be used within the process flow to make clear that the process will halt and wait for the expected Event to occur. The nature of the Event can be signified by additional symbols, e.g., for Messages, Time or Exceptions as shown in figure 6. BPMN specifies 13 different event types which are fully listed in the BPMN 2.0 specification [87].

The presented transformation approach is also able to generate Data Object artifacts. These can be connected to nodes and edges to demonstrate the kind of data which is used

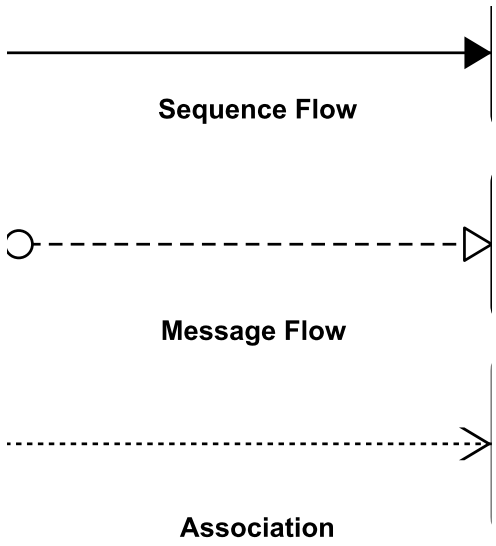

Figure 7: Considered subset of BPMN edges.

within this process step.

Lastly, Pools, and Lanes are an important part of the process. "A Pool is the graphical representation of a Participant in a Collaboration. A Participant (see page 115) can be a specific PartnerEntity (e.g., a company) or can be a more general PartnerRole (e.g., a buyer, seller, or manufacturer)" [87, p 112]. Hence, a Pool can represent a human, organization, or software system involved in a business process. While a Pool is used to show the boundary of an organization and to determine the involved process participants, a Lane can be used to partition the Pool and show the different process participants within that body. Again, this could be different individual, organizational units or software systems. Whenever the behavior of an involved participant is supposed to be left unspecified, a Black Box Pool can be used. This way other process participants and the interactions with them can be shown in a diagram without the need to specify their behavior directly.

BPMN distinguishes between three types of edges or Connecting Objects. Firstly, Sequence Flows which are used to connect Flow Objects within the same Pool. Secondly, Message Flows which have to be used whenever an edge crosses the boundary of a Pool. Therefore, it can be used to visualize the interactions between several process participants

in a Collaboration Diagram. Lastly, Associations are used to connect Artifacts to Flow or Connecting Objects.

According to zur Muehlen [84] only few BPMN diagrams use more than 15 different elements. The subset we defined contains 21 elements. Therefore, we are confident that the required elements for the majority of BPM projects can be provided by our transformation procedure as it covers the important and most widely used elements [84]. When creating process models questions about their understandability and quality arise. Therefore, the next section will provide an insight into research which is concerned with these aspects, which are relevant for the automated generation of process models.

2.1.2. Process Model Labeling and Quality Aspects

The quality of a conceptual model including process models can be characterized by three major aspects according to Lindland [73]. These are semantic, syntactic, and pragmatic quality. These aspects can be roughly described as:

Semantic Quality describes whether the model correctly reflects the modeled domain.

Syntactic quality defines if the model conforms to its modeling language and the metamodel.

Pragmatic Quality is concerned whether the involved actors are able to understand the model correctly.

These are the basic pillars of quality in the area of conceptual modeling. Lindland's model was specialized to take more detailed aspects into account (e.g. [61]) and has been applied to process modeling languages including BPMN ([86, 132]). Next we will take a look at the importance of the individual quality categories for our transformation approach. As the congruence of the model with the domain depends on the underlying text, it is hardly verifiable. Syntactic quality aspects are concerned with the usage of modeling constructs in our process model. As we followed the BPMN 2.0 specification

[87] during the generation phase and as we are creating a new model from scratch the syntactic quality should be assured. The pragmatic quality is remaining as an interesting research object. While the modeling constructs we are able to employ are fixed the composition of the model and the labels used for the elements are not. In order to create an understandable model structural guidelines were proposed [79, 5]. We should try to consider these guidelines, e.g. creating an appropriate joining gateway for each split, or minimizing the in- and out-degrees of nodes. However, as we are working with textual process descriptions appropriate labeling of nodes and edges will be a more important factor for the pragmatic quality of the generated models. Recently, there has been an increased academic interest in quality of labels within process modeling [37, 78, 66, 65]. According to a recent article [78] three main labeling styles for process models can be identified:

Verb-Object style consisting of a verb in the imperative and the according business object (e.g. approve claim)

Action-Noun style utilizing of a nominalized verb or its gerund instead of its infinitive (e.g. creation of a quotation, Notification printing)

Rest style which subsumes all other labels apart from those in the former two styles

An empirical evaluation led to the results that label formed in the verb-object style are easier to understand and, hence, this style should be preferred. Other works [66, 65] have shown that by using the, e.g. the *Stanford Parser* it is possible to automatically determine the labeling style and to disambiguate verb and business object. Therefore, we will generate verb-object style labels in our process model generation procedure. Furthermore, as we create the labels on a basis of a whole sentence, the detection of the verb and object is possible with high accuracy. Thus, we are able to enhance our models with meta information, which is useful in a later quality analysis.

While this section provided insights into the research revolving our business process modeling, the next section will deal with Natural Language Processing and will describe the tools and concepts which we will employ in order to process textual input.

2.2. Natural Language Processing

Methods in the area of Computational Linguistics and Natural Language Processing, which are a branch of artificial intelligence, try to analyze and extract useful information from natural language texts or speech. Therefore, it is concerned, e.g., with the recognition and synthesis of speech in natural languages such as English [55]. An exemplary area of application is sentiment analysis, where the goal is to automatically determine the attitude or opinion towards e.g. a product or company from online articles [90]. For the creation of our transformation approach, three concepts are of vital importance and will be described in more detail in the upcoming subsections.

- Syntax Parsing - the determination of a syntax tree and the grammatical relations between the parts of the sentence.
- Anaphora Resolution - the identification of the concepts which are references using pronouns ("we","he","it") and certain articles ("this", "that").
- Semantic Analysis - extraction of the meaning of words or phrases using the lexical databases WordNet and FrameNet.

2.2.1. Syntax Parsing

In the area of text processing, a goal is the automated determination of parts of speech (POS) and the recognition of syntactic structure, like which words form a phrase, and grammatical relations between words within a sentence. Examples for such syntactical parsers are the *UC Berkley parser*[2] and the *Stanford parser*[3]. Other parser or POS-taggers

[2] `http://nlp.cs.berkeley.edu/Main.html#Parsing`; last accessed 2010-10-27

[3] `http://nlp.stanford.edu/software/lex-parser.shtml`; last accessed 2010-11-25

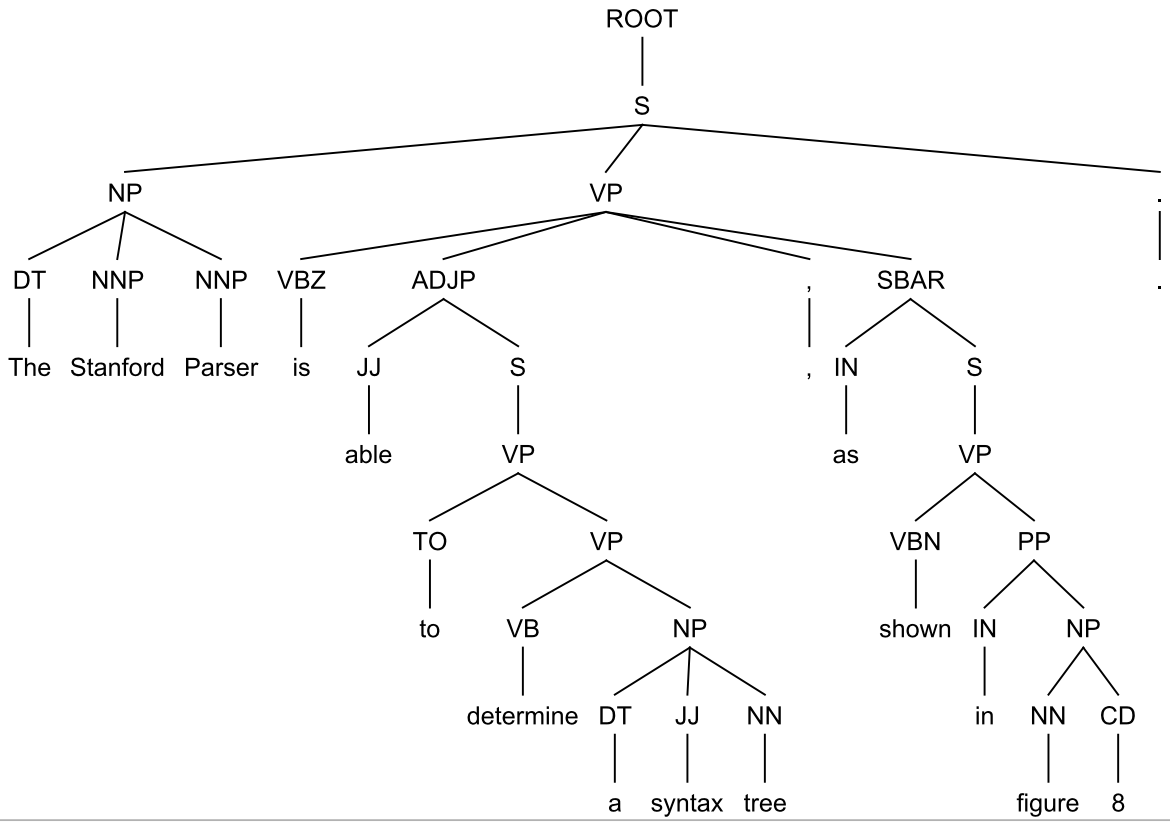

Figure 8: A syntax tree generated by the Stanford Dependency Parser.

are, for example, the *Brill tagger* [9], the parser developed by Eugen Charniak [14], or the freely available NLP toolkits *NLTK*[4], *OpenNLP*[5], *GATE*[6], or the *RASP* system [11], which apart from a POS-tagger also contain other instruments like tokenization and transformation utilities.

The Stanford Parser is able to determine a syntax tree, as shown in figure 8. This tree shows the dependencies [77] between the words of the sentence through the tree structure. Additionally, each word and phrase is labeled with an appropriate POS/phrase-tag. The tags the *Stanford Parser* uses are the same which can be found in the *Penn Tree Bank* [76].

[4] `http://www.nltk.org/`; last accessed 2010-10-05

[5] `http://opennlp.sourceforge.net/`; last accessed 2010-10-05

[6] `http://gate.ac.uk/`; last accessed 2010-10-05

det(Parser-3, The-1)	xcomp(able-5, determine-7)	advcl(determine-7, shown-12)
nn(Parser-3, Stanford-2)	det(tree-10, a-8)	prep_in(shown-12, figure-14)
nsubj(able-5, Parser-3)	amod(tree-10, syntax-9)	num(figure-14, 8-15)
cop(able-5, is-4)	dobj(determine-7, tree-10)	
aux(determine-7, to-6)	mark(shown-12, as-11)	

Table 1: Stanford dependencies as generated by the Stanford Parser.

But apart from that, the *Stanford Parser* also produces *Stanford Dependencies* [25, 23], which is the reason why it was used for the approach explained in section 3. These dependencies reflect the grammatical relationships between the words. As an example, the first sentence of this paragraph contains the dependencies shown in table 1.

The *det* relation shows that the first word in the sentence "The" is a determiner for the third word "Parser". the *nsubj* relation in turn shows that "Parser" is the subject of the sentence and that "(be) able" is its predicate. The first element of this grammatical relation is called the governor (gov) while the second element is called the dependent (dep). These grammatical relations provide an abstraction layer to the pure syntax tree. They also contain information about the syntactic role of all elements. Thus, the task of deriving appropriate transformation rules was simplified by using the Stanford Parser and our approach becomes more robust to changes in the word or phrase order. A list with detailed explanation of all 55 dependency relations can be found in the Stanford parser manual [24]. Nevertheless, we provide an overview about the 12 most frequently referred relations:

agent Describes the complement of a passive verb. It is usually found in a prepositional phrase introduced by the word "by".

ccomp "A clausal complement of a verb or adjective is a dependent clause with an internal subject which functions like an object of the verb, or adjective" [24] (Example:

"It tells the other department that they need to restart their work").

conj Describes a conjunction relation between two elements such as "and" or "or".

cop A copula shows the relation between the subject and the complement of a copula verb ("is", "seem", "appear" etc.).

dobj A direct object or "accusative" object directly follows a verb phrase.

mark The "mark" relation points to the word introducing an adverbial clausal complement, like "if", "because", or "after".

neg The negation modifier indicates that a word is negated, e.g., using "not".

nn The noun compound modifier shows that several nouns are part of a compound (e.g. "Stanford Parser").

nsubj The nominal subject is the subject of an active phrase.

nsubjpass The passive nominal subject is the syntactic subject in a passive clause.

prep "A prepositional modifier of a verb, adjective, or noun is any prepositional phrase that serves to modify the meaning of the verb, adjective, or noun" [24].

rcmod A relative clause modifier indicates the presence of a relative clause modifying a word.

xcomp "An open clausal complement (xcomp) of a VP or an ADJP is a clausal complement without its own subject, whose reference is determined by an external subject. These complements are always non-finite. The name xcomp is borrowed from Lexical-Functional Grammar" [24] (e.g. "Then we start **calling** the customer.") .

The parser uses a probabilistic context free grammar (PCFG) [57] to determine the best parse, which can be achieved, for example using conditional random fields (CRF) [63],

maximum entropy [101] or hidden markov models (HMM) [99]. As was shown in [58] the accuracy of the PCFG is further increased by applying a factored model which includes the probabilities of the respective dependency parse [18]. The Stanford parser achieved a F_1- measure, an average of the two standard accuracy measures in information retrieval - precision and recall [129, 2], of 86.7% when used on Wall Street Journal articles, which were also used as training data [58]. As mentioned, this is below other pure lexicalized PCFG parsers, like the one presented in [15], but mainly due to "many finely wrought enhancements", which can be extracted and incorporated into the *Stanford Parser*.

2.2.2. Anaphora Resolution

Another problem which has to be tackled to produce conceptual models from text is the resolution of anaphoric references. Anaphoras include possessive pronouns (e.g., "my", "your", "her"), personal pronouns (e.g., "I", "you", "she"), certain determiners ("this", "that"), relative pronouns ("who", "which") or phrases describing the object under investigation with different expressions (e.g., "Steve Jobs", "the CEO of Apple"). In [51] a simple algorithm for the resolution of pronouns is proposed. It was later extended in [38]. Here a personal pronoun is resolved by scanning the text for noun phrases which could be candidates for a resolution. For each candidate then a score is calculated consisting of:

1. the distance between the anaphoric reference and the candidate (closer candidates are preferred),
2. the gender, number, person, and animaticity of a candidate (e.g. "it" cannot refer to a person),
3. the role of the candidate within its own sentence (e.g. subjects are preferred), and
4. the number of previous occurrences of the candidate (a higher number of occurrences is preferred).

This technique is able to correctly resolve 84.2% in the tests which are described in [38]. Jurafsky and Martin [55, chapter 21] also mention further possibilities of restricting the

selection process through the usage of parallelisms, verb semantics, and selectional restrictions. Examples for implementations of anaphoric reference resolution algorithms are the *GuiTAR* Framework[7] [94], *BART*[8] [130] or the *Reconcile* framework[9] [121]. Although these libraries are written in Java, they require a special XML-Format as input or are not seamlessly usable with the output provided by the Stanford Parser. Thus, a simplified implementation, which is explained and evaluated in section 3.4.1, was used to develop our approach and compared with the performance of two of these libraries.

2.2.3. Semantic Analysis

The third part relevant for our system is the possibility to include semantics analyses. Apart from the syntactical role, each word in a sentence also has a specific meaning - the semantics. Systems which try to capture semantic relations are, e.g., FrameNet[10] [3], developed at the University of Berkley and the lexical database WordNet[11] [83]. Both were used for the development of our prototype.

WordNet is a semantic database which was developed in 1985 [83] at the University of Princeton. Since then, it has steadily grown and today it contains more than 155,000 unique words and more than 200.000 word-sense pairs[12]. These words are organized into so called SynSets (Synonym Sets). A SynSet contains several words which share the same meaning. Furthermore, the SynSets in WordNet are linked to each other through pointers of different types. Therefore, a program is able to extract different semantic relations for a given word. For example:

- Synonyms - Words which have the same meaning (to work on - to process).

[7] `http://cswww.essex.ac.uk/Research/nle/GuiTAR/`; last accessed 2010-10-05
[8] `http://www.bart-coref.org/`; last accessed 2010-10-05
[9] `http://www.cs.utah.edu/nlp/reconcile/`; last accessed 2010-10-05
[10] `http://framenet.icsi.berkeley.edu/`; last accessed 2010-11-26
[11] `http://wordnet.princeton.edu/`; last accessed 2010-11-26
[12] `http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html`; last accessed 2010-10-05

- Homonyms - Words which are written identically, but have a different meaning.
- Hypernyms/Troponyms - Nouns which are superordinate to the given noun. The opposite of a hypernym is a Hyponym. The same principle can also be applied to verbs which is called a troponym then (sue - challenge, tree - plant).
- Meronyms - Structures nouns in a "part-of" relationship (car - wheel).
- Antonyms - Mainly used for adjectives and adverbs and describes the opposite (wet - dry, hot - cold).
- Entailment - An implication which can be drawn logically (divorce - marry).
- Nominalizations - A verb, adjective or adverb which is used as a noun (creation - create).

Thus, WordNet can be used as a general purpose ontology to distinguish, e.g., an unanimated object from an acting person or system. Since version 3.0 WordNet also incorporates a word stemming technique. The purpose of word stemming is to reduce a word to its base form without any suffixes (lexeme). The technique implemented in WordNet makes use of morphological rules which do not obstruct the word as it is e.g. done by Porter's Stemming Algorithm [96]. Therefore, we can also use it to normalize the different representations of a lexemes during our transformation approach.

Other usage which can be relevant for an extension of our approach beyond what is described in this thesis include the calculation of word similarity [113] and semantic word sense disambiguation [122, 137].

The philosophy of FrameNet is different. The FrameNet project is based on the notion of frame semantics [31], which states that we cannot understand a word without its context — the Frame. To create these Frames and connect them with each other 135.000 sentences

[106] were annotated and over 1.030 Frames with 11.656 *Lexical Units* defined[13]. A Lexical Unit represents a Word-Sense pair similar to WordNet, but additionally each Lexical Unit belongs to a semantic frame. A frame describes a particular situation, event, or object and defines common properties and roles, which are called *Frame Elements*. As an example we look at the Lexical Unit "approve", which belongs to the "Grant Permission" Frame[14]. The "Grant Permission" Frame contains two core Frame Elements, Grantee, and Grantor. Addtional non-core Frame Elements are, e.g., Action, Allowed_Category, Manner, Place, Purpose, and Time. If the Frame is instantiated by the use of one of its Lexcial Units, e.g., "permit", "authorize" or "approve", the different elements of a sentence fill the roles of the Frame Elements as shown in these two annotated sentences taken from the report of the Lexical Unit "approve".

- [This study]$_{\text{Action}}$ was APPROVED [by the ethical committee of the hospital]$_{\text{Grantor}}$.
- Mrs Bradshaw had met my mother a couple of times and [they]$_{\text{Grantor}}$ [plainly]$_{\text{Manner}}$ APPROVED [of each other]$_{\text{Grantee}}$.

It is also possible that core Frame Elements are not expressed within a sentence. This is the case in our first example where no Grantee can be identified. In this case, Frame Elements are marked as "definite null instantiation" (DNI) or as "constructional null instantiation" (CNI) if the omission is "licensed by a grammatical construction" [106, p. 25]. The FrameNet data files do not only provide the defined Frames, their Frame Elements, and Lexical Units, but also the Annotation Corpus. Within this Annotation Corpus all annotated sentences and their syntactical occurrence patterns, which are called *Valence Pattern*, are included. Therefore, it is possible to look up each annotated sentence and

[13] `http://framenet.icsi.berkeley.edu/index.php?option=com_content&task=view&id=17881&Itemid=66;` accessed 2010-11-05

[14] `http://framenet.icsi.berkeley.edu/fnReports/data/lu/lu11734.xml?mode=lexentry`, last accessed 2010-11-05

the Valence Patterns of the different Frame Elements. A Valence Patterns describes the syntactic patterns which are associated to the Frame Elements within a certain annotated sentence. For example, the Grantor of our first example sentence would be marked with the *Valence Unit* "PP[by]", because it is contained in a prepositional phrase headed by the word "by". A Valence Pattern summarizes the usage of those Valence Units in combination, as it could be possible for a Frame Element to appear in a predominant syntactical pattern only in combination with other Frame Elements. This information can be used to automatically identify and assign semantic roles within a given sentence [42, 41] or to build a semantic parser [115].

2.3. Application of NLP for Process Model Creation

Recently, there has been an increased academic interest in defining methods for the derivation of conceptual models from text. One of such approaches is [21], from the federal University of Rio de Janeiro, which has been further extended and verified in [22]. There the authors focus on the derivation of models from group stories and provided a prototype which handles Portuguese texts. Participants of the process to be analyzed are asked to write down their experiences. These texts are then interpreted using NLP techniques and BPMN process models are derived. The approach was further tested with a course enrollment process modeled by students. The examples of this paper show that process models can be created successfully, but only a limited set of BPMN elements is considered. Furthermore, a couple of their exhibits show that syntactical problems occur in some cases — issues we want to tackle with our approach.

A research group at the University of Wollongong also developed a system called *R-BPD* [39]. The toolkit uses a syntax parser to identify verb-object phrases in the given text and it also scans the text for textual patterns, like "If <condition/event>, [then] <action>."[40]. The result are BPMN model snippets rather than a fully connected model. Nevertheless, this toolkit does not only derive BPMN snippets from unstructured text, but also takes existing model, e.g. an UML sequence diagram, into account. As

some of the models used as a source might be a graphical representation of the texts which were also analyzed, a cross validation and check for duplicates is performed.

At the University of Klagenfurt a procedure called KCPM (Klagenfurt Conceptual Predesign Model) [59] and a corresponding tool were developed [60]. It parses textual input in German and fills instances of a generic meta-model, the KCPM. Using the information stored in this meta-model an UML activity diagram [108] and a UML class diagram [34] can be created . The transformation from natural language input to the aforementioned meta-model is not a fully automated process, but rather semi-automated as a user has to be involved in the process. Using the tool, the user has to make decisions about the relevant parts of a sentence or has to correct the automatic interpretations. Therefore, the approach does not leverage the full time- and cost-savings potential.

In contrast to that, the approach described in [139] is fully automated. It uses use-case descriptions in a format called RUCM [138] to generate UML activity [140] and class diagrams. But, the system is not able to parse free-text. The RUCM input has to be in a very restricted format allowing only 26 types of sentence structures and relies on keywords like “VALIDATES THAT” or “MEANWHILE” to determine the semantics inherent in the text. Thus it cannot be used in an initial process definition phase as it would require rewriting of all documents present in a company to comply with the RUCM format.

A fifth approach is the one of Policy-Driven Process Mapping, a joint work of the University of Delaware, the City University of Hong Kong, and IBM Research in Hawthorne [133]. First, a procedure was developed which creates a BPMN diagram, given that data items, tasks, resources (actors), and constraints are identified in an input text document. Although the approach does not require a process description to be sequential, as items are combined e.g. on their required inputs and outputs, it only supports a very limited set of BPMN elements. Pools, Data Objects, and Gateways other than an exclusive split are not considered. Furthermore, a Gateway is not allowed to have more than two outgoing arcs, reducing the space of available modeling constructs considerably [133]. Furthermore, user-interaction is required at several stages throughout the process. In a more

recent study [68] the authors evaluated statistical classifiers to automatically detect (a) which sentences out of a large corpus are essential for the creation of process models and (b) which words correspond to the aforementioned items within a relevant sentence. To determine the quality of the detected sentences and elements, the standard measures *precision*, *recall*, and *F-measure* [129, 2] are computed. Although the approach achieves good results with respect to precision, only a low recall is reached, which is also represented in the F_1-measure which is below 60% for all cases.

Another approach, which is similar to ours was presented in [118, 117, 62]. The authors of these papers employ a linguistic analysis engine based on the UIMA Framework. The UIMA Framework enables the constructions of linguistic analysis systems by combining different blocks into a pipeline. Specifically, in [118] the following steps are mentioned: Texts are preprocessed with a part-of-speech tagger in combination with a shallow parser. Afterward, the words are annotated with dictionary concepts, which classify verbs using a domain ontology created by the authors. Then an anaphora resolution algorithms and a context annotator, which determines the likelihood of an identified noun phrase to be an actor in the system, is applied. The information is then transferred to a Use Case Description metamodel and later into a BPMN process model. The focus of this system is the analysis of very structured use-case descriptions. A single use case consisting of few sentences is turned into a small process model. The authors then combine Subprocesses containing these models into a def-use graph [120], which is subsequently optimized. Unfortunately, none of their works contains a full example text and model. Therefore, a verification of their results and a comparison to our approach is not possible. Furthermore, the approach can be regarded as text type specific as only use-case descriptions were used.

Furthermore, the dictionary concepts, which are a vital part of their approach, rely on the underlying domain ontology which has to be hand-crafted. This imposes a manual effort when transferring the system to other types of texts or languages. In contrast to that, our approach builds on the freely available WordNet and FrameNet lexical databases, which possess a large corpus and are available for different languages.

2.4. Other Related Work

We also inspected related work where NLP was used to achieve other BPM relevant goals or areas where the creation of process models was not the focus. This includes:

- Research on the automatic matching of WebServices to user queries written in natural language [7, 19].
- The identification of process model relevant sections in accompanying documentation [54].
- The mining of process models from event logs [126].
- The generation of process models from text without linguistic analysis [75].
- The usage of NLP techniques to provide machine-assistance during the modeling process [72].
- Works on the automated generation of data models, e.g. in UML, from text [69].

In [7, 19] frameworks for the analysis of user queries in natural language based on a domain ontology are used to determine an appropriate service composition. The services which were composed that way can then directly be parameterized with the information provided by the user to answer a query. The systems were used to answer question regarding entertainment, e.g. "Which cinemas play the movie x", or to control devices which support the UPnP protocol.

Ingvaldsen [54] described a methodology which can be used to automatically determine links from an existing process model to the corresponding passages within a textual description. Because both, the model and the text, are already assumed to be present, it becomes a matching problem, which is solved by applying a vector model to the text and the labels, respectively. As our approach uses the textual information to create the process model, such links can automatically be included. This enables the user to quickly gather additional information or verify the generated model.

Process Mining [126] is another approach for the automatic generation of process models. But instead of text, it uses event logs, e.g. from an ERP system, and then applies algorithms to construct a process model which is able to explain the logical and temporal relations between the events found in the log.

In [74] a different approach to the generation of business processes in the EPC notation from use case templates is presented. A strict tabular form of use-cases where the action, trigger, pre- and postconditions are clearly defined is required. By applying string matching to the pre- and postconditions, a process model is created. The approach was tested with a single use-case describing a student enrollment process. Additionally, the approach was transfered to BPMN in [75], but the requirement of a fully structured use-case template remain.

In [72] the Stanford Parser was used to analyze existing process models. The results of this analysis is a Descriptor Space which contains information about entity life-cycles and activity hierarchies. This information can then be utilized to assist a modeler when creating a new model. Succeeding activities can be proposed based on the life-cycle of the used business objects in the analyzed model repository.

The idea to automatically generate create conceptual models from natural language input was pursued ten years ago already. Examples are the CM-Builder system [48], GOOAL [91], or LIDA [89], which create UML class diagrams out of a software requirement text. They analyze textual input using a POS tagger and create classes out of the identified noun phrases. Additionally, relations between the classes are created out of the S-V-O structure of a sentence. The mentioned approaches were compared and the limited set of supported UML modeling constructs was mentioned as the major limitation in a study conducted at the Heriot-Watt University, Edinburgh [69]. Although not exactly related to our research goal, we considered these six lines of research during the construction of our transformation approach as it provided valuable insights into the issues and considerations which are relevant for the automatic transformation of natural language input.

3. Transformation Approach

The previous sections provided an overview of BPM and NLP techniques. This section focuses on the proposed transformation approach. Therefore, we first investigate the issues of natural language process descriptions in more detail. Based on these issues we can formulate our basic assumptions from which we then derive a structural framework comprising three phases: a sentence level analysis, a text level analysis and the process model generation phase. Additionally, a lane split-off procedure was developed, which enables the user to quickly adapt the model to his preferences. We will delve into each of these phases and their workings will be described in detail using procedural descriptions and pseudo code. A user is able to configure the algorithm using several options, which will be explained in the text. The presented algorithms were implemented in a research prototype which is briefly explained in Appendix D.

3.1. Categorization of Issues

The most important issue we are facing when trying to build a system for the understanding of natural language is its complexity. We collected issues related to the structure of natural language texts from the scientific literature discussed in section 2.3 and analyzed the test data that was collected and is thoroughly described in section 4. Thereby, we were able to identify four broad categories of issues which we have to solve in order to analyze natural language process descriptions successfully.

1. Semantics $\neq$ Syntax - the mismatch between the semantic and syntactic layer of a text
2. Atomicity - the question of how to construct a proper phrase-activity mapping
3. Relevance - parts of the text or just of a sentence might be irrelevant for the generated process model
4. Referencing - how to resolve relative references not just between words, but also between sentences and their content

For each of those categories we will show three specific issues and provide examples out of our test data set to illustrate the problems that arose.

Additionally, table 2 summarizes our findings of the conducted literature review. Interestingly, no comprehensive study on the problems relevant to the structural analysis was found, although this provide the starting point for the construction of a transformation engine.

Issue	References
1 Syntax $\neq$ Semantics	[1]
1.1 Active-Passive	[1]
1.2 Rewording/Order	[140, 59, 34]
1.3 Implicit Conditions	[40, 39]
2 Atomicity	
2.1 Complex Sentences	[68, 34]
2.2 Action Split over Several Sentences	[118]
2.3 Relative Clauses	[68]
3 Relevance	[118]
3.1 Relative Clause Importance	[68]
3.2 Example Sentences	[59]
3.3 Meta-Sentences	[68]
4 Referencing	
4.1 Anaphora	[1, 118, 22]
4.2 Textual Links	[33]
4.3 End-of-block Recognition	[59, 68]

Table 2: References in the literature to the analyzed issues.

Different solution strategies were applied in all of the mentioned works to overcome the stated problems, e.g. by constricting the format of the textual input [140], but no study considers all mentioned problems and offers a comprehensive solution strategy. Another interesting fact is that none of the works using a shallow parser shows how they deal with passive constructions [1, 118, 22, 140]. While we were able to solve this problem by the usage of the Stanford Parser and the grammatical relations it creates, those works should be facing problems when passive sentences are considered. The rest of this chapter is dedicated to analyzing and discussing these issues. We will then seize the developed suggestions and reference the issues as stated in table 2 during the description of our transformation approach in the following sections.

3.1.1. Semantics ≠ Syntax

To express the same semantic concept in a language different syntactic patterns can be used [1] and the syntactic structure of a clause is not necessarily directly linked to its semantics [31]. Thus, to build a process model, it is necessary to extract the different parts of an action for, e.g., a task node. To accomplish this goal the different semantic roles of the words in a sentence (Actor, Resource, Action) have to be determined.

As an example consider the following two sentences, the first one in active, the second one in passive voice:

- John broke the window with a football.
- The window was broken with a football by John.

In the first sentence “John” is the syntactic subject of the sentence and also the acting person. In the second sentence the syntactic subject is “the window” and “John” is only mentioned in a prepositional phrase, which could also be omitted, although the semantic pattern of both sentences is the same.

Another example is the exchangeability of certain words or phrases without changing the meaning of a sentence, as in this example:

- John should run away if the window is broken.
- In case the window is broken, John should run away.

One of the most difficult problems we are facing is the recognition of the rhetoric structure of the text, as conditions and the timely order are most relevant for process models. Whenever these discourse relations are expressed explicitly with an appropriate marker, we can detect and correctly handle it. Problems arise when the text contain implicit discourse relations, as in this sentence:

- "For new patients, a patient file is created."

Obviously, we would like to create a condition here. The patient file only needs to be created when we are dealing with a new patient. In case of a known patient, we can skip this step. But, to recognize this condition a thorough semantic analysis and domain knowledge are needed and we are not able to detect this conditions simply using the syntactic means that are provided by the tools we used. However, in [93] researchers of the University of Pennsylvania analyzed the amount of explicit[15] and implicit discourse relations within the *Penn Discourse TreeBank*[97]. According to them temporal and contingency relations, which are most important for us, are stated explicitly in 79.55% and 46.75% of the cases, respectively. While the high number of explicit discourse relations for temporal relations is promising a low degree of contingency relations is problematic. Nevertheless, we are mostly interested in conditional relations, which are only a subset of the contingency relations. Therefore, we expect the fraction of explicitly stated conditional relations to be higher. Further, we can expect a process description to be more structured and contain more explicit connectives than the corpus of Wall Street Journal articles, which was used to build the *Penn Discourse TreeBank*.

[15] A discourse relations is explicit if it is expressed through syntactic elements and implicit if it is "not related explicitly by any of the syntactically defined set of Explicit connectives" [98]

3.1.2. Atomicity

The question of Atomicity deals with the question of which parts of a sentence should be mapped to a BPMN Task. It is possible that some sentences require a 1–1 mapping of activities, but within our test data set we also encounter complex sentences like these:

- "Sometimes, we buy details for cold calls, sometimes, our marketing staff participates in exhibitions and sometimes, you just happen to know somebody, who is interested in the product."
- "The GO or the MPON confirms the invoice with payment advice to the MPOO or the MSPO, or the GO or the MPON rejects the invoice of the MPOO or the MSPO."

While we would expect the first case to be mapped to three distinct activities in the resulting process model ("buy details", "participate in exhibitions", and "happen to know somebody"), the second sentence even requires four activities to be represented correctly (GO: "confirm invoice", "reject invoice"; MPON: "confirm invoice", "reject invoice"), because of the conjunctions.

On the other hand a full action can also be split up over several sentences as in this example also taken from the test data set described in section 4.1.

- "Then we start calling the customer."
- "That is done by the call center staff."

Although these are two full sentences, the reference which is made by "That" indicates that the second sentence simply adds information to the first one. This is also represented in the corresponding model within the test data set, where the activity "call customer" can be found within the Lane "Call Center Staff". Therefore, we have to check whether we can combine the information of two sentences and create a single activity.

Another aspect is that a sentence can consist of several clauses, which can (but do not have to) represent actions important for the process model to generate. The following sentence contains two relative clauses:

- "If the treasurer accepts the expenses for processing, the report moves to an automatic activity that links to a payment system."

The first relative clause ("If the treasurer accepts the expenses for processing") represents an important condition which should be represented in a generated process model, while the second relative clause ("that links to a payment system") just explains further characteristics of the "automatic activity". Of course, this information can also be vital, but we would probably not include it into a process model in order to keep it concise and easily understandable.

3.1.3. Relevance

Thus, the usage of relative sentences does not only pose the question of whether it is the part of an action or an action itself, but also whether the relative sentence is relevant for the generated model or not. The following sentence are also illustrating the issue of relevance:

- "For instance, a sales person on a trip rents a car."
- "[...] is given a chance to edit it, for example to correct errors or better describe an expense."

The author of these sentences tried to create a more vivid text by using examples to clarify the abstract concepts. Unfortunately, this information is unwanted within a process model, as a model is supposed to be a generalized abstract representation of the underlying process. Therefore, sentences which give examples should be ignored during the process model generation.

Another issue of relevance which can be observed quite often is that the process is described on a meta level. This means that the author does not explain the steps which have to be conducted, but rather described the process model.

- "After the Process starts, a Task is performed to [...]"
- "If the design fails the test, then it is sent back to the first Activity."
- "After the payment is confirmed, the process ends."
- "The process starts when a customer submits [...]"

The information conveyed using these sentence will be explicitly presented in the process model, e.g. through the shape of a node within the process model or simply by an appropriate message flow, but parsing the text becomes harder. These few examples illustrate that an effective filtering technique is needed to reliably identify and ignore certain relative sentences, examples and meta descriptions.

3.1.4. Referencing

In section 3.1.2 we demonstrated that the information of two sentences has to be combined in order to determine the following activity:

- "[The call center staff]$_{\text{Agent}}$ calls [the customer]$_{\text{object}}$"

But, this example also demonstrates the problem of anaphoric references (section 2.2.2). The personal pronoun "we" is a cathaphora in this case, as it refers to "the call center staff" introduced in the next sentence. In contrast the determiner "That" is an anaphora, referring to the previously mentioned action "calling the customer". Within our test data set we also found several exaphoras, as in the sentence "*There* are circumstances, where [...]". Here the word "There" refers to a concept which is not captured linguistically within the text, but rather is determined from its context. Another special position is taken by the the word "it" which can sometimes be used as a pronoun, similar to "he" or "she",

but also pleonastically as in "Sometimes it also happens that [...]". Although, in [70] it was shown that such pleonastical usages of "it" can be identified with high accuracy. The problem of implementing this algorithm and treating such special instances of "it" remain. Thus, in order to determine a fully qualified actor and to be able to combine sentences, an accurate and robust anaphora resolution mechanism is required.

A different kind of referencing is used to describe alternative paths within process models. Whenever two or more alternative branches, which are not immediately joined again, have to be described, the previous position has to be referenced. By analyzing our test data, three different usage pattens were identified:

Forward references occur whenever an alternative path is described, where the same goal as in the main process flow can be reached. The intention is then to join the flow while moving forward with the description. As an example consider the following excerpt of of a textual process description:

> "Of course, asking the customer whether **he is generally interested** is also important. If this is not the case, we leave him alone, except if the potential project budget is huge. Then the head of development personally tries to acquire the customer. **If the customer is interested in the end**, the next step is [...]"

In the first sentence two new branches are opened. The customer could be interested or not. The following two sentences describe an attempt to convince the customer, though he did not show interest, by having the head of development contact him. If the manager succeeds and convinces the customer the process proceeds as if the customer had been interested in the beginning. Therefore, the flows from the two decision points where the customer can indicate that he is interested should be joined.

Backward references were used to create loops. Whenever the same task must be per-

formed again or loops to check a certain business object until it has an appropriate level of quality need to be created we found sentences like these:

- "Otherwise, he calls the next customer."
- "If receipts are missing or do not match the report, he sends it back to the employee."
- "Then I correct the description and submit it again for consideration."

An interesting characteristic all these sentences have in common is a word signaling the repetition or the backward loop, like "next", "back" or "again". We will use this feature to detect backward references within texts.

Jumps occur when a process can have different outcomes. After a split of the process flow was described, the author of a textual process description has to follow one of those paths to the end. Afterward, he has to reference the position of this split again, before he can start to describe the other branch.

- "[..] the supervisor can accept or reject the report [...]. If the supervisor rejects the report, the employee, who submitted it, [...]. If the supervisor approves the report, it goes to the treasurer."
- "If all is in order, the treasurer accepts the expenses for processing. If receipts are missing or do not match the report, he sends it back to the employee. [...] If the treasurer accepts the expenses for processing, [...]"

All three types of textual references have to be recognized and taken into account for the process model creation. Problems are whenever the reference is not made explicit or it is described on a meta level ("the first task has to be performed again" - see section 3.1.3). In those cases the reference cannot be inferred by syntactic means, but it requires knowledge of the semantics and problem domain.

Another issue which is strongly dependent on domain knowledge is the recognition of an end or join of a block. Consider the following example sentences:

- "The GO examines the application of the MSPN. The GO rejects the application of the MSPN or the GO confirmes the application of the MSPN. The GO assigns the MSPN."
- " In case the customer is premium, the process will link to an extra problem fix process (this process will not be detailed here)."
- "If the loan is small and the customer is low risk, the loan is approved. If the customer is high risk, the loan is denied. If the customer needs further review or the loan amount is for $10,000 or more, the request is sent to the approver Web service. The customer receives feedback from the assessor or approver."

In both cases a condition is the appropriate means to employ within the process model. A human can now infer that in case of a rejection of the application the process is finished an the task "assign MSPN" will not be executed. But, in order to recognize this implicit relationship knowledge of the semantics, of the word "reject", and domain knowledge is necessary.

The same applies to the second example. It is unclear whether the process is finished after linking to the "extra problem fix process" or if this path of the process should be joined with the other branch.

Example three highlights a similar issue. The text first describes three possible branches of the flow, each with an if-statement. The last sentence explains that the customer receives a feedback, but it is not clear whether this applies to the last if-statement or if the customer receives feedback in any case. Intuitively, adding a gateway and always sending feedback to the customer appears to be the correct way, but to automatically detect this pattern a deep semantic analysis of the text and its domain is required. Addi-

tionally, in a static system without learning mechanism like ours, the approach is always dependent on the amount of knowledge encoded a priori.

3.1.5. Solution Strategy

As shown, the automatic generation of process models from text poses problems which cannot be solved easily for the general case. However, as the goal of the proposed approach is not to replace a human modeler, but to complement him, we can accept some deviations in the generated model and leave the customization to the user.

Nevertheless, to overcome the issues outlined in this section we propose a syntactic and semantic analysis of the text input. Furthermore, sentences cannot be parsed in isolation of each other, but an overarching text level analysis has to be conducted, e.g., to resolve anaphoric and textual references and to combine information where possible. To achieve this we employ four computational linguistic tools, which were introduced in section 2. These tools are the Stanford Syntax Parser, WordNet, FrameNet, and an Anaphora Resolution Technique. Furthermore, the analysis will be conducted in different stages, which are independent of each other. To enable an efficient data transfer between the stages we will store the extracted information in a *World Model*. We borrowed this terms from the field of robotics, where the structural representation of all information known to a robot is usually referred to as a World Model (see e.g. [112]). A structural overview of the approach taken can be seen in figure 9. As shown, we also considered the possibility of using other sources of information than text documents. This could be structured information in other information systems, e.g., Content Management Systems (CMS) or Enterprise Resource Planning (ERP) software. Moreover, existing models can also be analyzed and taken into account during the generation procedure. Furthermore, organizational and data models are another possible output which can be created from the collected information.

Lastly, we base our transformation approach on five basic assumptions in order to restrict the set of required analyses. These assumptions are:

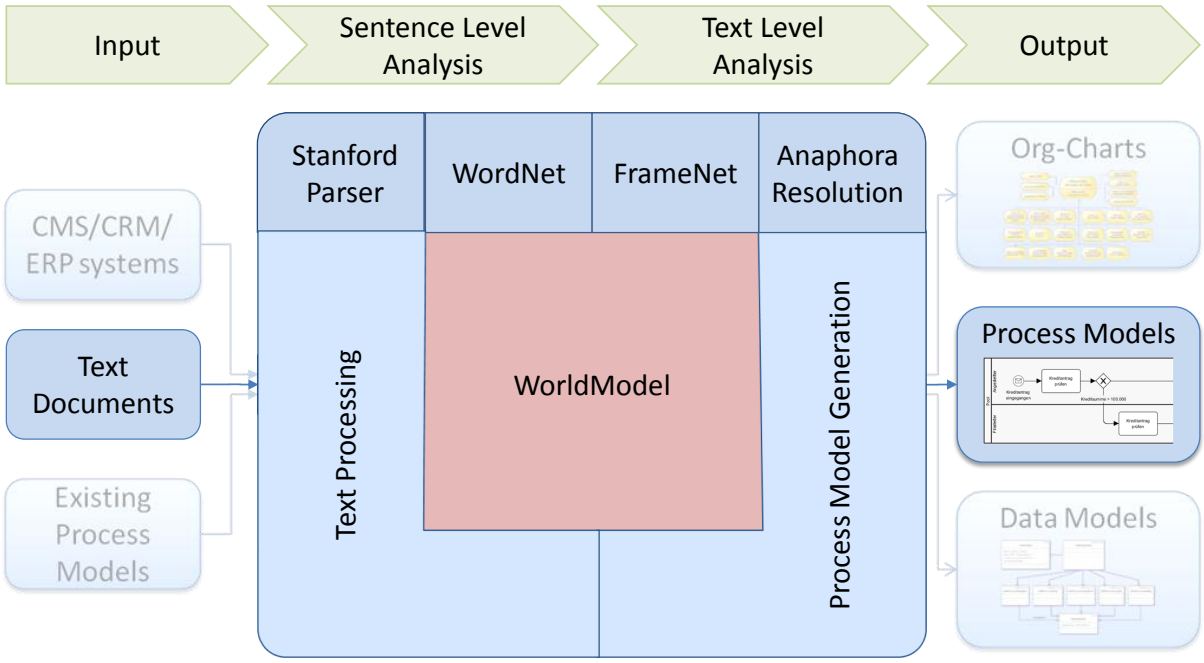

Figure 9: Structural overview of the presented transformation approach.

- The text actually describes a process and not, e.g., data objects or organizational structure.
- The text contains no questions.
- The text describes the process as perceived by an involved Actor and not on a meta level ("the next task is ...", "then the process...").
- The Process is described sequentially, meaning that the actions in adjacent sentences are related to each other.
- If a non sequential description is needed, the textual jump is made explicit in the text.

While the first assumption is self-evident, the second one is used to narrow the scope of dependencies and sentence structures to consider, as special tags exist for questions.

Furthermore, questions are expected to be very rare in general. Within our test data set no questions were contained and, therefore, this limitation appeared to be practical. While we will present mechanisms to detect and reduce the impact of meta-descriptions on the generated process model, in general a directly involved style of writing is preferred. By demanding a sequential description, we make sure that a process as a whole can be understood by our system. This assumption is different from e.g. [133] where the sentences can be processed independently of each other, but instead connections have to be expressed by the input and output of activities. While, the last assumption is the most restrictive one and was actually violated by several elements within our test data set it relieves us from conducting a deep semantic analysis. Otherwise, this analysis would be required, in order to detect textual dependencies which are not visible by analyzing the surface structure of the sentences. However, violating an assumption does not render the parsing of the text impossible, but will merely reduce the quality of the transformation result. Hence, we are confident that viable results are achievable despite these constraints.

3.2. Intermediate Data Structure (World Model)

To obtain a structured representation of the knowledge we extract from the text, we decided to store it in a *World Model*, as opposed to a direct straight through model generation. This approach was also taken by most of the other works which built a similar system [140, 118, 34, 22].

The data structure used by the approach of the University of Rio de Janeiro [22] was taken from the CREWS project [1]. The authors argue that it is suited well for this task as a scenario description corresponds to the description of a process model. Therefore, we used the CREWS scenario metamodel as starting point for the creation of our own intermediate data structure.

However, we modified several parts as, e.g., a distinction between a normal and an exceptional scenario was not necessary, because both can be expressed within the same process model. Furthermore, we renamed “Agent” to “Actor” so it is not confused with the

passive agent relation mentioned earlier. Furthermore, we explicitly represent connections between the elements using the class "Flow". During the construction of the World Model we introduced "SpecifiedElement" and "OriginatedElement" as further abstraction layers to avoid redundancies. Additionally, we explicitly considered traceability as a requirement. Thus, attributes which were required to relate an object to the text, sentence and word is was extracted from were added to our World Model.

The four main elements of our World Model are Actor, Resource, Action, and Flow. The following descriptions give an overview of the elements contained in the World Model and their purpose.

Originated Elements represent the most abstract class within our data structure. It represents an entity which was extracted from the text. In order to enable traceability it stores a link to the text and the sentence to which it relates. These properties are shared by all elements contained in the World Model.

Specified Elements are a further refinement of an Originated Element. They are used to describe structures which can be traced down on a word level. Thus it also stores a reference to the Word within its sentence. Furthermore, at handles and manages Specifiers.

Specifiers are part of an object and can be used to further describe or refine it. This includes, e.g., prepositional phrases (PP), compound parts of a noun compound (NN), relative clauses (RCMOD), infinitival modifiers (INFMOD), or numeric modifiers (NUM). If it is possible, an object is also extracted an stored if it directly belongs to his specifier, e.g., for prepositional phrases ("to the customer").

An Extracted Object represents a static parts of the text extracted from noun phrases. It corresponds to the element "Object" within the CREWS metamodel [1]. It can also refer to an anaphoric reference in which case the Extracted Object will have to be resolved. It is further refined by the sub-classes Actor and Resource.

An Actor describes an acting entity within the text. This can be a person, a software system an organization or department. Additionally, all entity which take the places of a subject within the text are classified as Actors, but are declared “unreal” if the first property is not fulfilled. A sub-class of these “unreal”-Actors are Actors which related to modeling concepts on a meta level (e.g., a process, a task, an activity). Such Actors are called ”meta”-Actors and are also marked.

Resources do not make this distinction. They possess in an objective position within the sentence and are not related to a person, software system, organization, or department. Like the Actor, it derives all its properties from the class ”Extracted Object” and can thus be resolved and further specified.

Actions are the core of the World Model. They represent an Activity extracted from the textual process specification. They can possess an Actor as subject and an Extracted Object as object. Furthermore, we incorporated the possibility of a link. Therefore, an Action can link to another Action within the World Model an establish textual links (Issue 4.2). As an Action can be supplemented with an open clausal complement, which possesses the same properties, another Action “xcomp” can be added.

Flows are used to represent the process flow and the relations of the Actions to each other. One side of the flow links to a single Action and to multiple Actions on the other side. A flow is directed and can be either defined as a “split” or a “join”. Depending on the direction, the single object either is the source or the target of that flow. The type of the flow indicate how the process flow is split or joined. We defined four different flow types: concurrency, sequence, choice and exception. The first three are analogous to the definition in the CREWS metamodel [1]. From these types we are able to determine the appropriate BPMN construct in the process model generation phase.

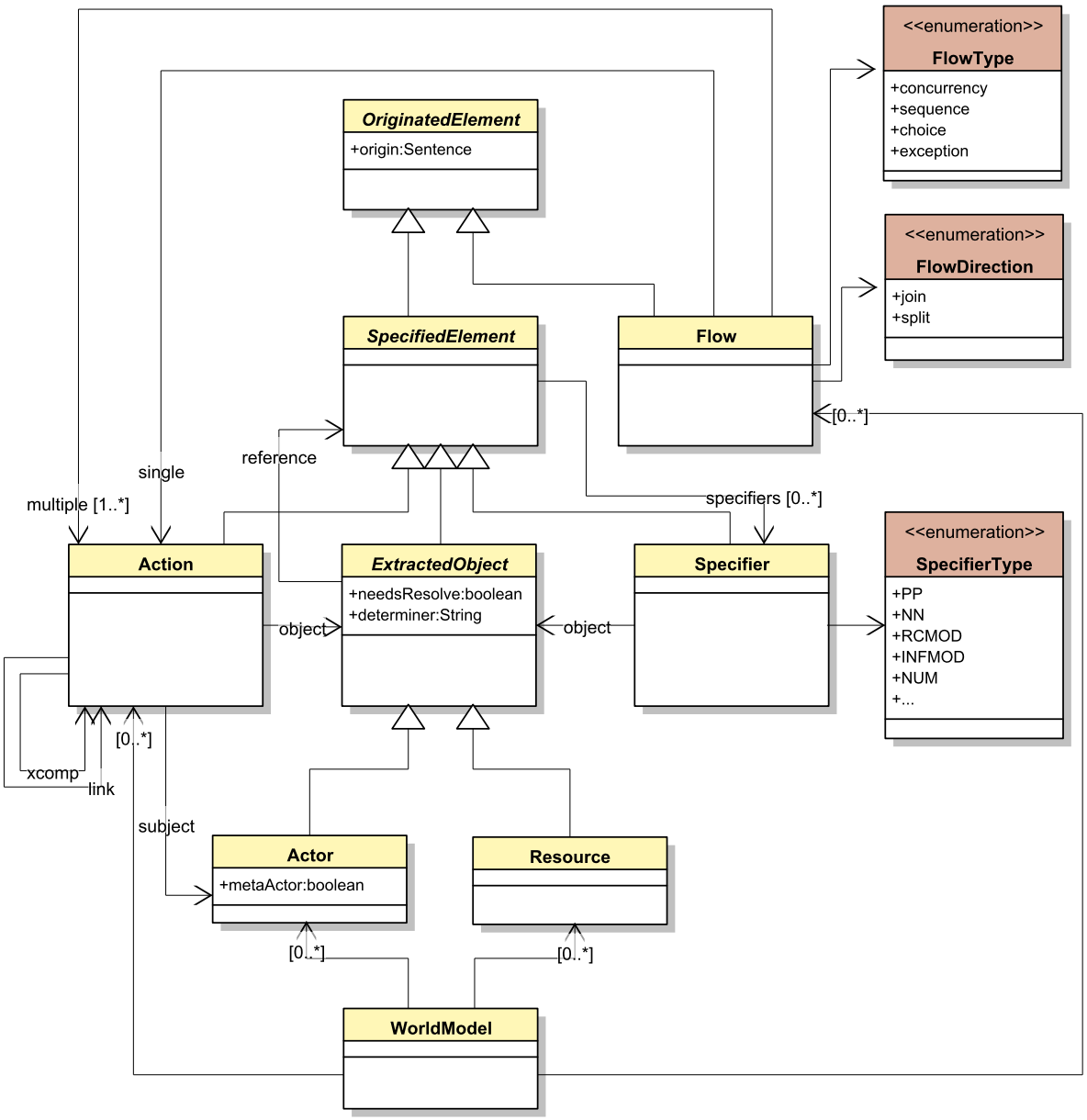

Figure 10: Structure of the intermediate data structure (World Model).

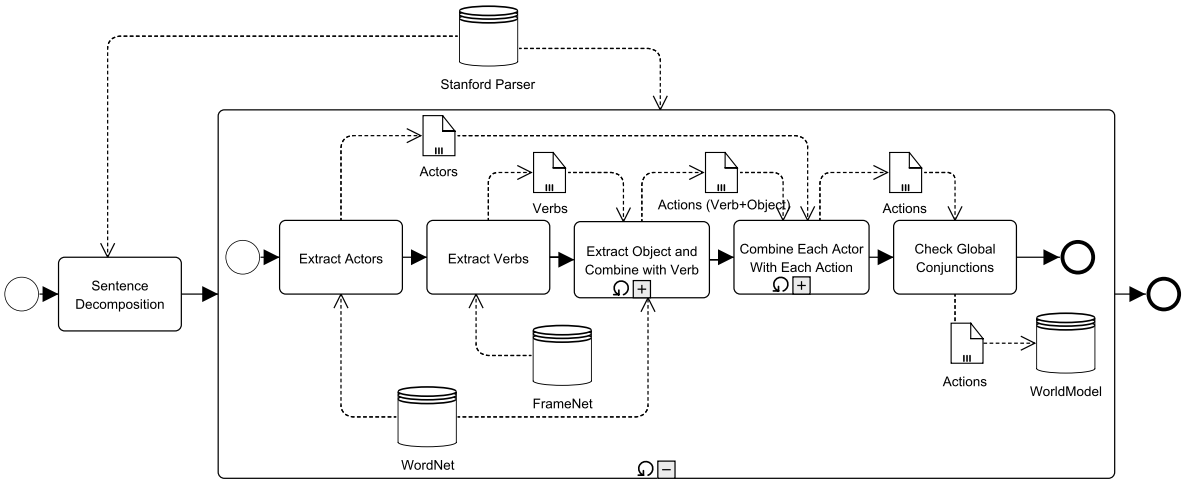

Figure 11: Structural overview of the steps of the Sentence Level Analysis.

By defining the World Model the manner described, we are able to reflect the information extractable through the syntactic and semantic analysis proposed in the previous section appropriately. This World Model will be used throughout all phases of our transformation procedure. Each phase is allowed to access, modify and add data. All objects are accessible through the World Model object which will be used in several algorithmic description in the following sections.

3.3. Sentence Level Analysis

Within this section we will describe the sentence level analysis using pseudo-code. The extraction procedure consists of several steps which are outlined as a BPMN process model in figure 11. Within this overview the different components upon which our transformation procedure builds and their usage are shown using Data Sources.

3.3.1. Text and Sentence Decomposition

The text is processed in several stages. The first stage is a tokenization of the text which includes splitting it up in individual sentences. The challenge here is to distinguish a period which is only user for an abbreviation (e.g. M.Sc.) from a period marking the

end of a sentence. This task can be handled by the document preprocessor included in the *Stanford Parser* library, which is a deterministic scanner created using the JFlex scanner generator[16].

Afterward, each sentence is parsed by the *Stanford Parser*, using the factored model for English, which is provided with the API. We decided to utilize the factored model instead of the PCFG (Probabilistic Context Free Grammar) model, because it provided better results in determining the dependencies between the sentence elements and markers as "if" or "then", which are important for the process model generation. The resulting Parse Tree and the collapsed dependencies [24] are then passed to algorithm 1, which splits up complex sentences into individual phrases. This is accomplished by scanning the tree for sentence tags (S,SBAR,SINV)[76] on the highest level of the Parse Tree and within prepositional (PP), adverbial (ADVP), and noun phrases (NP) they contain (line 1). If the sentence is simple and does not contain any sub-sentences, an extraction procedure can be triggered directly (line 3). If it contains only one sub-sentence (line 4), then this sentence is analyzed by applying the sentence decomposition algorithm recursively (line 6). For this analysis the typed dependencies have to be filtered as only those dependencies where governor and dependent are contained within the sub-sentence are relevant. Afterward, the sub-sentence is deleted from the syntax tree (line 7 and 8) and if the remaining parse tree and dependencies contain one of the necessary relations to extract an Action (line 9) it is also analyzed. If more then one sub-sentence was found we are probably dealing with several main clauses which can be analyzed independently of each other (lines 13-16). This algorithm detects top-level sentences and ignores relative sentences which could be found within one of the sub-sentences. Relative sentences will be analyzed in a later stage, in algorithm 4. By following this approach we tackle the problem of complex sentences (Issue 2.1) as described in section 3.1.2

[16]Klein, G. 2001. JFlex the fast scanner generator for Java. http://www.jflex.de/.

Algorithm 1 Sentence Decomposition

Require: **Tree** parsedSentence, **TypedDependencies** dependencies

List<Tree> subSentences ← determineSubSentences(parsedSentence)
if |subSentences| = 0 **then**
    extractElements(parsedSentence,dependencies)
else if |subSentences| = 1 **then**
    TypedDependencies filteredDependencies ← filter(dependencies,subSentence[0])
    sentenceDecomposition(subSentence[0],filteredDependencies)
    Tree restTree ← parsedTree \ subSentence[0]
    TypedDependencies restDependencies ← dependencies \ filteredDependencies
    if {"nsubj","nsubjpass","agent","dobj"} ∩ restDependencies ≠ ∅ **then**
        extractElements(restTree,restDependencies)
    end if
else
    for ∀ Tree subSentence ∈ subSentences **do**
        TypedDependencies filteredDependencies ← filter(dependencies,subSentence)
        sentenceDecomposition(subSentence,filteredDependencies)
    end for
end if

3.3.2. Element Extraction

After the sentence was broken down into individual constituent phrases actions can be extracted. That is accomplished using algorithm 2. It first determines whether the *parsedSentence* is in active or passive voice by searching for the appropriate grammatical relations (line 1), which enables use to handle active and passive sentences alike (Issue 1.1). Afterward, all Actors and Actions are extracted using the algorithms 3 and 4. To overcome the problem of example sentences mentioned in section 3.1.3 (Issue 3.2) the actions are then filtered (line 4). This filtering method simply checks whether the sentence which contains the action contains a word of a stop word list called *example indicators*. This and all other stop word lists used within the algorithm are listed in Appendix C. The next block (lines 6-16) first extracts all objects from the phrase (line 7). In line 8 these objects and the actors are then used within *filterSpecifiers* to remove Specifiers from the Action which contain any. This is necessary e.g. in passive sentences where the actor is mentioned in a prepositional phrase headed by the word "by". For such sentences a specifier is created as it is a prepositional phrase attached to the verb phrase. At the same time the corresponding Actor is also extracted from that phrase. Applying the filter solves this problem of redundancy. Afterward, each Action which was extracted from the sentence is combined with each Object. The same is done with all Actors in the following block (lines 18-26). This procedure is necessary as an Action is supposed to be atomic according to section 3.1.2 (Issue 2.1). Therefore, a new Action has to be created for each piece of information as illustrated in the following example sentences. In each sentence the conjunction relation which causes the extraction of several Actors,Actions or Resources is highlighted. As a last step all extracted Actions are added to the World Model.

- "Likewise the old supplier **creates and sends** the final billing to the customer." (Action)
- "It is given either by **a sales representative or by a pre-sales employee** in case of a more technical presentation." (Actor)

- "At this point, the Assistant Registry Manager puts **the receipt and copied documents** into an envelope and posts it to the party." (Resource)

The determination of Actors from a sentence, which is needed for 2 is straightforward (see algorithm 3). If the sentence is written using the active voice, we determine the Actor from the "nsubj" relation. In case of a passive sentence, "agent" is used instead. It is also possible that no such relation can be found, e.g. in an imperative sentence ("Send the report to the customer!") or in passive sentences. In those cases the variable "mainNode" is not assigned (represented as ϵ) and no actor can be determined. If an Actor was identified successfully, further grammatical relations are analyzed and a new Actor object is created (line 14). The next section will explain this procedure in more detail. The last step is to check whether the identified actor is also part of a conjunction relation (line 16) and to return the resulting list of Actors (line 18).

Algorithm 4 demonstrates how an Action is identified within a phrase. It works similar to the Actor extraction in algorithm 3. A difference is that if an "nsubj" relation was found it still has to be checked against all "cop" relations if they are present in the text. This issues arises due to the collapsing of *Stanford Dependencies*. As illustrates in table 3, the governor of the nsubj relation is not the copula verb, but its referent (interested in this case). Nevertheless, for the later analyses it is more appropriate to store "is" as the main verb of the Action. Therefore, we perform the check in lines 6-9. Another difference is that while it is possible not to instantiate an Actor, an Action is always required. Thus, even if no "nsubj" relation can be found (e.g. an imperative) it is possible to determine the main predicate in an "dobj" relation. In case of a passive sentence, the action can be found within the "nsubjpass" relation. In order to deal with Issue 2.3 and 3.1 we then proceed by analyzing relative clauses attached directly to the verb (line 19). In contrast to that, relative clauses attached to noun phrases will not be analyzed, but added as a Specifier. This heuristic is motivated by the observation that relevant relative clauses are usually attached to verbs instead of nouns. Similar to the Actor, the algorithm ends by

Algorithm 2 Extract Elements

Require: **Tree** parsedSentence, **TypedDependencies** dependencies

```
1: boolean active ← isActive(parsedSentence,dependencies)
2: List<Actor> actors ← determineActors(active,sentence,dependencies)
3: List<Action> rawActions ← determineActions(active,sentence,dependencies)
4: removeExampleSentences(actions)
5: List<Action> actionsWithObject ← new List<Action>
6: for ∀ Action action ∈ rawActions do
7:     List<ExtractedObject> objects ← determineObject(active,sentence,dependencies, action)
8:     filterSpecifiers(action,objects,actors)
9:     if |objects| > 0 then
10:        for ∀ ExtractedObject object ∈ objetcs do
11:            actionsWithObject ∪ (action+object)
12:        end for
13:    else
14:        actionsWithObject ∪ rawActions
15:    end if
16: end for
17: List<Action> finalActions ← new List<Action>
18: for ∀ Action action ∈ actionsWithObject do
19:    if |actors| > 0 then
20:        for ∀ Actor actor ∈ actors do
21:            finalActions ∪ (actor+action)
22:        end for
23:    else
24:        finalActions ∪ actionsWithObject
25:    end if
26: end for
27: addToWorldModel(finalActions)
```

Algorithm 3 Determine Actors

Require: **boolean** active, **Tree** parsedSentence, **TypedDependencies** dependencies

```
List<Actor> result ← new List<Actor>
if active then
    TypedDependency nsubj ← findDependency("nsubj", parsedSentence, dependencies)
    if nsubj ≠ ϵ then
        TreeGraphNode mainNode ← nsubj.dependent()
    end if
else
    TypedDependency agent ← findDependency("agent", parsedSentence, dependencies)
    if agent ≠ ϵ then
        TreeGraphNode mainNode ← agent.dependent()
    end if
end if
if mainNode ≠ ϵ then
    Actor actor ← createActor(mainNode, dependencies)
    result ∪ actor
    result ∪ checkConjunctions(actor,active,dependencies)
end if
return result
```

checking if the Action is part of any conjunction relation (line 21) and returns the result list (line 22).

mark(interested-5, If-1)	advcl(presentation-17, interested-5)
det(customer-3, the-2)	det(end-8, the-7)
nsubj(interested-5, customer-3)	prep_in(interested-5, end-8)
cop(interested-5, is-4)	

Table 3: Stanford dependencies for the copula phrase "If the customer is interested in the end, [...]".

Algorithm 5 describes the procedure of finding and adding elements which are in a conjunction relation. It was used within the algorithms 3,4, and 6. As it has to be able to handle Actors, Resources, and Actions as input, it uses their common superclass Specified Element (see 3.2) so it can be applied and used for all of them. It first determines all typed dependency relations with the name "conj" (line 2). Line 3 then handles Action which have an associated open clausal complement (xcomp) and checks whether they are the governor of a conjunction relationship. If a relation with either the element itself or the open clausal complement of an Action as governor was found and the element is not part of a copula relation (line 4), a new element has to be created. The check for the copula relation has to be performed, as it is possible to create an object from the copula element and have a conjunction assigned to it, as in this example sentence:

> "Established underwriters are careful of their reputation and will not handle a new issue [...]"

In this sentence a Resource is created from the noun phrase "careful of their reputation" and careful is in a conjunction relationship with "handle". But this relation was already captured during the conjunction analysis of the Action. Then, again, we make a distinction between a match of the open clausal complement and all other cases (line 6), as

Algorithm 4 Determine Actions

Require: **boolean** active, **Tree** parsedSentence, **TypedDependencies** dependencies

```
List<Action> result ← new List<Action>
if active then
    TypedDependency nsubj ← findDependency("nsubj", parsedSentence, dependencies)
    if nsubj ≠ ϵ then
        TreeGraphNode mainPredicate ← nsubj.governor()
        TypedDependency cop ← findDependency("cop", parsedSentence, dependencies)
        if cop.governor() = mainPredicate then
            TreeGraphNode mainPredicate ← cop.dependent()
        end if
    else
        TypedDependency dobj ← findDependency("dobj", parsedSentence, dependencies)
        TreeGraphNode mainPredicate ← dobj.governor()
    end if
else
    TypedDependency nsubjpass ← findDependency("nsubjpass", parsedSentence, dependencies)
    TreeGraphNode mainPredicate ← nsubjpass.governor()
end if
Action action ← createAction(mainPredicate, dependencies)
checkSubSentences(action,dependencies)
result ∪ action
result ∪ checkConjunctions(action,active,dependencies)
return result
```

we do not want to lose the information stored in the parent-action of the open clausal complement. Therefore, we first create a copy of the parent object and simply replace the xcomp element (line 7 and 8). This is necessary to create the desired actions as illustrated by the following example.

- "The customer then has the chance to check the contract details and based on this check **may decide to** either **withdraw** from the switch contract **or confirm** it."

Using algorithm 5 two activities are correctly recognized within this sentence:

1. $(\text{The customer})_{\text{actor}}$ $(\text{may decide})_{\text{action}}$ $(\text{to withdraw})_{\text{xcomp}}$ ...
2. $(\text{The customer})_{\text{actor}}$ $(\text{may decide})_{\text{action}}$ $(\text{to confirm})_{\text{xcomp}}$ ...

The last steps of the algorithms are to add the new element to the result set and call itself recursively, as the new element could also be in a conjunction relation (e.g., "A and B and C perform an action"). Furthermore, the relation between the two elements is stored globally in a structure called *Conjunctions*, which is needed e.g. in algorithm 6, which is introduced next.

Correctly determining the object of an Action is the most complex operation. As shown in algorithm 6 the first step is to determine the object of the open clausal complement of the previously extracted Action (line 4). Next, we try to determine a direct object by checking the "dobj" relation (line 7). If no appropriate relation was found, a direct object can still be determined by consulting the globally available conjunction relations of the action, which were created as outlined in algorithm 5. This is motivated by sentences like those:

- "Otherwise, the matter details (types of action) **are captured and provided** to the Cashier [...]."
- "[...] a Task is performed to **locate and distribute** any relevant existing designs [...]."

Algorithm 5 Check Conjunctions

Require: **boolean** active, **Tree** parsedSentence, **TypedDependencies** dependencies, **SpecifiedElement** element

```
List<SpecifiedElement> result ← new List<SpecifiedElement>
for ∀ TypedDependency conj ∈ findDependencies("conj",parsedSentence,dependencies,element) do
    xCompMatch ← element instanceof Action ∧ conj.governor() = ((Action)element).getXComp()
    if conj.governor() = element ∧ !partOfCop(conj.governor()) ∨ xCompMatch then
        TreeGraphNode conjNode ← conj.governor()
        if xCompMatch then
            Action newElement ← ((Action)element).clone()
            newElement.getxComp() ← createAction(conjNode,dependencies)
        else
            if element instanceof Action then
                Action newElement = createAction(conjNode,dependencies)
            else
                ExtractedObject newElement = createObject(conjNode,dependencies)
            end if
        end if
        result ∪ newElement
        result ∪ checkConjunctions(active, parsedSentence, dependencies, newElement)
        conjunctions ∪ new Conjunction(conj,element,newElement)
    end if
end for
return result
```

For example in the second sentence the direct object "any relevant existing designs" is directly attached to "distribute", while no direct object relation can found for "locate". By also checking the conjunctions we solved this problem and correctly attach the object "any relevant existing design" to locate. An important constraint for this procedure is that the object has to be placed after the conjunct verb to avoid false positives. This procedure is not necessary if the open clausal complement (xcomp) of the given action already has an object assigned. As demonstrated by the following sentences:

- "The customer then has the chance **to check the contract details** and [..] may decide to either withdraw from the switch contract or confirm it."

If no direct object was found (line 11) it is possible that we are dealing with a copula sentence, as in this example:

- "Customer service is a shared service center [...]"

We are then able to extract an object from the governor of the "cop" relation (line 16). If the sentence was written in passive voice the "nsubjpass" relation can be used to determine the object as shown in the lines 20 and 21. Similarly to the Actors, an Object does not necessarily have to exist (line 23), but if it a corresponding TreeGraphNode was found an ExtractedObject is created (line 24), added to the result list (line 25), and checked for conjunctions (line 26). In line 24 we defined the result to be of the type ExtractedObject, as the algorithm createObject, which is explained in detail in the next section, is able to create either an Actor or a Resource.

3.3.3. Element Creation and Semantic Analysis

In the last section, we explained how to decompose a text and how to detect and combine its main elements. This section describes how the Action, Actor, and Resource objects are created within the algorithms 3,4, and 6 and how WordNet and FrameNet are employed in order to add semantic information to the objects. In order to access these

Algorithm 6 Determine Objects

Require: **boolean** active, **Tree** parsedSentence, **TypedDependencies** dependencies, **Action** action

List<ExtractedObject> result ← new List<ExtractedObject>
TreeGraphNode objectNode ← ϵ
if action.getXComp() $\neq \epsilon$ **then**
    action.getXComp().getObject() ←
    determineObjects(active,parsedSentence,dependencies,action.getXComp())
end if
if active **then**
    TypedDependency dobj ← findDependency("dobj", parsedSentence, dependencies,action)
    if dobj $= \epsilon \wedge$ action.getXComp().getObject $= \epsilon$ **then**
        TypedDependency dobj ←
        findDependency("dobj", parsedSentence, dependencies, conjunctions)
    end if
    if dobj $\neq \epsilon$ **then**
        objectNode ← dobj.dependent()
    else
        TypedDependency cop ← findDependency("cop", parsedSentence, dependencies,action)
        if cop.governor().parent().parent() = NounPhrase **then**
            objectNode ← cop.governor()
        end if
    end if
else
    TypedDependency nsubjpass ← findDependency("nsubjpass", parsedSentence, dependencies)
    TreeGraphNode objectNode ← nsubjpass.dependent()
end if
if objectNode $\neq \epsilon$ **then**
    ExtractedObject extractedObject ← createObject(objectNode, dependencies)
    result $\cup$ extractedObject
    result $\cup$ checkConjunctions(extractedObject,active,dependencies)
end if
return result

lexical database we employed the JWNL library [17] and the FrameNet API [18] developed by Nils Reiter. As FrameNet API in its current version[19] was not able to handle FrameNet 1.5 we adjusted it to the new format and extended it to be able to extract Valence Patterns and Units.

Algorithm 7 describes the main creation steps for an Actor. First, a new Actor object is created (line 1). During this creation, the Actor is also provided with a link to the sentence that is currently under investigation and its position within that sentence. This was not explicitly shown in algorithm 7, but is important to mention as this ensures traceability between the text and the later model. In line 2 WordNet is used to check whether the node is a "real"-Actor, as a person, an organization or a software system. To perform that check WordNet is used as a general purpose ontology. First the value of the node is looked up in WordNet and the hypernyms for all its senses are traversed. If one of those hypernyms matches a concept of a previously defined stop word list "true" is returned. As all other stop words list used during the transformation, it can be adapted by the user. For the test data set provided with this thesis, three concepts, namely "person", "social_group", and "software_system" (*Real Actor Determiners*), were sufficient. Additionally, a stop word list which directly classifies a noun as a person or organization (*Person Corrector List*)[20] is also provided. This is necessary as domain specific abbreviations and proper nouns are not contains in WordNet and would be misclassified otherwise. Additionally, Names of companies, departments or individual have to be stated in this lists as well, as we excluded the "Hypernym Instance" relation [82] for our search. The reason for this exclusion is the high rate of false positives which were produced otherwise. For example in the sentence "The post is advertised", "post" would be identified as a person due to the containment of Mr. Wiley Post in WordNet.

[17] http://sourceforge.net/projects/jwordnet/

[18] http://www.cl.uni-heidelberg.de/trac/FrameNetAPI

[19] The latest version available at the time of development was 0.4.1

[20] This list is fully listed in Appendix C

Additionally, a check is performed so certain construction involving cardinal numbers are not misclassified, e.g. “one of the physicians”. Specifically, we check if a prepositional phrase headed by “of” contains a “real”-Actor if we encounter a cardinal number as the head of the given noun phrase. If all those checks fail the actor is classified as “unreal” in line 3. This distinction is important for the later process model generation as “unreal”-Actors will not be represented by Lanes. In line 5, important noun specifiers are determined from the grammatical relations and added as a Specifier to the Actor. These specifiers comprise:

- possession modifiers and determiners (“poss”,“det” - e.g. the, a, my, our)
- adjectival modifiers (“amod” - e.g. electronic, potential, social-medical)
- noun compound modifiers (“nn” - e.g. **customer service** department, **service** center)
- indirect objects (“iobj” - e.g. He send **the customer** the bill)
- infinitival modifiers (“infmod” - e.g. a chance **to edit**)
- participial modifiers (“partmod” - e.g. customer data **being stored multiple times**)
- prepositional modifiers (“prep” - e.g. to the customer, into the system)
- relative clause modifiers (“rcmod” - e.g. somebody, who is interested in the product)

In the lines 6 and 7 it is determined whether the created Actor will need to be resolved or not. If the Actor turns out to be a personal preposition (“PRP”), e.g. he, she, it, they, or a determiner (“DT”) without any dependencies within the parse tree of the sentence (line 7; e.g. *That* is done by...) the Actor is marked for resolution. Additionally, certain nouns, e.g. “someone”, can be added to the *Relative Resolution Words* stop word list and

Algorithm 7 Create Actor

Require: **TreeGraphNode** node, **TypedDependencies** dependencies

```
1: Actor actor ← new Actor()
2: if !WordNet.isPersonOrSystem(node) ∧ !checkPP(node,dependencies) then
3:     actor.getUnreal() ← true
4: end if
5: determineNounSpecifiers(actor,dependencies)
6: if node.parent.value() ⊂ {"PRP","DT} ∨ node.value() ⊆ RelativeResolutionWords then
7:     if node.parent().parent().children().size() = 1 ∧ ∄ Specifier sp:sp.getHeadWord()="of" then
8:         actor.needsResolve() ← true
9:     end if
10: end if
11: if WordNet.isMetaActor(node) then
12:     actor.setMetaActor(true)
13: end if
14: return actor
```

will then also be resolved, except if they occur in a genitive compound with another noun (e.g. someone of the sales department).

Lastly, we consult WordNet again to determine if the Actor helps to describe the process on a meta level. To achieve this, we apply the same mechanism as for the "real"-Actor determination described earlier. The stop word list used in this case (Meta Actor Determiners) contained "step","process","case", and "state" for our test data set. By tagging these "meta"-Actors we can handle the associated action differently, as they are most likely part of a meta-sentence (Issue 3.3) After creating the Actor and adding the necessary semantic information it is returned in line 14.

Within algorithm 8 we proceed similarly to the creation of an Actor. The main difference is that *createObject* is able to return either an Actor or a Resource depending on the check in line 1. Furthermore, another attribute of an Actor, called *SubjectRole*, is explicitly set to false within this algorithm. Per default *SubjectRole* is set to "true", but

Algorithm 8 Create Object

Require: **TreeGraphNode** node, **TypedDependencies** dependencies

```
1: if WordNet.isPersonOrSystem(node) ∨ isPersonalPronoun(node) then
2:     Actor actor ← createActor(node,dependencies)
3:     actor.getSubjectRole() ← false
4:     return actor
5: else
6:     Resource resource ← new Resource()
7:     determineNounSpecifiers(resource,dependencies)
8:     return resource
9: end if
```

as this algorithm is only called for objects, it gives us the opportunity to mark this Actor with its role in the sentence, which in turn is useful for the relative reference resolution in section 3.4.1.

Another detail which is not shown in algorithm 8 is the marking of prepositional modifiers using FrameNet. Whenever a new prepositional modifier is extracted the simple resolution procedure shown in algorithm 9 is triggered. The goal of this procedure is to determine an appropriate *Frame Element* according to the frame semantics as defined by Fillmore [32]. While optimally a *Frame Element* can be defined for each phrase, the procedure fails often, simply because a lexeme is not captured within *FrameNet*. Therefore, we shift our interest onto the *Phrase Type*. In his book Ruppenhofer defined four main types: “core”, “peripheral”, “extra-thematic”, and “core-unexpressed” [106, page 26]. For our disambiguation technique, we added two more types for phrases: “genitive” and “unknown”. Algorithm 9 is then able to heuristically classify a given prepositional phrase. First, the algorithm check if the type “Genitive” can be assigned to the given prepositional phrase. This is the case when the head word of the phrase is “of” and the Specified Element is not an Action (which means it is either an Actor or a Resource, but in either case a noun). If the element is an Action or the head word differed a search heuristic is applied (lines 4-14). A new map which we use to count the occurrences of a

Frame Element is initialized in line 4. In line 5 the element is transformed into its base form using the stemming procedure and look up capabilities of WordNet. Afterward, all matching lexical units with an appropriate part of speech (Verb for Actions, Noun for Actors/Objects) can be extracted from FrameNet (line 6). Consulting the Annotation Corpus of FrameNet, it is then possible to search for all matching *Valence Units* (line 7 and 8). The phrase type of the *Valence Unit* and the Specifier are treated as equal (line 9) if the phrase type is PP (prepositional phrase) and the head words match. If the *Valence Unit* matches the specifier, the corresponding *Frame Element*, and its count are added to the "countMap". After all *Valence Units* were checked the *Frame Element* with the highest overall count is selected (line 15). This maximizes the probability of determining the appropriate *Frame Element*. If no matching *Frame Element* was found for the given lexeme the phrase type "Unknown" is assigned (line 17). Otherwise, we save the information in the Specifier object (line 19 and 20). These pieces of information can then be used during the process model generation to create clear and concise labels for activities. Furthermore, the next algorithm will demonstrate how certain *Frame Elements* can be used to add semantic information to Actions.

The creation of a new Action is described in algorithm 10. Similar to the creation of an Actor or Resource, several Specifiers are created and added (line 2), but different grammatical relations are considered:

- auxiliaries ("aux" - e.g. has, should, is)
- adverbial modifiers and adjectival complements ("advmopd","acomp" - e.g. leave alone, to come back)
- negation modifiers ("neg" - e.g. not, neither)
- the copula ("cop" - e.g. interested, huge)
- phrasal verb particle ("prt" - e.g. send out, build up, turn out)

Algorithm 9 Determine Frame Element

Require: **SpecifiedElement** element, **Specifier** specifier

```
1: if specifier.getHeadWord() = "of" ∧ !(element instanceof Action) then
2:     specifier.getPhraseType() ← GENETIVE
3: else
4:     Map<FrameElement,Integer> countMap ← new Map<FrameElement,Integer>()
5:     String lexeme ← WordNet.getBaseForm(element)
6:     for ∀ LexicalUnit lu ∈ FrameNet.getLexcialUnits(lexeme) do
7:         for ∀ ValencePattern vp ∈ FrameNet.getAnnotationCorpus().getPatterns(lu) do
8:             for ∀ ValenceUnit vu ∈ vp.getValenceUnits() do
9:                 if vu.getPhraseType() = specifier then
10:                    countMap ∪ (vu.getFrameElement(),vp.getTotalCount())
11:                end if
12:            end for
13:        end for
14:    end for
15:    FrameElement bestElement ← countMap.getBestElement()
16:    if bestElement = ϵ then
17:        specifier.getPhraseType() ← UNKNOWN
18:    else
19:        specifier.getFrameElement() ← bestElement
20:        specifier.getPhraseType() ← bestElement.getPhraseType()
21:    end if
22: end if
```

Algorithm 10 Create Action

Require: **TreeGraphNode** node, **TypedDependencies** dependencies

1: Action action ← new Action()
2: determineVerbSpecifiers(resource,dependencies)
3: TypedDependency xcomp ← findDependency("xcomp", dependencies)
4: **if** xcomp ≠ ϵ **then**
5: TreeGraphNode xcompNode ← xcomp.dependent()
6: action.getXComp() ← createAction(xcompNode,dependencies)
7: **end if**
8: **return** action

- relative clause modifiers ("rcmod" - e.g. for the case that minor corrective actions are required)
- prepositional modifers ("prep" - e.g. in case of any errors)

Lines 3-7 describe the extraction of an open clausal complement from the "xcomp" grammatical relation. If an open clausal complement belonging to the action is found "create action" can be applied recursively and all aforementioned Specifiers are also added to it (line 6).

Using the algorithms which were described in this section each sentence was decomposed and analyzed. The different parts of a phrase (Actors, Action, Resources) were extracted, converted into their structured representation, combined and added to the World Model. By relying on the Stanford Dependency representation, we added an abstraction layer and avoided the direct analysis of the syntax tree. This helped to avoid problems originating from different word order (Issue 1.2).

3.4. Text Level Analysis

This section describes the second part of our transformation procedure — the text level analysis. It analyzes the sentences taking their relationships into account. The structural overview of this phase is shown in figure 12. While we make use of the Stanford Parser

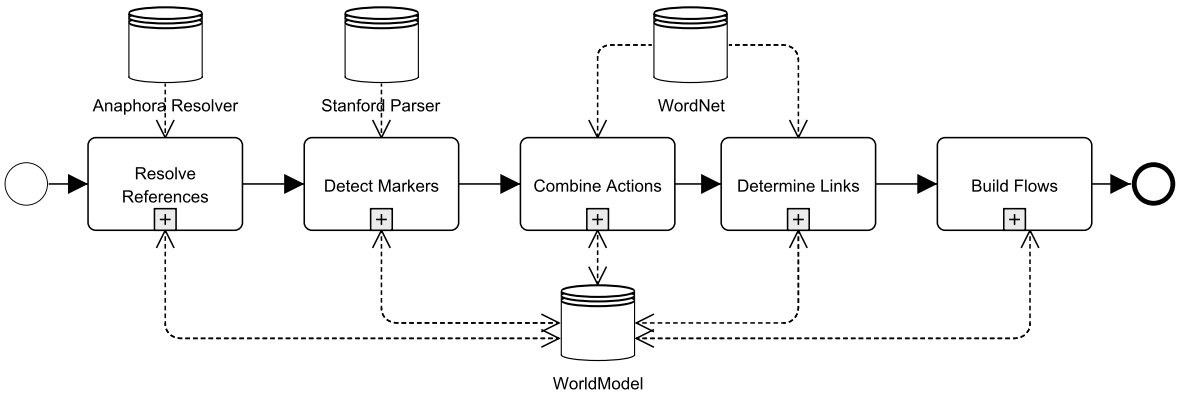

Figure 12: Structural overview of the steps of the Text Level Analysis.

and WordNet again, this stage also utilizes an anaphora resolution algorithm. During each of the five steps, which are described in more detail in the following sections, the Actions previously added to the World Model are augmented with additional information.

3.4.1. Anaphora Resolution Technique

An important part of the algorithm presented here is the determination heuristic for resolving relative references within the text (Issue 4.1). As described in section 2.2.2 existing libraries are not seamlessly integrateable into our system and the output provided by the Stanford Parser. Furthermore, a full resolution of coreference chains as it is provided by other systems is not necessary. Additionally, we wanted to provide an opportunity to the users to manually adjust the reference resolution whenever they think that is necessary. Therefore, we decided to implement a simple anaphora resolution technique into our prototype for the resolution of determiner and pronouns. This procedure is described in algorithm 11. The anaphora resolution is triggered whenever an object is marked with "needsResolve" as it was determined in algorithm 7 (line 2). Afterward, a globally available object "ManualResolutionMap", which stores the results of the manual resolution performed by a user, is checked. If the user manually performed or adjusted

the resolution, the reference is set to the object the user picked (line 4). Otherwise, our determination procedure is triggered. If the object is a determiner, like “the”, “this” or “that”, a corresponding action is searched (line 7). This technique simply determines the action which contains the ExtractedObject “object” and searches for its predecessor either within the same sentence or the previous sentence. In line 9 the resolution procedure for pronouns is triggered. First, the type of animacy is determined. We distinguished three types: “ANIMATE”, “INANIMATE” or “BOTH”. If the ExtractedObject we are dealing with is of type Resource or if it is an Actor which was marked as “Unreal”, the AnimateType “INANIMATE” is returned. If the ExtractedObject is a 3rd person pronoun (“it”,“they”,“them”) it could either refer to a group of people or to a department, organization or software system, which we all defined as valid “Real”-Actors. Therefore, we return the AnimateType “BOTH” which means that no restrictions regarding the animacy of the target reference can be applied. For all other cases we assume the AnimateType “ANIMATE”. This information is needed within algorithm 12 to compile a list of potential candidates for the resolution and to assign a score to them.

Algorithm 12 is the core of the algorithm “findReferenceObject” which we omitted. Within “findReferenceObject” we determine the ID of the sentence which contains the object to resolve. Furthermore, we look up the action which contains the object and determine whether it was created from a copula sentence and save that value as a boolean variable “copAction”. Based on the animacy of the object to resolve, a list of potential candidates is collected in lines 4-12. While all Actors and Resources we collected so far have to be considered if the animacy type BOTH was chosen, “unreal”-Actors and Resources can be filtered in the case of ANIMATE and “real”-Actors in the case of an INANIMATE object to resolve (line 6). The following lines 13-29 assign a score to every candidate. First, if a candidate object itself was resolved we consider the the resolution result (line 15 and 16). Afterward, the distance between the sentences of the object to resolve and the candidate is added using a negative value - the sentence distance penalty

Algorithm 11 Anaphora Resolution

```
1: for ∀ ExtractedObject object ∈ {WorldModel.getActors() ∪ WorldModel.getResources()} do
2:   if object.needsResolve() = true then
3:     if object ∈ ManualResolutionMap then
4:       object.getReference() ← manualResolutionMap.get(object)
5:     else
6:       if object.getName() ∈ Determiners then
7:         object.getReference() ← findActionBefore(object)
8:       else
9:         AnimateType type ← determineAnimateType(object)
10:        object.getReference() ← findReferenceObject(object,type)
11:      end if
12:    end if
13:  end if
14: end for
```

(lines 18 and 19). Therefore, the score of a candidate object will decrease with increasing distance to its anaphora and closer occurrences will be preferred. The following lines utilize the "copAction" variable. This is necessary as we distinguish the resolution of copula and regular sentences. This is motivated by the syntactic patterns which are predominant within our test data set. There copula sentence were mostly used to describe conditions, e.g. as in these sentence:

- "Then, an assessment is performed. If it is positive, [...]".
- "[...] tries to acquire the customer. If he is interested, [...]"

While "an assessment" and "the customer" are both the passive element in the first sentence, they take a subjective and active role in the following if-clause, which is expressed using a copula sentence. Therefore, the preference for the same grammatical status is inverted in case of a copula sentence. For example in line 21 a "role match score" is added to the overall score when both object share the same role within a sentence (either

Algorithm 12 Get Resolution Candidate

Require: **int** sentenceID, **ExtractedObject** objectToResolve, **AnimateType** type, **boolean** copAction

```
1: if sentenceID ¡ 0 then
2:     return ϵ
3: end if
4: Set<ExtractedObject> candidates ← WordlModel.getActors(sentenceID)
5: for ∀ Actor actor ∈ candidates do
6:     if (actor.isMetaActor()) ∨
       (type = ANIMATE ∧ actor.isUnreal()) ∨ (type = INANIMATE ∧ !actor.isUnreal()) then
7:         candidates \ actor
8:     end if
9: end for
10: if type ≠ ANIMATE then
11:     candidates ∪ WorldModel.getResources(sentenceID)
12: end if
13: Map<ExtractedObject.Integer> result ← new Map<ExtractedObject.Integer>()
14: for ∀ ExtractedObject object ∈ candidates do
15:     if object.needsResolve() = true then
16:         object ← object.getReference()
17:     end if
18:     Integer score ← objectToResolve.getSentenceID() - sentenceID
19:     score ← score * Sentence_Distance_Penalty
20:     if object.isSubjectRole() = (copAction ⊕ objectToResolve.isSubjectRole()) then
21:         score ← score + Role_Match_Score
22:     end if
23:     if !copAction ∧ object.isSubjectRole() then
24:         score ← score + Subject_Role_Score
25:     else if copAction ∧ !object.isSubjectRole() then
26:         score ← score + Object_Role_Score
27:     end if
28:     result ∪ (object,score)
29: end for
30: return result
```

subject or object), but, in case of a copula sentence, it is only added when both roles are different. Furthermore a bonus for candidates which hold a subject role is assigned in line 24. In contrast to that, an object role bonus is assigned if the candidate has the object role and the object to resolve was found in a copula sentence (line 25 and 26). The algorithm follows this procedure backwards through the text and processes it sentence by sentence. It stops when the first sentence is reached, or when the maximum attainable score within a sentence with respect to the sentence distance penalty is below the the score of the best candidate. Therefore, in most cases it is not necessary to scan the whole text. Finally, the element with the highest score is returned as a reference for algorithm 11 (line 10). The score values we used are shown in table 4.

Sentence_Distance_Penalty	-15
Role_Match_Score	20
Subject_Role_Score	10
Object_Role_Score	10

Table 4: Score values used in algorithm 12.

To evaluate the performance of this simple approach we compared it to two reference resolution toolkits, which were mentioned in section 2.2.2- *BART* and the *Reconcile* framework. We applied all algorithms to all texts from our test data set which contained at least three references. In total 17 texts containing 111 references of interest were analyzed. All Frameworks were applied using their standard configuration. The results of this evaluation can be seen in table 5.

The first column in the results table contains the ID of the text which was analyzed. All referenced texts are listed in the Appendix and can be used for further verification. The second column indicates the number of relative references which were identified in the text. The following columns show how many references were resolved correctly by the three algorithms as absolute and relative figures. The evaluation was performed manually.

Text ID	# of ref.	BART		Reconcile		our approach	
1-1	3	0	0.00%	3	100.00%	3	100.00%
1-3	4	4	100.00%	0	0.00%	4	100.00%
1-4	13	7	53.85%	7	53.85%	11	84.62%
2-1	8	2	25.00%	4	50.00%	5	62.50%
2-2	7	2	28.57%	2	28.57%	5	71.43%
3-3	3	1	33.33%	3	100.00%	2	66.67%
3-6	3	2	66.67%	1	33.33%	1	33.33%
4-1	4	1	25.00%	1	25.00%	1	25.00%
5-4	5	2	40.00%	4	80.00%	5	100.00%
6-1	12	8	66.67%	1	8.33%	2	16.67%
6-3	7	0	0.00%	1	14.29%	4	57.14%
6-4	13	2	15.38%	1	7.69%	12	92.31%
7-1	5	2	40.00%	4	80.00%	4	80.00%
8-2	8	4	50.00%	5	62.50%	3	37.50%
8-3	7	5	71.43%	5	71.43%	0	0.00%
9-1	4	1	25.00%	1	25.00%	3	75.00%
9-6	5	3	60.00%	1	20.00%	5	100.00%
Total	**111**	**46**	**41.44%**	**44**	**39.64%**	**70**	**63.06%**

Table 5: Comparison of the performance of our anaphora resolution technique to BART and Reconcile

Whenever the co-reference chain identified by BART or Reconcile contained the correct resolution result, the reference was marked as correct, even if other entities were contained. Although our approach only achieved an accuracy of 63.06% it still outperformed both co-reference resolution toolkits, which were not able to correctly identify more then 41.44%. However, this does not mean that our approach is superior, as our results are not generalizable. We have to consider the results as very specialized for the purpose of process model generation and our evaluation parameters. A fundamental difference is that both toolkit generate co-reference chains, while our approach only determines a single reference. Therefore, the output BART and Reconcile create is much more sophisticated and more versatile. Additionally, we only evaluated the resolution accuracy for the references we require for our transformation approach, which are personal pronouns and determiners. But, both libraries also resolve, e.g. relative pronouns and similar noun phrases. Furthermore, we introduced a special handling for the resolution of the determiners "this" and "that" which can link to a whole action instead of a noun phrase. This is useful for our purposes, but not in the scope of the two toolkits. However, references of this kind constitute a very minor share in the test data set. Interestingly, even those tools, which are specifically built for anaphora resolution are not able to handle pleonastic usage of "it".

To conclude, we can state that although our resolution algorithm is only applicable for a very special type of anaphora resolution problem, it is better suited for this kind of problem then general anaphora resolution frameworks.

3.4.2. Conditional Marking

The second step in our analysis is the detection of conditional markers. These markers can either be a single word like "if", "then", "meanwhile" or "otherwise", or a short phrase like "in the meantime" or "in parallel". All of these markers have specific characteristics and can be mapped to different BPMN constructions. In order to capture this semantic information we compiled four lists, namely ConditionIndicators (exclusive gate-

way), ParallelIndicators (parallel gateway), ExceptionIndicators (for Error Intermediate Events), and SequenceIndicators (for the continuation of a branch of a gateway). The lists containing all markers we identified during the analysis of our test data set are fully listed in Appendix C for further reference. These lists also do not claim completeness and can be extended by the user, if necessary.

We first check several grammatical relations for single word markers in algorithm 13 and then analyze the Specifiers of an Action in algorithm 14 for special phrases. Furthermore, as outlined in 3.1.1, some of those markers could be implicit. We will try tackle that problem by detecting an implicit "then" and propagating markers through conjunctions in algorithm 15. The detection of a marker, can also influence the desired position of an action. Thus, to correctly reflect the intended rhetoric structure actions containing a conditional are reordered in algortihm 16.

The markers we are looking for can be contained in different grammatical relations. Most importantly, the "mark" relation which references the word introducing an adverbial clause complement (relation "advcl"). If an action was created from that complement, the word introducing it will determine the relation between the main and the adverbial clause. This applies, e.g., for an if-clause like "If the customer is interested, [...]". The relation "mark(interested,if)" provides the important piece of information, that the clause "the customer is interested" was introduced using "if". Therefore, we start by identifying these "mark" relations (line 2) and add the introducing word to the Action in line 5 of algorithm 13.

Some of the indicators as "while", "then" or "otherwise" are recognized as adverbial modifiers by the Stanford Parser. Thus, another relation we have to analyze is "advmod", which is looked up in line 7. Afterward, we also determine the action to which the governor of that grammatical relations belongs to (line 9) and add the appropriate marker (line 10).

The third block within algorithm 13 inspects prepositional clause modifiers ("prepc").

Algorithm 13 Marker Detection

Require: **List**<**AnalyzedSentence**> sentences

```
1: for ∀ AnalyzedSentence sentence ∈ sentences do
2:     List<TypedDependency> markers ← findDependencies("mark",sentence.getDependencies())
3:     for ∀ TypedDependency marker ∈ markers do
4:         Action a ← findAction(marker.governor(),sentence.getAllActions())
5:         a.getMarker() ← marker.dependent().value()
6:     end for
7:     List<TypedDependency> mods ← findDependencies("advmod",sentence.getDependencies())
8:     for ∀ TypedDependency mod ∈ mods do
9:         Action a ← findAction(mod.governor(),sentence.getAllActions())
10:        a.getMarker() ← mod.dependent().value()
11:    end for
12:    List<TypedDependency> prepcs ← findDependencies("prepc",sentence.getDependencies())
13:    for ∀ TypedDependency prepc ∈ prepcs do
14:        Action a ← findAction(marker.dependent(),sentence.getAllActions())
15:        a.getMarker() ← prepc.getRelationName().getSpecific()
16:    end for
17:    List<TypedDependency> complms ← findDependencies("complm",sentence.getDependencies())
18:    for ∀ TypedDependency complmentizer ∈ complms do
19:        Action a ← findAction(marker.governor(),sentence.getAllActions())
20:        a.getMarker() ← complementizer.dependent()
21:        a.isMarkerFromComplementizer() ← "true"
22:    end for
23: end for
```

A prepositional clause modifier is a clause introduced by a preposition, which changes the meaning of the associated element. An example is the sentence:

- "If this is not the case, we leave him alone, **except** if the potential project budget is huge."

Here the preposition "except" introduces the if-clause. As except was classified as an exception marker within the ExceptionIndicator list (see Appendix C) it is important to capture it as the marker of that action. Contrary to the other relations, the key word "except" is not the dependent nor the governor of the relation, but instead a specification of the relation name[21]. Therefore, it has to be extracted differently as shown in line 15.

The last relation we look into are complementizers ("complm"). A complementizer introduces a clausal complement (ccomp), which means it can be either "that" or "whether" [106]. As "whether" was classified as a conditional indicator it is also looked up (line 17) and added as a marker to the corresponding action (line 19 and 20). But, as "whether" plays a different role during the order correction explained in algortihm 16, we also set a flag "isMarkerFromComplementizer" to true, in order to reflect these circumstances appropriately.

Within algorithm 13 we successfully identified markers which consist of a single word. In contrast to that algorithm 14 is used to detect marking phrases like "in the meantime" or "in case of", which can be part of a prepositional phrase or relative clause. To detect them, we scan each prepositional and relative clause modifier ("pp" and "rcmod") of an action (line 3 and 4). If the prepositional phrase or the relative clause starts with an indicator, the marker which represents the meaning appropriately is set. For a conditional indicator (line 5) we set the marker "if" (line 6) and we set a flag which tells us that the marker was found within a specifier and was not directly attached to the action as a single word (line 7). For a sequence indicator (line 8) and for parallel indicators (line 10) we

[21] Due to the usage of the collapsed dependency representation.

Algorithm 14 Detect Compound Indicators

```
Require: List<AnalyzedSentence> sentences
 1: for ∀ AnalyzedSentence sentence ∈ sentences do
 2:   for ∀ Action action ∈ sentence.getAllActions() do
 3:     List<Specifier> specifiers ← action.getSpecifiers("PP" + "RCMOD")
 4:     for ∀ Specifier specifier ∈ specifiers do
 5:       if startsWithAny(specifier,ConditionalIndicators) then
 6:         specifier.getMarker() ← "if"
 7:         specifier.isCompoundMarker() ← "true"
 8:       else if startsWithAny(specifier,SequenceIndicators then
 9:         specifier.getMarker() ← "then"
10:       else if startsWithAny(specifier,ParallelIndicators then
11:         specifier.getMarker() ← "while"
12:       end if
13:     end for
14:   end for
15: end for
```

place the marker "then" (line 9) or "while" (11), respectively.

Next, we identify some implicit markers in algorithm 15. The implicit markers we are able to detect is an implicit "then" following an if-clause (lines 3-10), and all sequence indicators which are attached to Actions which are part of a conjunction (lines 11-16). Again, we apply our procedure to all Actions within all analyzed sentences. First, we define a variable called "nextMarker" (line 3) and then while iterating over all Actions (line 4) assign the value "then" to it whenever an Action with a conditional indicator, which was not found in a compound (see algorithm 14), is encountered (line 8 and 9). Once this variable was set it will be applied to the following Action within that sentence during the next iteration (line 6). The second part of this method is used to propagate sequence indicators through conjunctions. If the action under investigation carries a sequence indicator (line 11), then all Actions which are related to this Action by a conjunction are determined (line 12) and the same marker is added to them (line 14). Of course, this

Algorithm 15 Add Implicit Markers

```
Require: List<AnalyzedSentence> sentences
 1: for ∀ AnalyzedSentence sentence ∈ sentences do
 2:     List<Action> linkedActions ← ϵ
 3:     Marker nextMarker ← ϵ
 4:     for Action a: sentence.getAllActions() do
 5:         if nextMarker ≠ ϵ then
 6:             a.getMarker() ← nextMarker
 7:         end if
 8:         if a.getMarker() ∈ ConditionalIndicators ∧ !a.isCompoundMarker() then
 9:             nextMarker ← "then"
10:         end if
11:         if a.getMarker() ∈ SequenceIndicators then
12:             linkedActions ← determineConjunctElements(sentence,a)
13:             for Action linkedAction ∈ linkedActions do
14:                 linkedAction.getMarker() ← action.getMarker()
15:             end for
16:         end if
17:     end for
18: end for
```

covers only a small amount of implicit markers which are needed (Issue 1.3). A deeper semantic analysis is required to detect implicit conditions as in "For new customers, a patient file is created".

The last step during the marking phase is the reordering of actions according to their semantic structure. This is necessary as we assumed a sequential ordering of the Actions (see section 3.1.5), but it is possible to add an if-clause after the main clause. Therefore, in algorithm 16 we walk through all sentences and all Actions identified within them (line 1 and 2). If an action was marked using "if" and that marker is not from a complementizer, then we determine the action within that sentence that precedes it (line 4) and switch change order (line 5) within the World Model.

Algorithm 16 Correct Order

Require: **List**<**AnalyzedSentence**> sentences, **List**<**WorldModel**> world

```
1: for ∀ AnalyzedSentence sentence ∈ sentences do
2:     for Action a: sentence.getAllActions() do
3:         if a.getMarker() = "if" ∧ a.isMarkerFromComplementizer ≠ "true" then
4:             Action actionBefore = findActionBefore(a)
5:             world.switchPositions(a,actionBefore)
6:             break
7:         end if
8:     end for
9: end for
```

3.4.3. Action combination

This section describes how we can use the information gathered so far to combine the information contained in two different Actions. This procedure tackles the problem of Actions which are split up over several sentences (Issue 2.2) as laid out in section 3.1.2. To consider two Actions as a candidate for a merger, a reference had to be established between them during the anaphora resolution phase, which is explained in section 3.4.1. This reference can either directly point from the Actor or from the Object of this Action (lines 2 or 5). But, for the case that the Object points to another Actor or Resource we also consider the Action which contains it as a possible candidate (line 8). Next, it is checked whether the objects can be merged (line 12). The method "canBeMerged" checks the following four or five characteristics of both Actions, depending on whether the third argument is "true" or "false":

- Negation modifier (in case of "false" as third argument)
- Both Actions have either no or the same marker
- One of the Action is considered "weak" (depending on the stop word list WeakVerbs as defined in Appendix C)

- One of the Actors is ϵ, needs to be resolved or is meta actor
- One of the Objects is ϵ or needs to be resolved

If the actions truly complement each other in all categories they can be merged and form one single action. This procedure is explained in algorithm 18. When both Actions complement each other except for the negation modifier we can still enhance the information content of one action by copying information (line 15), as the initiating Actor, the Object, and/or the copula attribute. An example for such a case are these sentences:

- "Of course, asking the customer whether he is generally interested is also important."
- "If this is not the case, we leave him alone, [...]"

Here the reference is established by the word "this" which was resolved to reference the Action "the customer in interested". Here all conditions except for the negation are fulfilled. Both Actions have the same marker "if", the verb "is" is considered "weak", the Actor of the second sentence "this" needed to be resolved and the first Action does not contain an Object ("interested" is the copula modifier). But, as the negation modifier differs those actions cannot be merged completely. This is desired as both describe different branches of a split. However, one of the branches is incomplete regarding its information content. We can close this gap by copying information, and obtain two Actions:

- If the customer is interested, [...]
- If the customer is not interested [...]

This enables the generation of clearer models which we expect to be understood more easily.

When both Actions are equal regarding all their characteristics described before, a full merge can be conducted. This procedure is described in algorithm 18. Here the first task is to determine a weak Action, which will be removed in the end of the process

Algorithm 17 Combine Actions

Require: **WorldModel** world

```
1: for ∀ Action action ∈ world.getActions() do
2:     if action.getActor().getReference instanceof Action then
3:         referencedAction ← action.getActor().getReference()
4:     else if action.getObject().getReference() ≠ ϵ then
5:         if action.getObject().getReference() instanceof Action then
6:             referencedAction ← action.getObject().getReference()
7:         else
8:             referencedAction ← findActionContaining(action.getObject().getReference())
9:         end if
10:    end if
11:    if referencedAction ≠ ϵ then
12:        if canBeMerged(action,referencedAction,"true" then
13:            mergeActions(action,referencedAction)
14:        else if canBeMerged(action,referencedAction,"false") then
15:            copyInformation(action,referencedAction)
16:        end if
17:    end if
18: end for
```

Algorithm 18 Merge Actions

Require: **Action** referenceAction, **Action** mainAction,**WorldModel** world,

```
1: Action weakAction ← getWeakAction(referenceAction,mainAction)
2: Action strongActions {referenceAction;mainAction}\ weakAction
3: if (strongAction.getActor() = ϵ ∧ weakAction.getActorFrom() ≠ ϵ) ∨
   strongAction.getActor().needsResolve() ∧ !weakAction.getActorFrom().needsResolve() then
4:     strongAction.getActor() ← weakAction.getActorFrom()
5: end if
6: if (strongAction.getObject() = ϵ ∧ weakAction.getObject() ≠ ϵ) ∨
   strongAction.getObject().needsResolve() ∧ !weakAction.getObject().needsResolve() then
7:     strongAction.getObject() ← weakAction.getObject()
8: end if
9: for Specifier sp ∈ weakAction.getSpecifiers() do
10:    strongAction.addSpecifier(sp)
11: end for
12: world.removeAction(weakAction)
```

and a strong Action, which acquires the new information content. The weak Action is determined using WordNet in combination with our stop word list "WeakVerbs". Then, we determine which information can be merged. In line 3 we check whether the Actor of Action which we considered as strong can be replaced by the Actor of the weak Action (line 4). This is the case when the strong Action has no Actor yet or if the Actor of the strong Action has to be resolved and the other does not. The same procedure is applied to the Object of the Action in the lines 6-8. Lastly, all Specifiers of the weak Action are transferred (line 9-11) and the Action which was classified as weak before is removed from the World Model (line 12).

3.4.4. Inter-Action Link determination

In 3.1.4 (Issue 4.2) we defined three types of textual references: forward, backward, and jump references. This section will introduce our approach to identifying those links in the text automatically. In algorithm 19 we start by comparing all actions within our

World Model to another. To do that we iterate over all actions in two nested for-loops (line 2 and 3). In order to reduce complexity the second for-loop in line 3 only visit Actions, whose index is smaller then the index of the first for loop. Hence, the complexity of the algorithm is $O(\frac{1}{2}n^2)$. Using the method "isLinkable", it is then determined whether the selected actions can be linked or not (line 6). Within this method we compare the following characteristics of both Actions:

- Copula Specifier
- Negation Status
- The initiating Actor (ActorFrom)
- The Object
- The open clausal complement (xcomp) by calling the method recursively
- Prepositional Specifiers, whose head word is "to" or "about"

The elements are compared using their root form, which is determined by the internal word stemming of WordNet. If the elements differ or an element is defined for one Action, but not for the other, the function will return "false". Otherwise, the action are considered equal and a link relationship can be established. This link relationship is saved to the action appearing later in the text (action1) in line 7. This way we can ensure that a link relation is always directed backwards concerning the position of the Actions in the text. Additionally, the type of the link relationship is determined and saved along with the link (line 8). How the type of the link relationship is determined is explained in algorithm 20.

3.4.5. Flow Generation

The last step of the text level analysis is the generation of Flows. As explained in section 3.2 a flow describes how activities are interaction with each other. Therefore, during the process model generation such Flows can be translated to BPMN connecting objects.

Algorithm 19 Determine Inter-Action Links

Require: **List**<**WorldModel**> world

```
List<Action> actions ← world.getActions()
for int i=|actions|-1; i≥0; i++ do
    for int j=i-1; j¿=0; j– do
        Action action1 = actions.get(i)
        Action action2 = actions.get(j)
        if isLinkable(action1,action2 then
            action1.setLink(action2)
            action1.setLinkType(determineLinkType(action1,action2))
        end if
    end for
end for
```

Algorithm 20 Determine Link-Type

Require: **Action** linkSource,**Action** linkTarget

```
if linkSource.getMarker() ∈ ConditionIndicators then
    if WordNet.containsForwardLinkSignal(linkSource.getSpecifiers()) then
        return ActionLinkType.FORWARD
    end if
    DCR linkResult = determineConjunctElements(linkTarget)
    if linkResult.getType() = OR ∨ linkTarget.getMarker ∈ ConditionIndicators then
        return ActionLinkType.JUMP
    end if
else
    if WordNet.containsLoopLinkSignal(linkSource then
        return ActionLinkType.LOOP
    end if
end if
return ActionLinkType.NONE
```

When creating the Flows we build upon the assumption that a process is described sequentially (see section 3.1.5) and upon the information gathered in the previous steps. Therefore, we save the current position within the process in 2 lists: “lastActions” and “openSplit” (line 1 and 2). They determine the position into which the following elements can be inserted, like a cursor in a word processing program. Likewise, their position can be changed, e.g., by the textual jumps, which we identified in section 3.4.4. Thus we start by iterating over all sentences and their extracted Actions sequentially (line 3) and whenever a jump link is encountered, we clear the two lists and add the Action which was linked before to the lastActions list. This behavior is embedded in the method clearLists in line 4. Furthermore, we define the current Action as “transient” which means it is not necessary to directly represent it with a Task node in our process model and continue with the next Action. Otherwise, we search the current sentence for conjoined elements. These are all Actions which are connected to Action “a” with a conjunction. This connection could have been established by a conjunction of verbs, actors or objects, which in turn is represented by a status code within the ConjoinedResult created in line 10.

The word, which connected the items, captured in the ConjunctionType attribute, is important to determine how to proceed and what type of gateway we have to create. So far we support a distinction between “or”, “and/or” (line 14), and “and” (line 18). Other conjunctions, e.g., “but” will not be represented graphically in the process model. If the Action under investigation is not part of a conjunction relationship, the method “handleSingleAction”, which is explained in further detail in algorithm 22 is called in line 12. When the word used to conjoin the to Actions was “or” or “and/or” a gateway has to be created (line 14). Therefore, we first create a so called Dummy Node. This is a node which serves as a placeholder for the source or target of a join. This is necessary as, according to our definition of a Flow in the WorldModel, the gateway is an attribute of the flow and is not explicitly represented. Furthermore, as a Flow is only able to create a one-to-many connection, constructions like the ones shown in figure 13 would not be

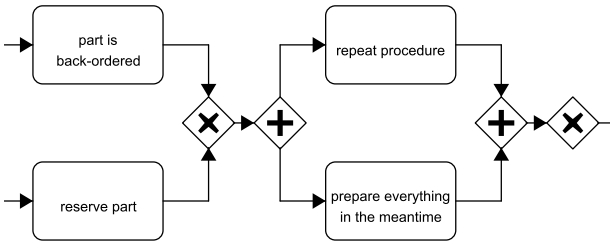

Figure 13: Excerpt of an example model where two Gateways are directly following each other.

possible otherwise. The problem is solved by placing a temporary Dummy Node as the target of a join and from there creating the new split. The Dummy Node is then removed during the process model generation procedure.

Afterward, the appropriate gateway is constructed. In case of an "OR" conjunction an exclusive gateway is constructed and in case of an "and/or" an inclusive one. The next block in the lines 19-26 deals with the "and" connective. From the understanding of natural language an "and" does not necessarily imply a parallel gateway. Therefore, it will only be constructed when the "and" is associated with two different actors (line 22), e.g. as in the example sentence following below. Otherwise, the standard procedure handleSingleAction is called for each item individually (line 25). The previous call to "buildJoin" (line 24) before, ensures that all Actions contained in "lastActions" are joined on a appropriate gateway prior to continuing. Of course, this procedure is also used within "createGateway" (lines 17 and 22).

Additionally, it is possible to have a mixed situation, where several conjunctions between objects are created, as in this example sentence:

- "The MPON and the MPOO perform the equipment acquisition and/or the device change."

In this case the type will be classified as "MIXED" and appropriately handles in line 29. Within "handleMixedSituation" the different connection types are analyzed and multiple

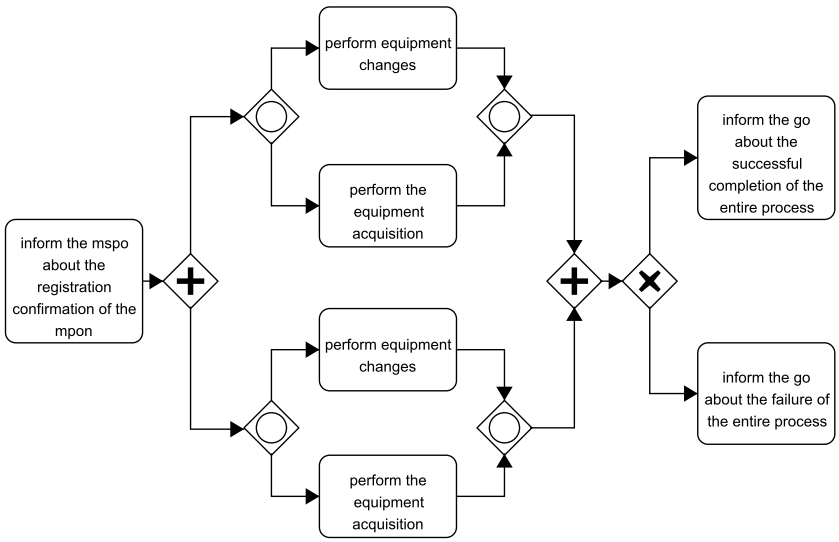

Figure 14: Process model resulting from a “Mixed Situation”.

gateways are created. For the case of our example sentence, the created diagram is depicted in figure 14.

Algorithm 22 describes one part of the flow creation process on a deeper level. Whenever the current Action is not marked with a Conditional Indicator or a Sequence Indicator we close the currently opened split and thereby end a block which consisted of several paths. This attempts to tackle the problems outlined in section 3.1.4. When closing the open split, all Actions which are contained in the openSplit list are joined with a appropriate gateway (line 5) and the list is cleared (line 2). Afterward, the actual handling of the action begin (line 7). We check whether the Action is marked with a parallel (line 7) or conditional (line 10) indicator. If so, we retrieve the last last flow which was added to the WorldModel and add the new Action in a parallel fashion (lines 8 and 9). For conditional indicators the procedure is similar, but as more than one additional condition can be added we obtain the last split which was created. If no split was created before, the

Algorithm 21 Build Flows

Require: **List**<**AnalyzedSentence**> sentences, **List**<**WorldModel**> world

```
List<Action> lastActions ← new List<Action>
List<Action> openSplit ← new List<Action>
for ∀ Action a ∈ ∀ AnalyzedSentence sentence ∈ sentences do
    if a.getLink() = ActionLinkType.JUMP then
        clearLists(a.getLink())
        a.getTransient() ← true
        continue
    end if
    Flow flow ← new Flow(sentence)
    ConjoinedResult ← determineConjoinedElements(a,sentence.getConjunctions())
    if ConjoinedResult.size() = 1 then
        handleSingleAction(flow,a)
    else
        if ConjoinedResult.getType() ∈ {ConjunctionType.OR;ConjunctionType.ANDOR} then
            createDummyNode(lastActions,flow)
            buildGateway(lastActions,openSplit,ConjoinedResult)
        else if ConjoinedResult.getType() = ConjunctionType.AND then
            if ConjoinedResults.getStatusCode() = ACTOR then
                createDummyNode(lastActions,flow)
                flow.getType() ← ConjoinedResult.getType()
                buildGateway(lastActions,openSPlit,ConjoinedResult)
            else
                buildJoin(flow,lastActions,a)
                handleSingleAction(flow,a)
            end if
        else
            createDummyNode(lastActions,flow)
            handleMixedSituation(flow,ConjoinedResult)
        end if
    end if
end for
world.addFlow(flow)
```

Algorithm 22 Handle Single Action

Require: **Flow** flow, **Action** action, **List<Action>** lastActions, **List<Action>** openSplit

```
1: if action.getMaker ∈ SequenceIndicators ∨ action.getMarker ∈ ConditionalIndicators then
2:     closeOpenSplit()
3: end if
4: if lastActions.size >1 then
5:     buildJoin(new DummyAction())
6: end if
7: if action.getMarker() ∈ ParallelIndicators then
8:     getLastFlowAdded().getMultipleObjects() ∪ action
9:     getLastFlowAdded().getType ← FlowType.CONCURRENCY
10: else if action.getMarker() ∈ ConditionalIndicators then
11:     openSplit.clear()
12:     openSplit ∪ getLastSplit().getMultipleObjects()
13:     getLastSplit().getMultipleObjects() ∪ action
14:     getLastSplit().getType() ← FlowType.CHOICE
15: else
16:     standardSequence(flow,action)
17: end if
```

method “getLastSplit” will simply return the last Flow which was added. Next, we add the Action to the list of multiple objects and remember that all former elements in the openSplit list (lines 11-14). For the case that no marker was determined for the action, a simple sequence is created (line 16).

The concrete details of building splits and joins and the handling of a mixed situation process the actions and flows in a similar fashion and are, thus, not further detailed. The Flows which we created in this section are stored in the World Model and will be used to construct Sequence Flows and Gateways for the process model in the next section

3.5. Process Model Generation

In the last phase of our approach the information contained in the World Model is transformed into its BPMN representation. We follow a nine step procedure, as depicted

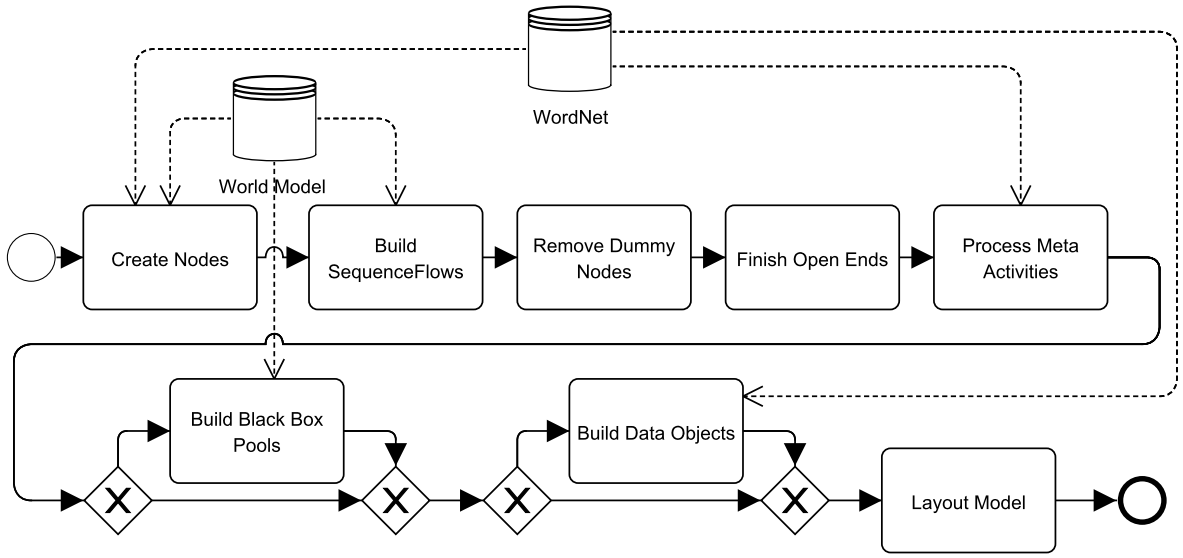

Figure 15: Structural overview of the steps of the Process Model Generation phase.

in figure 15. The first 4 steps: creation of nodes, buidling of Sequence Flows, removal of dummy elements, the finishing of open ends, and the processing of meta activities are used to create an initial and complete model. Additionally the model can be augmented by creating Black Box Pools and Data Objects. These two steps are dependent on the preferences of the user and can be disabled in our prototype. Lastly the model is layouted to achieve a human-readable representation.

3.5.1. Model Creation

The first step required for the model creation is the construction of all nodes of the model. The overall procedure is shown in algorithm 23. We iterate over all Actions contained within the WorldModel (line 1). Whenever we encounter an action which is marked using a conditions, an event nodes is created (line 3). Otherwise, we can create a standard BPMN Task node (line 5). When creating an event, we look up the contained words within WordNet and create an Message or Timer Intermediate event if applicable. This is, e.g., the case when the event text contains a word which can be

related to the concept of a “time_period” in WordNet. This behavior can be controlled by adding/removing concepts in our stop word lists (see Appendix C). When creating a label for the nodes we respect the concepts of good labeling style which were mentioned in section 2.1.2. Thus, we try to create correct verb-object labels. The phrases of Specifiers associated to a Specified Element are added to the label when they were tagged as either ”CORE” or ”GENETIVE” (see section 3.3.3). As it is usually not possible to classify all phrases we also add Specifiers classified as ”UNKNOWN” when their headword is ”to”, ”in”, or ”about” for Actions and ”into”, ”under”, or ”about” for objects. This behavior can be controlled by the user again as he can disable to addition of unknown specifiers to the label of a node. Additionally we included a technique to transform nominalizations. For example consider the sentence:

- “The last phase is the creation of a quotation.”

Using WordNet we can determine that “creation” was derived from the verb “create”. Additionally, the current verb “is” will be classified as “weak” according to the principles explained in section 3.4.3. Therefore, the action text can be transformed to “create a quotation”.

Afterward the Flow Object was generated, we create a Lane Element representing the Actor initiating the Action (line 8). If the Action has no assigned Actor or the verb describing the Action is considered “weak”, ϵ will be assigned to the Lane. In the following parts of the code we then utilize two global variables called “lastLane” and a list of nodes “notAssigned”. If no Lane was determined for an Action, it is added to the last Lane which was created successfully as we assume that the process is described in a sequential manner. Therefore, even if not mentioned explicitly again, we expect the Action to be performed by the same Actor. The list “notAssigned” is employed when no Lane was determined successfully so far (line 17). We then store all nodes in that list and add them to the first Lane which is created later (line 14). It is also possible that no Lane is found within an entire text. In this case no Lane and no surrounding Pool can be created for

the Process Model.

The second step required during the model creation is the construction of all edges. In our case only Sequence Flows have to be created as all objects were placed in a single Pool). Due to the definition of Flows within our World Model, this transformation is straight forward. Whenever a Flow which is not of the type "Sequence" is encountered, a Gateway appropriate for the type of the flow is created. An exception to that is the type "Exception". If the World Model contains a flow of this type, an exception intermediate event is attached to the task which serves as a source and this Intermediate Event is connected to the target instead of the node itself.

As mentioned in section 3.4.5 we applied the concept of "Dummy Actions", which were inserted between gateways directly following each other. As after the construction of all Sequence Flows such Dummy Actions are not required anymore, they are removed from the process model, while keeping their predecessor and successor connected.

Step four is concerned with open ends. So far, no Start and End Events were created. This is accomplished in this step. The procedure is also straight forward. We create a preceding Start event to all Tasks which do not have any predecessors (in-flow = 0) and succeeding End Events to all Tasks which do not have any successors (out-flow = 0). Additionally, Gateways whose in- and out-flow is one receive an additional branch ending in an End Event.

The last step in the model creation phase tries to deal with Meta-Activities. Therefore, we try to remove redundant nodes in front directly adjacent to Start or End Events. This is required as several texts contain sentences like those:

- "[...] the process flow at the customer also ends."
- "[...] If the request is not finished within 30 days, then the process is stopped."
- "The process of "winning" a new customer ends here."
- "The intake workflow starts with [...]"

Algorithm 23 Create Nodes

Require: **WorldModel** world

```
1: for ∀ Action a ∈ world.getActions() do
2:     if a.getMarker() ∈ Conditional Indicators ∧ !a.isCompoundMarker() then
3:         FlowObject fow ← createEvent(a)
4:     else
5:         FlowObject fow ← createTask(a)
6:     end if
7:     if !WordNet.isWeakAction(a) then
8:         Lane lane = getLane(a.getActor())
9:     end if
10:    if lane ≠ ϵ then
11:        lane.addNode(fow)
12:        Lane lastLane = lane
13:        for ∀ Node n ∈ notAssigned do
14:            lastLane.addNode(n)
15:        end for
16:    else
17:        if lastLane ≠ ϵ then
18:            lastLane.addNode(fow)
19:        else
20:            notAssigned.add(fow)
21:        end if
22:    end if
23: end for
```

If such sentences were not filtered so far we might find Tasks label “process ends” right in front of an end event or “start workflow” following a start event. This information is redundant, as the start/end of the process is already expressed through the Start/End Event. Therefore, we remove nodes whose verb is synonymous or contained in the hypernym tree of “end” or “start” according to WordNet if they are adjacent to a Start or End Event, respectively.

After the execution of these five steps a full BPMN model was created which can already be sued by the user. Additionally, the model can be augmented adding further elements as Black Box Pools and Data Objects, which will be described in the next section.

3.5.2. Model Augmentation

The last three steps of our transformation procedure are the creation of Black Box Pools and Data Objects and the layouting of the model. Just as the labeling of the nodes, the creation of Black Box Pools and Data Objects can be influenced by the user. The procedure we follow is described in algorithm 24 and algorithm 25. First we check whether an Action can be classified as an verb of interaction. This is accomplished by checking the WordNet hypernym tree of the main verb of the action for the occurrence of an “interactionVerb” (see Appendix C) like “inform”, “send”, “ask”, or “report”. Afterward, we have to determine the corresponding source or target of that communication activity. This can either be the direct object (line 3), one of the Specifiers of the Action containing a sender (line 7) or containing a Receiver (line 11). To identify a Sender in the Specifiers of an Action, we check the Frame Element which was assigned during the element creation in section 3.3.3. The presence of the Frame Elements “Addressee” or “Recipient” indicate a receiver of our message while “Donor” or “Source” indicate a the sender. Additionally, If this sender or receiver object is an Actor it can be represented as a Pool. These checks are performed for main verb of the Action, but also for its open clausal complement (xcomp) if it exists.

The actual creation of the Black Box Pool is shown in algorithm 25. If the element

Algorithm 24 Create Black Box Pools

Require: **WorldModel** world

```
1: for ∀ Action a ∈ world.getActions() do
2:     if WordNetWrapper.isInteractionVerb(a) then
3:         ExtractedObject object ← a.getObject()
4:         if object ≠ ϵ ∧ object instanceof Actor ∧ !isMetaActor(object) then
5:             buildBlackBoxPool(a,object)
6:         else
7:             Specifier ← getContainedSender(a.getSpecifiers())
8:             if sender ≠ ϵ ∧ sender instanceof Actor ∧ !isMetaActor(sender) then
9:                 buildBlackBoxPool(a,sender)
10:            end if
11:            Specifier receiver ← getContainedReceiver(a.getSpecifiers())
12:            if receiver ≠ ϵ ∧ receiver instanceof Actor ∧ !isMetaActor(receiver) then
13:                buildBlackBoxPool(a,receiver)
14:            end if
15:        end if
16:    end if
17: end for
```

Algorithm 25 Build Black Box Pool

Require: **Action** action **SpecifiedElement** element

```
1: if model.getPools().containsByName(getName(element)) then
2:     communicationLinks.put(a,getName(element))
3: else if Options.getBuildBlackBoxPools() then
4:     BlackBoxPool BBPool ← new BlackBoxPool(getName(element))
5:     model.getNodes() ∪ BBPool
6:     Task taskNode ← getTaskNode(action)
7:     taskNode.getSubType() ← WordNet.getVerbType(action)
8:     model.getEdes() ∪ new MessageFlow(tasNode,BBPool)
9: end if
```

which represents the source or target of the communication was already created as a Pool during the model creation phase, we cannot create a Black Box Pool, as BPMN does not provide any means to express this communication. Nevertheless, we store the identified relation in a map called "communicationLinks" (line 2) of which we can make use in the Lane Split-off Procedure, which is described next. If the element is not already represented in the model as a Pool, a Black Box Pool can be created. Of course, only if the user selected the appropriate option (line 3). The Black Box Pool is created with the name derived from the given Specified Element (line 4). It is then added to the process model (line 5) and connected to the task corresponding to the action (line 8). Additionally, this task will be assigned an appropriate BPMN sub-type (Receive or Send), which depends on the nature of the main verb of the action (line 7).

The creation of Data Objects is only performed when the user selected the option "Create Data Objects". The creation itself analyses the existing BPMN model. For each task within the model a set of data object candidates is determined based on the underlying Action, as explained in algorithm 26. For each pair of adjacent nodes those sets are then compared and if both contain the same element a data object which passes from the first action to the second one is created. If a task does not have any data objects candidates with its neighbors in common, data objects which are only associated to this

Algorithm 26 Get Data Object Candidates

Require: SpecifiedElement object

```
 1: Set<String> result ← new Set<String>
 2: if object instanceof Action then
 3:     Action action ← (Action)object
 4:     result ∪ getDataObjectCandidates(action.getActor())
 5:     result ∪ getDataObjectCandidates(action.getObject())
 6:     result ∪ getDataObjectCandidates(action.getXComp())
 7: else if object instanceof Resource ∨ (object instanceof Actor ∧ ((Actor)object).isUnreal()) then
 8:     if object.needsResolve() then
 9:         object ← object.getReference()
10:     end if
11:     if WordNet.canBeDataObject(object) then
12:         result ∪ getName(object)
13:     end if
14: end if
15: for ∀ Specifier specifier ∈ object.getSpecifiers() do
16:     result ∪ getDataObjectCandidates(specifier.getObject())
17: end for
18: return result
```

node are created. The Determination of data object candidates is explained in algorithm 26. It first created an empty set for storing the result (line 1). The type of the incoming element is then checked (line 2). If it is an Action the method will be called recursively by obtaining all data object candidates from the Actor, Extracted Object, and XComp-Action if they are available. If the incoming element is a resource or an "unreal"-Actor (line 7) it is checked using WordNet (line 11). This check examines the hypernym tree of the main noun of that object. If this hypernym tree contains an element of a list called *Data Object Determiners* (see Appendix C), the check was successful and the object will be considered as a possible data object. Furthermore, all Specifiers of the incoming element are checked as well independently of its type (line 15-17).

As the elements of the generated model do not contain any position information yet, our generation procedure concludes with an automated layout algorithm. The layouting of BPMN diagrams presents an interesting research problem in itself. We based our algorithm on a simple grid layout approach [56]. To improve the layouting results we enhanced it with standard layout graph layouting algorithms as Sugiyama's layout algorithm [114] or principles of the topology-shape-metric approach [30], which were successfully applied to UML class diagrams [29]. A detailed description of the implementation is omitted as it is not the main point of the research presented in this thesis.

3.6. Lane Split-off Procedure

According to the BPMN 2.0 specification [87, page 114] a "Pool represents a Participant in a Collaboration or a Choreography" and a "Participant can be a specific PartnerEntity (e.g., a company) or can be a more general PartnerRole (e.g., a buyer, seller, or manufacturer)". Therefore, different companies or general PartnerRoles should be depicted by seperate Pools. But, so far all actors that we encountered were represented by a Lane within a single Pool. We did this as it is not easily inferable whether an actor represents a separate company or PartnerRole or if he/she is just a part of a modeled company. Furthermore, a modeler can decide to explicitly model different entities with different pools, although they represent parts of the same company (see for example B.78 or B.80). This discrepancy is known and partial empirical support was provided in [103] that users might have difficulties to decide which of the two modeling constructs to use. Therefore, we decide to generate each actor as a Lane initially and provide the user with a refactoring mechanism which splits off a lane and converts it into a separate Pool. This post-processing mechanism is explained in detail in this subsection and some examples will be provided.

3.6.1. SequenceFlow Transformation

The first step of the Lane split-off mechanism is the creation of a new Pool and the identification of the edges of the model which have to be transformed (algorithm 27). A

Algorithm 27 Lane Split-off Mechanism

```
Require: ProcessModel model, Lane lane, Map<Node,Lane> communicationLinks
 1: Set containedNodes ← lane.getContainedNodes()
 2: Pool newPool ← new Pool(lane)
 3: model.Nodes ∪ newPool
 4: for ∀ SequenceFlow sqf ∈ model.getEdges() do
 5:    if sqf.getSource() ∈ containedNodes ⊕ sqf.getTarget() ∈ containedNodes then
 6:       transformToMessageFlow(model,sqf)
 7:    end if
 8: end for
 9: if Options.getBuildCommunicationLinks() = true then
10:    for ∀ Node n ∈ communicationLinks.keySet() do
11:       if n ∈ containedNodes then
12:          buildExtraCommunicationLink(model,n,communicationLinks.getValue(n))
13:       end if
14:    end for
15: end if
16: layout(model)
```

new Pool is created in line 2, which takes all properties, e.g. the name, size, and contained nodes, from the Lane the user selected. We then check all Sequence Flows. If it connects an element which is inside of our selected Lane with an element outside of the lane or vice versa, it has to be transformed into a Message Flow (algorithm 28 called in line 6). This check can be performed elegantly by logically connecting the containment checks with an exclusive OR (line 5). Lastly, we call algorithm 29 in line 12, whenever the nodes of the lane the user selected was marked as containing a message interaction and the corresponding options were set. The last step is to apply the automated layout algorithm as explained in section 3.5 to visualize the changes in the process model.

Algorithm 28 is responsible for replacing the former Sequence Flow with an actual Message Flow. In order to transform the Sequence Flow without creating an invalid model, two things have to be considered: first, the usage of Start and End Events and,

second, keeping all nodes of the process within a Pool connected. To accomplish that we determine where the flow will return to our pool by following the successors of the target of the Sequence Flow. A simple breadth-first search is used to find all nodes which return to the lane of the source of the Sequence Flow again (line 1). Similarly, we can determine a predecessor within the same lane by following the predecessors of the target or the Sequence Flow. Afterward, we can decide whether we have to create a Message Start Event or if it is possible to simply set the type of the Task under investigation to either "Send" or "Receive" (line 5 and 12). An explicit Message Event has to be created if:

- the user set the corresponding option to always create events,
- the source of the original Sequence Flow is not a Task (e.g. a Gateway),
- the source of the original Sequence Flow is a Task, but a type was assigned already, or
- no successor/predecessor within the same lane was found, which means the node will become the starting/end point for the process within that pool.

Figure 16 demonstrates the result of the transformation of Sequence Flows. When starting the transformation procedure from the model depicted on the left of figure 16 three Sequence Flows are selected for the transformation (from A to the Gateway, from B to D, and from C to E). For the first Sequence Flow the successors within the same Lane are D and E. For the Gateway no predecessor in the same Lane can be found, therefore $predecessor \leftarrow \epsilon$. As a result A can simply be transformed into a Task of type "Send", while a Message Start Event has to be created for the Gateway. Moreover a new Gateway is created after Task A and is connected to its successors (D and E). If this connection was not established no relation between A, D, and E would be visible when collapsing the upper Pool. By building those new connections the validity of the model can be preserved. For the other to Task the transformation works likewise. For B no successor can be found,

Algorithm 28 Transformation To Message Flow

Depending on the outcome of the evaluation a Start/End Event is created or the corresponding type of the Task is set (lines 6, 9, 13, and 16). To ensure that the all nodes are still connected new Sequence Flows are then added (lines 21-30) either from the newly created Message Event or the Task node to its successors within the same lane. If there is more then one successor, a gateway is build to ensure that all nodes remain connected (lines 23-29).

Require: **ProcessModel** model, **SequenceFlow** sqf

1: Set successors ← findSuccessors(model.getClusterForNode(sqf.getSource()),sqf.getTarget())
2: Node predecessor ← findPredecessor(model.getClusterForNode(sqf.getSource()),sqf.getSource())
3: Node start ← ϵ
4: Node end ← ϵ
5: **if** needToCreateEvent(sqf.getSource(),successors) **then**
6: Node startEvent ← new MessageSendStartEvent(sqf.getSource())
7: start ← startEvent
8: **else**
9: setSendStereotype(sqf.getSource())
10: start ← sqf.getSource()
11: **end if**
12: **if** needToCreateEvent(sqf.getTarget(),predecessor) **then**
13: Node endEvent ← new MessageReceiveEvent(sqf.getTarget())
14: end ← endEvent
15: **else**
16: setReceiveStereotype(sqf.getTarget())
17: end ← sqf.getTarget()
18: **end if**
19: model.getEdges() \ sqf
20: model.getEdges() ∪ new MessageFlow(start,end)
21: **if** |successors| = 1 **then**
22: model.getEdges() ∪ new SequenceFlow(start,successors.get(0))
23: **else if** |successors| ≥ 1 **then**
24: Node gateway ← new Gateway()
25: model.getNodes() ∪ gateway
26: model.getEdges() ∪ new SequenceFlow(start,gateway)
27: **for** ∀ Node n ∈ successors **do**
28: model.getEdges() ∪ new SequenceFlow(gateway,n)
29: **end for**
30: **end if**

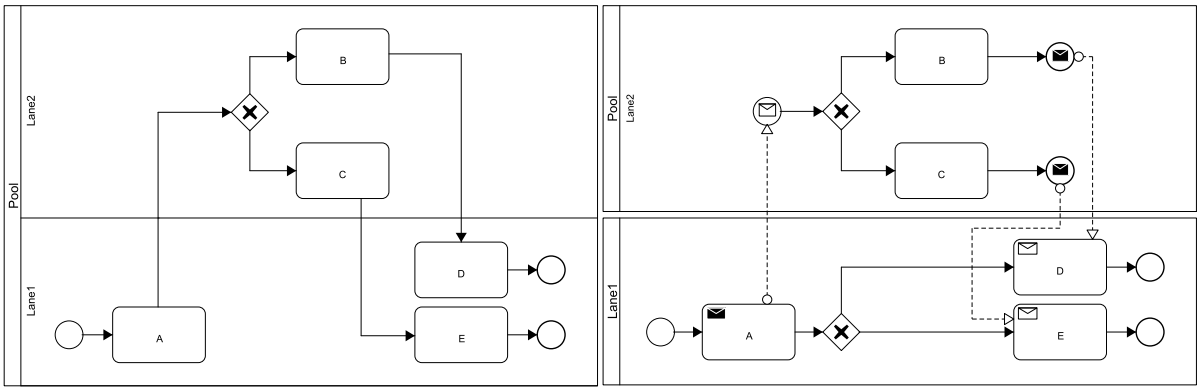

Figure 16: Example model before (left) and after (right) splitting off Lane2 - after the transformation the communication is realized entirely via Message Flows.

but a predecessor of D, namely Task A, is found within Lane 2. Therefore, Task B has to be followed by a Message End Event, while for D it is enough to set the correct type. If the user had selected the option to always create Message Event, the type of the Tasks A, D, and E would be unchanged, but instead they would be followed/preceded by Message Start/End Events.

3.6.2. Building Semantic Communication Links

Splitting off a Lane also gives the opportunity to make communication, which was detected during the semantic analysis (section 3.3.3), explicit. Beforehand, this was not possible as the BPMN specification provides no means to visualize the exchange of information between two lanes, without creating a new Task. The technique which was used to create these extra communications is described in algorithm 29. The input of this algorithm is the node for which communication was detected and the Lane representing the target or source of that communication. First a predecessor of the given Node within the Lane with which it seeks to communicate is searched using the method we already employed in algorithm 28. If this predecessor Node is an End Event, it is refactored into an intermediate event, so we can continue the flow from that position (line 3). The next step assumes that a mapping called *NodeCache* is globally available. It simply maps all

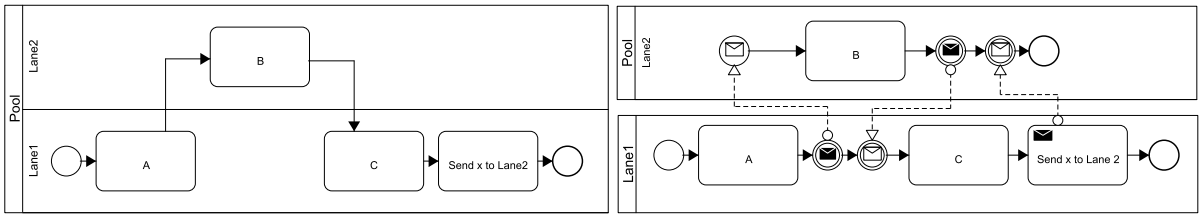

Figure 17: Example model which shows how an extra communication link is created between a Task and a Lane.

newly created Message Intermediate Events to their respective predecessor so it can be reused if this predecessor is to be used again in another iteration within algorithm 27. Otherwise, a new Message Intermediate Event has to be created (lines 8-20) and is placed in between the node which we called "predecessor" and its successors.

The example shown in figure 17 shows the effect of the application of this algorithm. Because a semantic communication relationship was detected between the Task "Send 'x' to Lane2" and the Participant which we named "Lane2", it is possible to alter the model and add an Intermediate Message Receive Event in Lane2, right after its Message End Event. Thereby, we enriched the Process Model by explicitly visualizing the communication relationships between several Pools. Furthermore, the Message Intermediate Event which was created in "Lane2" can be used as a starting point to further specify the behavior of that participant. The drawbacks of the application of this mechanism are that several edges are added to the diagram and readability could suffer. However, as we provided the user with the option to disable the creation of those extra links (see algorithm 27), these drawbacks are not severe.

Algorithm 29 Build Extra Communication Link

Require: **ProcessModel** model, **Node** node, **Lane** lane

```
1: Node predecessor ← findPredecessor(lane,node)
2: if predecessor instanceof EndEvent then
3:     convertToIntermediateEvent(predecessor)
4: end if
5: if predecessor ∈ NodeCache then
6:     Node messageEvent ← NodeCache.getValue(predecessor)
7: else
8:     Node messageEvent ← new IntermediateMessageEvent()
9:     model.getNodes() ∪ messageEvent
10:    model.getEdges() ∪ new SequenceFlow(predecessor,messageEvent)
11:    NodeCache ∪ (predecessor,messageEvent)
12:    successorEdges ← {a:a ∈ model.getEdges(), a.getSource() = predecessor}
13:    for ∀ SequenceFlow sqf ∈ successorEdges do
14:        sqf.setSource(messageEvent)
15:    end for
16:    if |successorEdges| = 0 then
17:        Node endEvent ← new EndEvent()
18:        model.getNodes() ∪ endEvent
19:        model.getEdges() ∪ new SequenceFlow(messageEvent,endEvent)
20:    end if
21: end if
22: model.getEdges() ∪ new MessageFlow(node,messageEvent)
```

4. Evaluation of Generated Process Models

In order to evaluate the quality of the transformation procedure presented in section 3, a test data set was collected. In the next subsections we will describe the composition and contents of this test data set in more detail. Afterward, the methodology we followed in order to compare the models which were generated by our prototype to the manually modeled ones are explained in section 4.2. The results of the application of this methodology are then illustrated in section 4.3 and discussed in section 4.4.

4.1. Test Data Set

An element of this test data set consists of a textual process description and a BPMN process model which was created by a human modeler from the text. In total we collected 47 of those text-model pairs. We did not restrict the collection to a specific domain or type. Thus, different sources were incorporated into our test data set. We classified each element into one of the four categories:

- Academic - elements provided by universities or university employees
- Industry - elements provided by corporations or their employees
- Textbook - elements taken from modeling textbooks
- Public Sector - elements taken from national public sector institutions

The distribution of the test data according to these categories is shown in 18. While academic models account for the largest share, all other categories are also adequately represented.

The academic elements of the test data set are a courtesy of the Humboldt Universität zu Berlin, the Technische Universität Berlin, the Queensland University of Technology, and the Technische Universiteit Eindhoven. The Humboldt Universität and the Queensland University of Technology provided us with four and eight exercises and solutions

Type	Amount	Frequency
Academic	15	31.91%
Industry	9	19.15%
Textbook	9	19.15%
Public Sector	14	29.79%
Total	47	100.00%

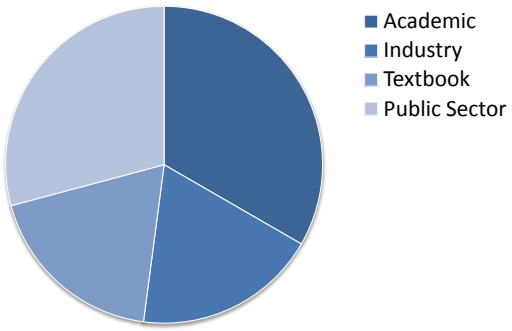

Figure 18: Test-data by source type.

which were used in their BPMN modeling tutorials. The models of the other two institutions were used in research projects and are discussed in [53, 52, 104]. This is also reflected by the complexity of the models as the models which were used within research projects represent the largest ones in the whole data set regarding the number of sentences and words. Thus, in total we were able to collect 15 elements for the test data set from these academic institutions.

The industry models are taken from two main sources. First, we gathered four models from the websites or online help documentation of three BPM tool vendors, namely Active VOS, Oracle, and BizAgi. All provide examples of textual descriptions and a modeling solution implemented in their tool. Another four models, which are usually used for client and employee training, were provided by the inubit AG. Those models were provided in German and translated in order to allow an evaluation. Furthermore, Frank Puhlmann supplied us with an additional model. Thus, a total of 9 models from vendors were collected.

We also consulted two standard textbooks and BPMN modeling, the *BPMN Modeling and Reference Guide* [136] and the *Praxishandbuch BPMN* [36]. While the first one provided five modeling exercises and solutions we found three suitable passages within the later one. These three models were also translated from German to English in order

to allow an evaluation.

Lastly, we found the definition of switch processes of the *Federal Network Agency* of Germany, contained in attachment one of enactment BK6-09-034 / BK7-09-001[22], helpful. Therein the textual descriptions were not free text, but semi-structured in a tabular form and the corresponding model was provided as a UML sequence diagram. In order to use this source, both — the text and the sequence diagrams — were transformed as described in the next section.

Unfortunately, the works mentioned in section 2.3 do not provide test data elements or the process descriptions are not available in English. Therefore, we were not able to compare our results to those of other research groups. However, all models including their textual description, the manually created BPMN diagram, and the one generated by our transformation approach are fully listed in the Appendix. This will enable other researchers to create improved transformation algorithms and to compare their results to ours easily.

The different descriptions and models cover various domains, including e.g. insurance, medical, banking, marketing, sales, and electric power supply. That we are able to transform texts of all those domains highlights the domain-independence of the proposed technique.

In table 6 we listed and compared several characteristics of the texts and models, which comprise our data set. Each line contains the average value taken for all models within that part of the collection.

- An ID which was assigned uniquely for each part of the test data set.
- The number of contained models (M).
- Number of sentences (n).

[22] `http://www.bundesnetzagentur.de/cae/servlet/contentblob/159722/publicationFile/8504/WiM_Anlage_1_Wechselprozesse.pdf`; last accessed 2010-11-05

ID	Source	M	Type	n	∅l	$\|N\|$	$\|G\|$	$\|E\|$
1	HU Berlin	4	academic	10.00	18.14	25.75	6.00	24.50
2	TU Berlin [53, 52]	2	academic	34.00	21.17	71.00	9.50	79.50
3	QUT	8	academic	6.13	18.26	14.88	1.88	16.00
4	TU Eindhoven [104]	1	academic	40.00	18.45	38.00	8.00	37.00
5	Vendor Tutorials	4	industry	9.00	18.20	14.00	2.25	11.50
6	inubit AG	4	industry	11.50	18.38	24.00	4.25	21.25
7	BPM Practicioners	1	industry	7.00	9.71	13.00	1.00	9.00
8	BPMN Prac. Handbook [36]	3	textbook	4.67	17.03	13.00	1.33	14.67
9	BPMN M&R Guide [136]	6	textbook	7.00	20.77	23.83	3.00	23.67
10	FNA - Metrology Processes	14	public sector	6.43	13.95	24.43	3.14	25.93
	Total	**47**		**9.19**	**17.16**	**23.21**	**3.38**	**23.64**

Table 6: Characteristics of the test data set by source (average values for all contained models).

- Average length of the sentences (∅l).
- Size of the models regarding nodes ($|N|$).
- The number of Gateways within the model ($|G|$).
- Size of the models regarding edges ($|E|$).

While most of the models are short with 6 to 11 sentences, the models provided by the TU Berlin and TU Eindhoven are comparably large with 34 and 40 sentences, respectively. The average length of the sentences ranges from 9 to 21 words. Therefore, the sentences within our test data set are a bit shorter compared to the sentence lengths of Wall Street Journal articles, which can be found, e.g., in[10], or other scientific corpora [131] where the average sentence length ranges from 22 to 28 words. This shows that our test data is not overly structured, but conforms to a natural writing style of humans.

According to [79] the size of a model directly influences the probability of it containing errors and models which are larger than 50 elements should be divided. Only the models of set two, of the TU Berlin, violate this recommendation and can be considered large.

The following sections will illustrate how these test data elements were prepared, models were automatically generated and compared to the given models created by humans.

4.2. Evaluation Methodology

In order to avoid unintended effects while parsing, minor corrections were applied to the texts. Additionally, some of them still had to be translated as explained in section 4.2.1. Accordingly, some of the models within our test data set had to be converted from other process modeling languages to BPMN in order to compare them. This is illustrated in section 4.2.2. The last section describes the metrics which we employed to compare the generated and manually created models to each other.

4.2.1. Text Preparation

To avoid any form of experimenter bias only minor corrections were performed on the test data. For example, uncommon punctuation like "–" were replaced with real parenthesis as otherwise the *Stanford Parser* failed to correctly classify these elements. We also observed that commas were replaced by hyphens "[...] to re-prioritize these measures - otherwise the process continues". We replaced these hyphens with commas, which also improved the accuracy of the parser. Additionally, run-on sentences and comma splices were corrected.

Six of the texts we collected were written in German. In order to evaluate them, we had to translate them to English. Therefore, an initial machine translation using Google Translate[23] was applied and only grammatical mistakes were corrected.

An exceptional position is taken by the model of the federal network agency of Germany, as the textual description were presented in a semi-structured textual form and

[23] `http://translate.google.com`; last accessed 2010-10-28

the corresponding models were UML sequence diagrams [88]. Nevertheless, we wanted to include these models as they represent an instance of an economically used process specification. Furthermore, an extension of the transformation procedure presented in this thesis to process semi-structured texts is possible. It could then represent another possibility to enter data into the World Model as mentioned in section 3.1.5. We did not implement this functionality into our prototype as it is not within the scope of this thesis. So, in order to apply our transformation approach, free texts were created. This was achieved by combining the elements of three columns "sender", "receiver", and "message" given in the original document. The message was always represented as a gerund or nominalization. In order to create a grammatically correct sentence this nominalization was turned into a verb. For example the values: "GO", "MPON", and "Rejection of Application" were transformed to "The GO rejects the application of the MPON". Sometimes alternative paths were described by labeling a row using lower case characters. In this case the two sentences were simply joined using "or". Another complication was that in an additional remarks column jumps or flow conditions were given, e.g. "Continue with step 5". If we encountered such a condition we added the current sentence headed by an if to the mentioned step. Thus, taking the examples we would add "If the GO rejects the application of the MPON, [..]" in front of the sentence which is generated out of step 5.

4.2.2. Model Preparation

As most of the models were available as pictures only, we had to remodel them in order to be able to apply an automatic comparison procedure. Some of the models used implicit splits and joins. While remodeling these splits and joins were replaced with explicit gateways. This does not change the semantics of a model, but enables a fair comparison to the generated models, as we did not consider the possibility of implicit splits and joins in our transformation procedure. Within our test data set, some models were not available as BPMN diagrams. Specifically, the model provided by the TU Eindhoven was modeled as a workflow net and the 14 models of the Federal Network Agency were supplied as UML sequence diagrams.

We asked a final year master student to convert the workflow net without adding or removing any information. A one-to-one transformation was possible with the help of the accompanying text. To transform the UML sequence diagrams to BPMN diagrams, we manually applied a simple transformation technique. First, a pool is created for each lifeline of the sequence diagram. If a lifeline only receives message and never acts as a sender, a Black Box Pool is created. Otherwise, we specify the behavior of the participant with an open horizontal Pool. For each message within the diagram, a task is created with the name of the message as its caption. If the receiver of the message is different from its sender, the task is turned into a send activity and a message flow between both participants is created in the BPMN diagram. When the receiver of a message is the sender of a later message, the appropriate Message- End/Start or Intermediate Events are created. To ensure the structural validity of the generated model, the same message event building constraints which were already applied during the lane-split-off technique described in 3.6 were applied. As not all decisions were explicitly modeled in the sequence diagrams, the textual information had to be consulted. Whenever, the sequence flow was not strictly sequential a column containing remarks contained a phrase like “continue with step x”. Thus, the message or sequence flow was redirected accordingly for each of those remarks. Additionally, to ensure a lossless transformation of the sequence diagrams, an Intermediate Message Receive Event is added for each received message even if the reaction of the receiver of that message was not specified any further. This simple transformation seemed appropriate given the quality of the provided material.

According to the UML 2.3 superstructure specification, a sequence diagram either only describes a single trace or has to use a *CombinedFragment* [88, section 14.3.3,p. 483] to depict alternative paths. Therefore the provided models are not completely formally correct. Nevertheless, the results of the transformation are satisfying. All transformations were performed manually and both the original sequence diagram and the BPMN diagram are included in Appendix B.10 for verification.

4.2.3. Evaluation Metrics

Following the text and model metrics mentioned in section 4.1, we measured the size of the generated model with respect to the number of nodes (N), the number of gateways (G), the number of edges (E), and the relative in- or decrease compared to the manually created model. Additionally, we analyzed the text for:

- Number of sentences (n)
- Average length of the sentences (∅l)
- Number of sentences describing the process on a meta level (m)
- Number of identified relative references (r)
- Number of textual links links and jumps (j)
- Size of the models regarding nodes (N)
- Size of the models regarding edges (E)
- Similarity of the models (sim)

To measure the similarity (sim) of the manually and automatically created models we employ the metric of *Graph Edit Distance*. To compute the Graph Edit Distance, the graph representation of the process models is analyzed. The labels, attributes, the structural context, and behavior (see [128]) are compared [27]. Afterward a greedy graph matching heuristic ([26]) is employed to create pairs of nodes and edges. We decided to use the greedy heuristic as it was shown to have the highest performance without considerable accuracy trade-offs. This procedure has much in common with existing research on graph similarity [12, 17, 43, 127]. After the mapping is created a Graph Edit Distance value can be calculated given:

- N_i - set of nodes in model i

- E_i - set of edges of model i
- $\overline{N_i}$ - the set of nodes in model i which were not mapped
- $\overline{E_i}$ - the set of edges in model i which were not mapped
- M - The mapping between the nodes of model 1 and 2 with a similarity (sim) assigned to each pair

An overall indicator for the remaining difference between the models can be calculated as:

$$m^* = \begin{cases} \sum_{i=1}^{|M|} 1 - sim(M_i) & \text{if} |M| > 0 \\ 1.0 & \text{otherwise} \end{cases} \quad (1)$$

As a last step weights for the importance of the differences (w_{map}), the unmapped Nodes (w_{uN}), and the unmapped Edges (w_{uE}) have to be defined. For our experiments we gave the difference a slightly higher importance and assigned $w_{\text{map}} = 0.4$ and $w_{\text{uN}} = w_{\text{uE}} = 0.3$. The overall graph edit distance then becomes:

$$\text{sim}(m_1, m_2) = 1 - (w_{\text{map}} * \frac{m^*}{|M|} + w_{\text{uN}} * \frac{\overline{|N_1|} + \overline{|N_2|}}{|N_1| + |N_2|} + w_{\text{uE}} * \frac{\overline{|E_1|} + \overline{|E_2|}}{|E_1| + |E_2|}) \quad (2)$$

This value lies between 0 and 1. For the case that all nodes could be mapped with a similarity of 1.0 the terms will also become 1.0. If the mapping is not optimal, because the similarity of the nodes is less then 1.0 or if some nods or edges could not be mapped the term in parenthesis will grow steadily and the similarity decreases. If no nodes were mapped at all or we compare something to an empty model the similarity will be 0. For convenience the overall similarity will be presented as a percentage.

Most of the works dealing with the automatic creation of model from text do not conduct an evaluation of the quality of their approach at all or have a different focus. In [117] the standard information retrieval metrics *precision* and *recall* are employed in order evaluate the analysis engine. The experimenters marked an action as correctly identified themselves, where it is unclear whether the generated models are compared to manually

ID	m	r	j	$\|N_{gen}\|$	$\Delta\|N_{gen}\|$	$\|G_{gen}\|$	$\Delta\|G_{gen}\|$	$\|E_{gen}\|$	$\Delta\|E_{gen}\|$	sim
1	3	5,25	0	30,25	14,88%	5,50	-9,09%	28,75	14,78%	77,94%
2	7,50	7,50	2,50	91,50	22,40%	13,00	26,92%	94,00	15,43%	70,79%
3	0,50	1,38	0,00	20,25	26,54%	2,63	28,57%	20,13	20,50%	78,78%
4	8,00	4,00	1,00	63,00	39,68%	1,00	-700,00%	52,00	28,85%	41,54%
5	1,25	1,75	1,75	24,75	43,43%	4,25	47,06%	23,00	50,00%	63,63%
6	2,25	8,00	0,50	29,75	19,33%	2,75	-54,55%	25,25	15,84%	60,93%
7	0,00	5,00	0,00	14,00	7,14%	2,00	50,00%	11,00	18,18%	74,35%
8	0,00	5,00	0,33	13,33	2,50%	1,00	-33,33%	10,33	-41,94%	77,49%
9	0,83	1,50	0,33	22,33	-6,72%	3,33	10,00%	20,83	-13,60%	71,77%
10	0,00	0,21	0,36	25,29	3,39%	3,71	15,38%	27,29	4,97%	89,81%
Total	**1,23**	**2,60**	**0,49**	**27,43**	**15,36%**	**3,72**	**9,14%**	**26,77**	**11,69%**	**76,98%**

Table 7: Result of the application of the evaluation metrics to the test data set.

created ones or if the experimenter judged the importance of an action. This procedure is highly threatened by internal validity [109] and hard to reproduce, especially as the texts and models were not provided. Moreover, the focus of the study was different as use-cases which are very structured were analyzed and the goal was not to create an overarching process model, but several sub processes which are combined in a def-use graph [118]. In contrast to that, the evaluation of the similarity is based on an automatic mapping, which is not influenced by the experimenter and uses an independent human judgment as a benchmark. Furthermore, the similarity of two nodes is factored into the our metric. Thus even when all nodes were successfully mapped, the similarity is not necessarily 100% as, e.g., differences in the labels or flows are taken into account. A simple precision/recall metric would produce 100% in such a case. Therefore, we consider the similarity metric as more appropriate.

4.3. Test Results

After having defined the metrics we want to employ they are applied to each element in the test data set. The options regarding the creation of Black Box Pools or data objects were set appropriately to the desired outcome, which was given by the manually created process models. After the models were generated, the lane split-off procedure as described in section 3.6 was applied to those lanes which are also shown as individual pools in the manually created models. Finally the similarity metric as defined in section 4.2.3 was used to compare the generated model to the desired output. The results of this application to the models of each source are shown in table 7. Columns 2-4 show that the concepts of meta sentences, relative references, and textual jumps are important for almost all elements within our test data. The following six columns show the average values of nodes, gateways, and edges within the generated models. We can see that the transformation procedure tends to produce models which are in average 9-15% larger in size then what a human would create. On the one hand, this behavior can be explained by noise and meta sentences within the text which were not filtered appropriately (see Appendix B). On the other hand, humans tend to abstract during the process of modeling. Therefore, we can often find details which are included in the text also in the generated model. The results are highly encouraging as in average our approach is able to recreate 76% of the model. On a model level up to 96% of similarity can be reached, which means that only minor corrections by a human modeler are required, presuming that the manually created models are the desired outcome of the modeling process. The interested reader will find a full list containing all metrics for each individual text and model in Appendix A.

Additionally, to determine whether there are dependencies between the textual features and the similarity of the generated models, a linear regression was conducted [28]. The results are shown in table 8. This table shows that all analyzed features have a negative impact on the generation quality. Thus an increase of the number of sentences etc.

Feature	β_0	β_1	R^2
Number of meta sentences	0,8821	-0,0318	0,5428
Number of relative references	0,7793	-0,0131	0,0926
Number of textual links	0,8295	-0,0223	0,2664
Number of Sentences	0,8898	-0,0316	0,5350
Number of avg. Sentence length	0,8317	-0,0227	0,2753

Table 8: Results of a linear regression on 5 textual features regarding similarity.

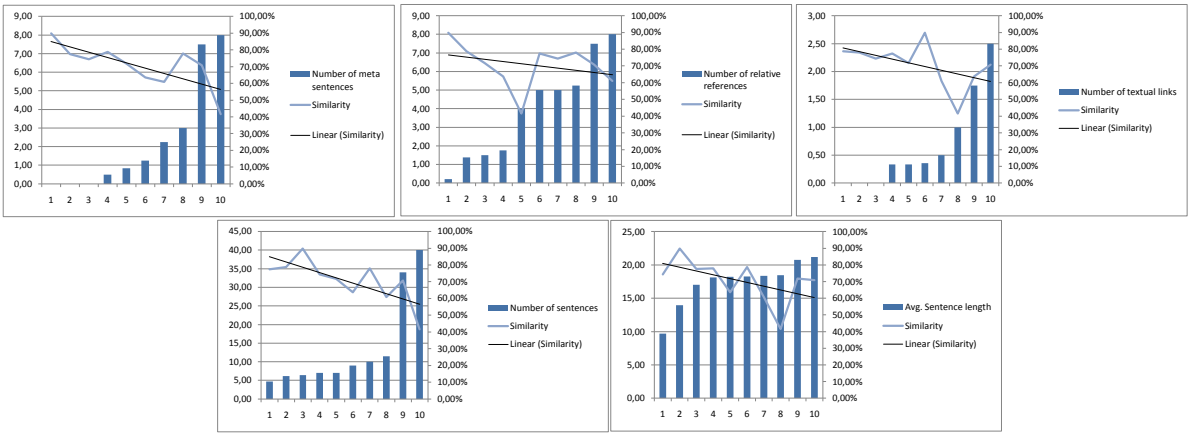

Figure 19: Graphical representation of the conducted linear regressions.

reduces the similarity. Most interestingly the length of the text and the number of meta sentences have the strongest impact on the similarity. However, due to the composition of the test data set, the number of meta sentences is heavily dependent on the length of the text with a β_1 of 0.88. Therefore, we can conclude that the existence of meta sentences and noise in general is the most important factor to consider for the transformation approach.

4.4. Discussion

Our evaluation has shown that in average a similarity of 76.98% was achieved. These results are highly encouraging as it means that a modeler has to revise about 23.02% of a model to obtain the desired representation. The process of model creation is, therefore, simplified to model correction. During the detailed analysis we determined different sources of failure, resulting in a low similarity for certain models. These are noise, different levels of abstractions, and processing problems within our system. With noise we mean sentences or phrases which are not part of a process description, but rather detail the used data objects or add information which lie outside of the process. Some examples are:

- "This object consists of data elements such as the customers name and address and the assigned power gauge."
- "An intaker keeps this registration with him at times when visiting the patient and in his close proximity when he is at the office."
- "Service Management deals on a first level with violations of quality in services that are provided to customers."
- "Thus, in addition to handling the sale of a company's issue, the underwriters in effect give their seal of approval to it."

While this information can be important for the understanding of a process description or in the context of a requirements specification, they lead to unwanted Activities within the generated model, reducing the similarity. To tackle this problem further analyses and filtering mechanisms, possibly using probabilistic measures are required.

Low similarity sources were also achieved whenever the manually created model described the process on a different level of granularity. As our transformation procedure sticks to the structure and usage of words within the text the generated model will be as

detailed as the text itself. To solve this problem, we could integrate automated abstraction procedures [95, 107] into our prototype.

Lastly, the employed natural language processing components failed during the analysis. The Stanford Parser, e.g., failed at correctly classifying verbs. We depict two instances of these failure in figure 20. In the example to the left the parser classified "the second

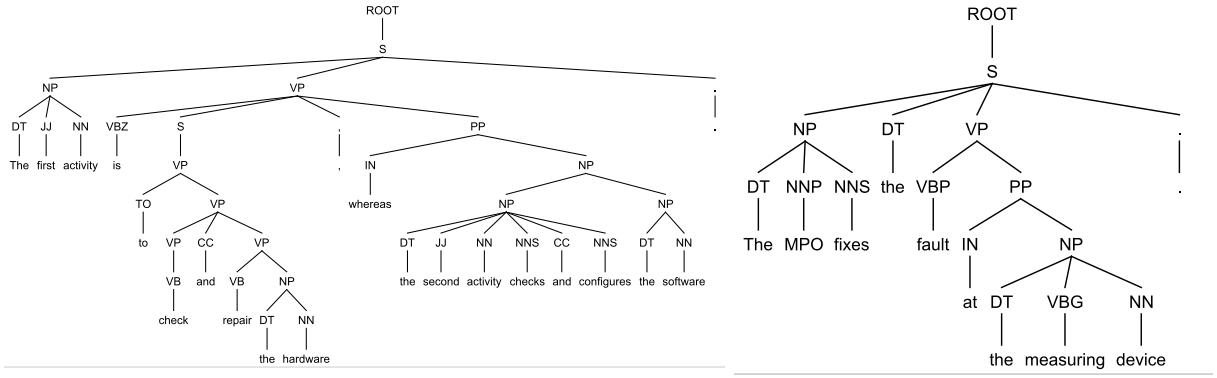

Figure 20: Examples of failed syntax parses.

activity checks and configures" as a noun phrase. Due to that the verbs "check" and "configure" cannot be extracted into Actions during our transformation and this information gets lost. In the second example the actual verb "fix" was classified as a noun and the "fault" was identified as verb. This leads to the creation of an activity "fail a measuring device" conducted by the Actor "the MPO fixes"[24]. While such failures are expected as the parser is based on a probabilistic model, we could improve the transformation success either by, e.g., combining the input of several parser or improving the parsing accuracy, or by training the parser on annotated data specialized on process model descriptions. But also the other components caused mistakes in the transformation. Some important verb in the area of business process management were simply not contained in FrameNet, as "report". Therefore, no message flow was created between report activities and a Black Box Pool, as no recipient was identified. We expect this problem to be solved in the future

[24]MPO - Measuring Point Operator.

as the annotation process proceeds and the FrameNet database grows. With WordNet, the opposite problem was observed. WordNet contains many very specialized sentences which are uncommon in process descriptions. An example is the word “house” in the sense of an aristocratic family line (“the House of York”), which correctly leads to “social group”. Therefore, in the sentence “The customer wants to buy a house”, house is identified as an Actor and not as a Resource, which leads to problems during the anaphora resolution stage. Similar problems were observed with times like “2:00 pm”, where pm as an abbreviation for “Prime Minister” is classified as an Actor. To solve this problem a reliable sense disambiguation has to be conducted. Nevertheless, overall good results were achieved by using WordNet as a general purpose Ontology.

5. Conclusion

In this thesis we investigated possibilities to automatically transform textual process specifications into a formal process model. To achieve this goal we combined research in the areas of NLP and BPM. Through the automated generation of process models from textual description, we are able to increase the efficiency of business analysts and leverage saving potentials. Furthermore, we enable domain experts to create formal process models without the efforts of learning a modeling language. We believe that our results will contribute to this stream of research. The contributions of this thesis, therefore, can be summarized as:

- We collected and analyzed recent literature on the generation of process model from text.
- Thereof we extracted and categorized the dominant issues.
- We provided a solution strategy and transformation approach for the automated generation of business process model from unrestricted natural language text.
- We demonstrated the capabilities of our procedure with our research prototype.
- We gathered a test data set containing 47 model-text-pairs, which cover various domains and styles of writing.
- And, we introduced a novel evaluation metric for the evaluation of such transformation systems and applied it to our approach.

Throughout our analysis we highlighted the important issues for the construction of a system capable of processing textual process specifications and identified the main areas of improvement. Our evaluation shows that on average we are able to automatically generate models which reach a similarity score of 76% compared to those created by humans. Thus, we are confident that our approach simplifies the process of model creation and simultaneously the effort required by a business analyst can be significantly reduced.

5.1. Limitations

Despite our encouraging results an evaluation solely based on an automatic similarity metric is insufficient to prove its value. In order to evaluate the usefulness of our procedure, an empirical user study has to be conducted. This is also the case as the similarity metric based on the graph edit distance does not necessarily reflect the opinion of a human modeler. While the structural and label similarity of our generated model is considered high by our system and user might find little value in it. On the other hand, it is also possible that, despite a low similarity, the generated model could have a higher utility than those provided by a human modeler. As shown during our evaluation the transformation approach tends to create more detailed models with a higher number of nodes and edges. Exactly, these further details could be valuable for a user in order to understand the modeled process. Thus, it is not clear in how far the similarity metric compares to the understanding of a human, emphasizing the need for an empirical user study.

Another issue is that our test data set comprising 47 text-model is relatively small according to the recommendations given in, e.g., [16]. Therefore, our test results are not fully generalizable.

So far the system we constructed is able to read process descriptions consisting of full sentences. Furthermore, we assumed the description to be sequential and to contain no questions and little process-irrelevant information. Another prerequisite is that the text is grammatically correct and constituent. Thus, the parsing of structured input, like tables or texts making use of indentions, or texts which are of low quality is not possible at the moment and presents opportunities for further research.

5.2. Further Research

While the evaluation conducted in this thesis evinced encouraging results different lines of research could be pursued in order to enhance the quality or scope of our process model generation procedure. As shown the occurrence of meta-sentences or noise in general is one of the severest problems affecting the generation results. Therefore, we could

improve the quality of our results by adding further rules and heuristics to identify such noise. Alternatively, the training of a statistical classifier as it was tried in [68] could be worthwhile. Another major source of failures was the syntax parser we employed. Hence, assessing the performance of other parser regarding our test data set or the training of a statistical parser model specific for process descriptions might be valuable. An alternative to traditional syntax parsers are semantic parser as the one presented in [115] which could be investigated. Some errors were also attributed to the misclassification of Actors and Objects using WordNet. The problem thereby is that WordNet covers a broad range of language and, as we have not applied a disambiguation technique, had to check all senses of a word. Thus, taking the textual context into account and applying a disambiguation technique [137, 122], e.g. based on the similarity of concepts within WordNet [113], presents an opportunity for improvement. Optionally, it is possible to exchange WordNet as lexical component with a domain ontology. As shown in [71] it is not even necessary to create this domain specific ontology manually, as, given a sufficiently large text corpus, the important concepts and relations can be extracted automatically.

Instead of entirely relying on the given textual information, it is also imaginable to complement the system with a domain expert. Missing information could then be requested from the user in an interactive fashion an further relieve the business analyst from interviewing tasks. However, this would require substantial semantic analysis and reasoning capabilities. At the same time adding the possibility to analyze and process larger amounts of textual information, e.g. from a CMS, and extending the generation capabilities to include organizational charts and data models as outlined in figure 9, is important in a practical context. This in turn will require methods from the area of text mining and information retrieval to identify important passages and to perform a thematic clustering of the acquired information [64].

Another line of research is creating a textual description out of a text document. As shown in [117] this can be used to create a round-tripping mechanism. Thus an analyst

could either modify the textual process description or the BPMN model itself and both are kept consistent.

Lastly, transferring the system to other languages is an important task. An integration should be easily possible as little language specific components were utilized. The Stanford Parser is as of today already trained on different corpora for German [100], Chinese [67], and Arabic [44]. The SUMO projects which combines and links "WordNets" of different languages (Chinese, Hindi, Tagalog, Czech, German, Italian, Korean, Romanian, Arabic) [85] was created and a link to FrameNet has also been established [111]. An example is the GermanNet Project which recreated the WordNet lexical database for German [47]. Unfortunately, the Stanford Dependency representation is currently only available for English and Chinese [13]. As the parsing of Chinese poses different challenges, e.g. as words are usually not separated using whitespace, it was not within the scope of this thesis, but presents an interested direction for further research.

References

[1] Camille Ben Achour. Guiding scenario authoring. In *8th European-Japanese Conference on Information Modelling and Knowledge Bases*, pages 152–171. IOS Press, 1998.

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. Addison-Wesley Reading, MA, 1999.

[3] C.F. Baker, C.J. Fillmore, and J.B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, volume 1, pages 86–90. Association for Computational Linguistics, 1998.

[4] J. Becker, M. Kugeler, and M. Rosemann. *Process management: a guide for the design of business processes*. Springer Verlag, 2003.

[5] J. Becker, M. Rosemann, and C. Von Uthmann. Guidelines of business process modeling. *Business Process Management*, 1806:241–262, 2000.

[6] R. Blumberg and S. Atre. The problem with unstructured data. *DM Review*, 13:42–49, 2003.

[7] A. Bosca, F. Corno, G. Valetto, and R. Maglione. On-the-fly construction of web services compositions from natural language requests. *Journal of Software*, 1(1):40, 2009.

[8] Richard A. Brealey, Stewart C. Myers, and Franklin Allen. *Principles of Corporate Finance*. McGraw-Hill, 2007.

[9] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on applied natural language processing*, pages 152–155. Association for Computational Linguistics, 1992.

[10] E. Brill. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the workshop on Human Language Technology*, pages 237–242. Association for Computational Linguistics, 1993.

[11] T. Briscoe, J. Carroll, and R. Watson. The second release of the RASP system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80. Association for Computational Linguistics, 2006.

[12] Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255 – 259, 1998.

[13] P.C. Chang, H. Tseng, D. Jurafsky, and C.D. Manning. Discriminative reordering with Chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59. Association for Computational Linguistics, 2009.

[14] E. Charniak. Statistical techniques for natural language parsing. *AI magazine*, 18(4):33, 1997.

[15] E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Morgan Kaufmann Publishers Inc., 2000.

[16] A.L. Comrey and H.B. Lee. *A first course in factor analysis*. Lawrence Erlbaum, 1992.

[17] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1367–1372, 2004.

[18] M.A. Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102. Citeseer, 2001.

[19] M. Cremene, J.Y. Tigli, S. Lavirotte, F.C. Pop, M. Riveill, and G. Rey. Service Composition based on Natural Language Requests. In *2009 IEEE International Conference on Services Computing*, pages 486–489. IEEE, 2009.

[20] R. Davis and E. Brab
"ander. *ARIS design platform: getting started with BPM*. Springer-Verlag New York Inc, 2007.

[21] de AR Gonçalves, J.C. and Santoro, F.M. and Baião, F.A. Business Process Mining from Group Stories. In *Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design*, pages 161–166, 2009.

[22] de AR Gonçalves, J.C. and Santoro, F.M. and Baião, F.A. A case study on designing processes based on collaborative and mining approaches. In *International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Shanghai, China, 2010.

[23] M.C. De Marneffe, B. MacCartney, and C.D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC 2006*. Citeseer, 2006.

[24] M.C. de Marneffe and C.D. Manning. Stanford typed dependencies manual. Technical report, Stanford University, 2008.

[25] M.C. de Marneffe and C.D. Manning. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

[26] R. Dijkman, M. Dumas, L. Garcıa-Banuelos, and R. Käärik. Graph Matching Algorithms for Business Process Model Similarity Search. In *Proceedings of the 7th International Conference on Business Process Management (BPM 2009)*, page 48. Springer-Verlag New York Inc, 2009.

[27] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36:498–516, 2010.

[28] N.R. Draper, H. Smith, and E. Pownell. *Applied regression analysis*, volume 706. Wiley New York, 1998.

[29] H. Eichelberger and J.W. von Gudenberg. UML class diagrams - State of the art in layout techniques. In *nternational Workshop on Visualizing Software for Understanding and Analysis*, pages 30–34. Citeseer, 2003.

[30] M. Eiglsperger, M. Kaufmann, and M. Siebenhaller. A topology-shape-metrics approach for the automatic layout of UML class diagrams. In *Proceedings of the 2003 ACM symposium on Software visualization*, page 189. ACM, 2003.

[31] C.J. Fillmore. The case for case, 1967.

[32] C.J. Fillmore. Frame semantics. *Cognitive linguistics: basic readings*, 34:373–400, 2006.

[33] G. Fliedl, C. Kop, and H.C. Mayr. From textual scenarios to a conceptual schema. *Data & Knowledge Engineering*, 55(1):20–37, 2005.

[34] G. Fliedl, C. Kop, H.C. Mayr, A. Salbrechter, J. Vöhringer, G. Weber, and C. Winkler. Deriving static and dynamic concepts from software requirements using sophisticated tagging. *Data & Knowledge Engineering*, 61(3):433–448, 2007.

[35] PJM Frederiks and T.P. Van der Weide. Information modeling: the process and the required competencies of its participants. *Data & Knowledge Engineering*, 58(1):4–20, 2006.

[36] Jakob Freund, Bernd Rcker, and Thomas Henninger. *Praxishandbuch BPMN*. Hanser, 2010.

[37] F. Friedrich. Measuring Semantic Label Quality Using WordNet. *Nüttgens M, Rump FJ, Mendling J, Gehrke N (Hrsg)*, 8:7–21, 2009.

[38] N. Ge, J. Hale, and E. Charniak. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170, 1998.

[39] Aditya Ghose, George Koliadis, and Arthur Chueng. Rapid business process discovery (R-BPD). In *ER'07: Proceedings of the 26th international conference on Conceptual modeling*, pages 391–406, Berlin, Heidelberg, 2007. Springer-Verlag.

[40] AK Ghose, G. Koliadis, and A. Chueng. Process Discovery from Model and Text Artefacts. In *2007 IEEE Congress on Services*, pages 167–174. IEEE Computer Society, 2007.

[41] D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[42] Ana-Maria Giuglea and Alessandro Moschitti. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of the 21st International Conference on Computational Linguistics*, ACL-44, pages 929–936, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[43] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1100–1111, 2005.

[44] S. Green and C.D. Manning. Better Arabic Parsing: Baselines, Evaluations, and Analysis.

[45] J.L. Gross and J. Yellen. *Handbook of graph theory*. CRC, 2004.

[46] T.R. Gruber. Automated knowledge acquisition for strategic knowledge. *Machine Learning*, 4(3):293–336, 1989.

[47] B. Hamp and H. Feldweg. Germanet-a lexical-semantic net for german. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, 1997.

[48] HM Harmain and R. Gaizauskas. CM-Builder: an automated NL-based CASE tool. In *The Fifteenth IEEE International Conference on Automated Software Engineering*, pages 45–53. IEEE, 2000.

[49] J. Herbst and D. Karagiannis. An inductive approach to the acquisition and adaptation of workflow models. In *Proceedings of the IJCAI*, volume 99, pages 52–57. Citeseer, 1999.

[50] A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *Mis Quarterly*, 28(1):75–105, 2004.

[51] J.R. Hobbs. Resolving pronoun references. *Lingua*, 44(4):311–338, 1978.

[52] Oliver Holschke. *Granularitt als kognitiver Faktor in der adaptiven Wiederverwendung von Geschftsprozessmodellen*. PhD thesis, Technische Universitt Berlin, 2010.

[53] Oliver Holschke. Impact of granularity on adjustment behavior in adaptive reuse of business process models. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management*, volume 6336 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2010.

[54] J.E. Ingvaldsen, J.A. Gulla, X. Su, and H. Rønneberg. A text mining approach to integrating business process models and governing documents. In *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, pages 473–484. Springer, 2005.

[55] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, second edition, February 2008.

[56] I. Kitzmann, C. Konig, D. Lubke, and L. Singer. A Simple Algorithm for Automatic Layout of BPMN Processes. In *IEEE Conference on Commerce and Enterprise Computing (CEC09)*, pages 391–398. IEEE, 2009.

[57] D. Klein and C.D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[58] Dan Klein and Christopher D. Manning. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS)*, pages 3–10. MIT Press, 2003.

[59] C. Kop and H.C. Mayr. Conceptual predesign–bridging the gap between requirements and conceptual design. In *Third Internation Conference on Requirements Engineering*, page 90, 1998.

[60] C. Kop, J. Vöhringer, M. Hölbling, T. Horn, C. Irrasch, and H.C. Mayr. Tool Supported Extraction of Behavior Models. In *Proc. 4th Int. Conf. on Information Systems Technology and its Applications ISTA2005*, 2005.

[61] J. Krogstie, G. Sindre, and H. Jørgensen. Process models representing knowledge for action: a revised quality framework. *European Journal of Information Systems*, 15(1):91–102, 2006.

[62] P. Kumanan, A. Paradkar, A. Sinha, and S.M. Sutton Jr. Automated Inspection of Industrial Use Case Models Inferred from Textual Descriptions, 2010.

[63] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Internatioal Conference on Machine Learning*, pages 282–289. Citeseer, 2001.

[64] S. Lamprier, T. Amghar, B. Levrat, and F. Saubion. Using an Evolving Thematic Clustering in a Text Segmentation Process. *Journal of Universal Computer Science*, 14(2):178–192, 2008.

[65] H. Leopold, S. Smirnov, and J. Mendling. On Labeling Quality in Business Process Models. *Nüttgens M et al.(Hrsg)*, 8:42–57, 2009.

[66] H. Leopold, S. Smirnov, and J. Mendling. Refactoring of Process Model Activity Labels. In *15th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 268 – 283, 2010.

[67] R. Levy and C. Manning. Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 439–446. Association for Computational Linguistics, 2003.

[68] J. Li, H.J. Wang, Z. Zhang, and J.L. Zhao. A policy-based process mining framework: mining business policy texts for discovering process models. *Information Systems and E-Business Management*, 8(2):169–188, 2010.

[69] K. Li, RG Dewar, and RJ Pooley. Object-oriented analysis using natural language processing. In *Proceedings of International Conference for Young Computer Scientists (ICYCS2005)*. Citeseer, 2005.

[70] Y. Li, P. Musilek, M. Reformat, and L. Wyard-Scott. Identification of pleonastic it using the web. *Journal of Artificial Intelligence Research*, 34(1):339–389, 2009.

[71] Dekang Lin and Patrick Pantel. Concept discovery from text. In *Proceedings of the*

19th international conference on Computational linguistics, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[72] Maya Lincoln, Mati Golani, and Avigdor Gal. Machine-assisted design of business process models using descriptor space analysis. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management*, volume 6336 of *Lecture Notes in Computer Science*, pages 128–144. Springer, 2010.

[73] O.I. Lindland, G. Sindre, and A. Sølvberg. Understanding quality in conceptual modeling. *IEEE software*, 11:42–49, 1994.

[74] D. Lübke. Transformation of Use Cases to EPC Models. *EPK 2006*, 5:137, 2006.

[75] D. Lübke, K. Schneider, and M. Weidlich. Visualizing use case sets as bpmn processes. In *Requirements Engineering Visualization, 2008. REV'08.*, pages 21–25. IEEE, 2009.

[76] M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):330, 1993.

[77] I.A. Melčuk. *Dependency syntax: theory and practice*. State University of New York Press, 1988.

[78] J. Mendling, H.A. Reijers, and J. Recker. Activity labeling in process modeling: empirical insights and recommendations. *Information Systems*, 35(4):467–482, 2010. Defines the best labeling style (Verb-Object vs Action-Noun vs. Rest).

[79] J. Mendling, HA Reijers, and WMP van der Aalst. Seven process modeling guidelines (7pmg). *Information and Software Technology*, 52(2):127–136, 2010.

[80] J. Mendling, BF Van Dongen, and WMP van der Aalst. Getting rid of the or-join in business process models. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, page 3. IEEE, 2007.

[81] Jan Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer Publishing Company, 2008.

[82] G.A. Miller and F. Hristea. WordNet nouns: Classes and instances. *Computational Linguistics*, 32(1):1–3, 2006.

[83] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[84] M. Muehlen and J. Recker. How much language is enough? Theoretical and practical use of the business process modeling notation. In *Advanced Information Systems Engineering*, pages 465–479. Springer, 2008.

[85] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9. ACM, 2001.

[86] A.G. Nysetvold and J. Krogstie. Assessing business process modeling languages using a generic quality framework. *Advanced topics in database research*, 5:79–93, 2006.

[87] Object Management Group. Business process model and notation (bpmn) version 2.0, June 2010.

[88] Object Management Group. OMG Unified Modeling Language Superstructure Specification, Version 2.3, 05 2010.

[89] S.P. Overmyer, B. Lavoie, and O. Rambow. Conceptual modeling through linguistic analysis using LIDA. In *icse*, page 0401. Published by the IEEE Computer Society, 2001.

[90] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[91] H.G. Perez-Gonzalez and J.K. Kalita. GOOAL: a Graphic Object Oriented Analysis Laboratory. In *Companion of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 38–39. ACM, 2002.

[92] J.L. Peterson. *Petri net theory and the modeling of systems*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1981.

[93] E. Pitler, M. Raghupathy, H. Mehta, A. Nenkova, A. Lee, and A. Joshi. Easily identifiable discourse relations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008), Manchester, UK, August*. Citeseer, 2008.

[94] M. Poesio and M.A. Kabadjov. A general-purpose, off-the-shelf anaphora resolution module: Implementation and preliminary evaluation. In *Proc. LREC*. Citeseer, 2004.

[95] A. Polyvyanyy, S. Smirnov, and M. Weske. On application of structural decomposition for process model abstraction. In *Proceedings of the 2nd International Conference on Business Process and Services Computing*, Leipzig, March 2009.

[96] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[97] R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. The Penn Discourse Treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 2961–2968. Citeseer, 2008.

[98] R. Prasad, E. Miltsakaki, N. Dinesh, A. Lee, A. Joshi, L. Robaldo, and B. Webber. The penn discourse treebank 2.0 annotation manual. *December*, 17:2007, 2007.

[99] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in-speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[100] A.N. Rafferty and C.D. Manning. Parsing three German treebanks: Lexicalized and unlexicalized baselines. In *Proceedings of the Workshop on Parsing German*, pages 40–46. Association for Computational Linguistics, 2008.

[101] A. Ratnaparkhi et al. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, volume 1, pages 133–142, 1996.

[102] J. Recker and M. Rosemann. Business Process Modeling-A Comparative Analysis. *Journal of the Association for Information Systems*, 10(4):333–363, 2009.

[103] J.C. Recker, M. Indulska, M. Rosemann, and P. Green. How good is BPMN really? Insights from theory and practice. In *Proceedings 14th European Conference on Information Systems*, 2006.

[104] H.A. Reijers. *Design and control of workflow processes: business process management for the service industry*. Eindhoven University Press, 2003.

[105] H.A. Reijers, S. Limam, and W.M.P. Van Der Aalst. Product-based workflow design. *Journal of Management Information Systems*, 20(1):229–262, 2003.

[106] J. Ruppenhofer, M. Ellsworth, M.R.L. Petruck, C.R. Johnson, and J. Scheffczyk. Framenet ii: Extended theory and practice. Technical report, International Computer Science Institute, University of California at Berkeley, 2006.

[107] W. Sadiq and M.E. Orlowska. Analyzing process models using graph reduction techniques. *Information systems*, 25(2):117–134, 2000.

[108] A. Salbrechter, HC Mayr, and C. Kop. Mapping Pre-designed Business Process Models to UML. In *Proceedings of the Eighth IASTED International Conference*. Acta Press, 2004.

[109] M. Sandelowski. The problem of rigor in qualitative research. *Advances in Nursing Science*, 8(3):27, 1986.

[110] A.W. Scheer. *ARIS-business process modeling*. Springer Verlag, 2000.

[111] J. Scheffczyk, A. Pease, and M. Ellsworth. Linking FrameNet to the suggested upper merged ontology. In *Proceeding of the 2006 Conference on Formal Ontology in Information Systems (FOIS 2006)*, pages 289–300. IOS Press, 2006.

[112] S. Schiffer, A. Ferrein, and G. Lakemeyer. Qualitative world models for soccer robots. In *Workshop at KI 2006*, page 3. Citeseer, 2006.

[113] Aaron D. Scriver. Semantic distance in wordnet: A simplified and improved measure of semantic relatedness. Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 2006.

[114] J. Seemann. Extending the sugiyama algorithm for drawing UML class diagrams: Towards automatic layout of object-oriented software diagrams. In *Graph Drawing*, pages 415–424. Springer, 1997.

[115] L. Shi and R. Mihalcea. Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. In *proceedings of the 6th international conference on computational linguistics and intelligent text processing*, page 100. Springer Verlag, 2005.

[116] H.A. Simon. *The sciences of the artificial*. MIT Press, 1996.

[117] A. Sinha and A. Paradkar. Use Cases to Process Specifications in Business Process Modeling Notation. In *2010 IEEE International Conference on Web Services*, pages 473–480. IEEE, 2010.

[118] A. Sinha, A. Paradkar, P. Kumanan, and B. Boguraev. An Analysis Engine for Dependable Elicitation on Natural Language Use Case Description and its Application to Industrial Use Cases. Technical report, IBM, 2008.

[119] H. Smith and P. Fingar. *Business process management: The Third wave*. Meghan-Kiffer Press, 2003.

[120] A.L. Souter, L.L. Pollock, and D. Hisley. Inter-class def-use analysis with partial class representations. In *Proceedings of the 1999 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*, pages 47–56. ACM, 1999.

[121] V. Stoyanov, C. Cardie, N. Gilbert, E. Riloff, D. Buttler, and D Hysom. Coreference Resolution with Reconcile. In *Proceedings of the Conference of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, 2010.

[122] Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *proceedings of the second international conference on Information and knowledge management (CIKM '93)*, pages 67–74, New York, NY, USA, 1993. ACM.

[123] M. Swan. *Practical english usage*. Oxford University Press, 2005. list of adverbs taken from here.

[124] W.M.P. Van Der Aalst, A.H.M. Hofstede, and M. Weske. Business process management: a survey. In *Proceedings of the 2003 international conference on Business process management*, pages 1–12. Springer-Verlag, 2003.

[125] W.M.P. Van Der Aalst and A.H.M. Ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[126] BF Van Dongen, A.K.A. de Medeiros, HMW Verbeek, A. Weijters, and WMP Van der Aalst. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005*, pages 444–454. Springer, 2005.

[127] BF van Dongen, RM Dijkman, and J. Mendling. Measuring similarity between business process models. In *CAiSE*, pages 450–464. Springer, 2008.

[128] BF Van Dongen, J. Mendling, and WMP van der Aalst. Structural patterns for soundness of business process models. In *10th IEEE International Conference on Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 116–128. IEEE, 2006.

[129] CJ Van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[130] Y. Versley, S.P. Ponzetto, M. Poesio, V. Eidelman, A. Jern, J. Smith, X. Yang, and A. Moschitti. BART: A modular toolkit for coreference resolution. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session*, pages 9–12. Association for Computational Linguistics, 2008.

[131] K. Verspoor, K.B. Cohen, and L. Hunter. The textual characteristics of traditional and Open Access scientific journals are similar. *BMC bioinformatics*, 10(1):183, 2009.

[132] T. Wahl and G. Sindre. An analytical evaluation of BPMN using a semiotic quality framework. *Advanced topics in database research*, 5:94 – 105, 2006.

[133] Harry Jiannan Wang, J. Leon Zhao, and Liang-Jie Zhang. Policy-Driven Process

Mapping (PDPM): Discovering process models from business policies. *Decision Support Systems*, 48(1):267 – 281, 2009. Information product markets.

[134] M. Weske. *Business process management: concepts, languages, architectures*. Springer-Verlag New York Inc, 2007.

[135] M White. Information overlook. EContent(26:7), 2003.

[136] S.A. White and D. Miers. *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies Inc., 2008.

[137] David Yarowsky. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proceedings of the 14th Conference on Computational linguistics*, pages 454–460, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[138] T. Yue, L. Briand, and Y. Labiche. A Use Case Modeling Approach to Facilitate the Transition towards Analysis Models: Concepts and Empirical Evaluation. In *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, pages 484–498. Springer-Verlag, 2009.

[139] T. Yue, L. Briand, and Y. Labiche. Automatically Deriving a UML Analysis Model from a Use Case Model. Technical report, Carleton University, 2009.

[140] T. Yue, L. Briand, and Y. Labiche. An Automated Approach to Transform Use Cases into Activity Diagrams. In *Modelling Foundations and Applications*, pages 337–353. Springer, 2010.

Appendix A. Detailed Evaluation Results

In this section the results of our evaluation will be presented in more detail then in section 4.3. First, table A.9 will show the and ID, the source, the name, and the type of each of the 47 models. Using these IDs table A.10 will display the characteristics of the textual description, the process model generated by our approach, and the process model as created by a human. The measured characteristics are the same as in section 4.3:

- Number of sentences (n)
- Average length of the sentences (∅l)
- Number of sentences describing the process on a meta level (m)
- Number of identified relative references (r)
- Number of textual links links and jumps (j)
- Size of the models regarding nodes (N)
- Size of the models regarding edges (E)
- Similarity (sim)

Table A.9: Model ID, Source and Name Overview

ID	Source and Model	Type
1-1	Bicycle manufacturing	academic
1-2	Computer repair	academic
1-3	Hotel Service	academic
1-4	Underwriters	academic
1	**HU Berlin - Total - 4 models**	**academic**
2-1	SLA Violation	academic

Continued on next page

Table A.9 – continued from previous page

ID	Source and Model	Type
2-2	Supplier Switch	academic
2	**TU Berlin - Total - 2 models**	**academic**
3-1	2009-1 MC Finalise SCT Warrant Possession	academic
3-2	2009-2 Conduct Directiosn Hearings	academic
3-3	2009-3 Repetition/Cylces	academic
3-4	2009-4 event-based gateways	academic
3-5	2009-5 P&E Lodge Originating Document by Post	academic
3-6	2010-1 Claims Notification	academic
3-7	2010-2 Claims Creation	academic
3-8	2010-3 Claims Handling	academic
3	**QUT - Total - 8 models**	**academic**
4-1	Intaker Workflow	academic
4	**TU Eindhoven - Total - 1 model**	**academic**
5-1	Active VOS	industry
5-2	BizAgi 1	industry
5-3	BizAgi 2	industry
5-4	Oracle	industry
5	**Vendor Tutorials - Total - 4 models**	**industry**
6-1	ACME	industry
6-2	Help - Tutorial	industry
6-3	Powerlicht	industry
6-4	Turbopixel	industry
6	**inubit AG - Total - 4 models**	**industry**
7-1	Frank Puhlmann - Calling Leads	industry
7	**BPM Practicioners - Total - 1 model**	**industry**
8-1	HR Process - Simple	textbook
8-2	HR Process - HR	textbook
8-3	HR Process - Functional Department	textbook
8	**BPMN Practical Handbook - Total - 3 Models**	**textbook**

Continued on next page

Table A.9 – continued from previous page

ID	Source and Model	Type
9-1	Exercise 1	textbook
9-2	Exercise 2	textbook
9-3	Exercise 3a	textbook
9-4	Exercise 3b	textbook
9-5	Exercise 4	textbook
9-6	Exercise 5	textbook
9	**BPMN Modeling and Reference guide - Total - 6 models**	**textbook**
10-1	B2	public sector
10-2	B3	public sector
10-3	B4	public sector
10-4	B5.1	public sector
10-5	B5.2	public sector
10-6	B6	public sector
10-7	B7	public sector
10-8	B8	public sector
10-9	C1	public sector
10-10	C2	public sector
10-11	C3	public sector
10-12	D1	public sector
10-13	D2	public sector
10-14	D3	public sector
10	**FNA - Metrology Processes - Total - 14 models**	**public sector**

Table A.10: Detailed characteristics of the analyzed text and models

ID	n	∅l	m	r	j	$\|N_{gen}\|$	$\|N_{hum}\|$	$\Delta\|N\|$	$\|G_{gen}\|$	$\|G_{hum}\|$	$\Delta\|G\|$	$\|E_{gen}\|$	$\|E_{hum}\|$	$\Delta\|E\|$	sim
1-1	12	14.92	4	3	0	34	22	35.29%	8	6	25.00%	31	20	35.48%	76.03%
1-2	6	18.83	3	1	0	24	26	-8.33%	3	4	-33.33%	23	26	-13.04%	85.46%
1-3	11	16.91	2	4	0	35	29	17.14%	7	9	-28.57%	34	29	14.71%	77.17%
1-4	11	21.91	3	13	0	28	26	7.14%	4	5	-25.00%	27	23	14.81%	73.10%
1	**10.00**	**18.14**	**3.00**	**5.25**	**0.00**	**30.25**	**25.75**	**14.88%**	**5.50**	**6.00**	**-9.09%**	**28.75**	**24.50**	**14.78%**	**77.94%**
2-1	38	20.26	6	8	4	98	83	15.31%	17	9	47.06%	104	87	16.35%	69.32%
2-2	30	22.08	9	7	1	85	59	30.59%	9	10	-11.11%	84	72	14.29%	72.25%
2	**34.00**	**21.17**	**7.50**	**7.50**	**2.50**	**91.50**	**71.00**	**22.40%**	**13.00**	**9.50**	**26.92%**	**94.00**	**79.50**	**15.43%**	**70.79%**
3-1	7	16.85	0	1	0	27	17	37.04%	0	0	0.00%	23	18	21.74%	71.85%
3-2	4	22.00	0	1	0	13	12	7.69%	3	4	-33.33%	13	13	0.00%	82.46%
3-3	5	17.40	1	3	0	14	8	42.86%	3	2	33.33%	14	8	42.86%	75.57%
3-4	4	19.75	0	1	0	11	8	27.27%	2	2	0.00%	11	8	27.27%	78.85%
3-5	9	19.33	0	2	0	40	33	17.50%	4	2	50.00%	43	39	9.30%	80.54%
3-6	8	15.25	0	3	0	20	12	40.00%	7	3	57.14%	22	12	45.45%	74.94%
3-7	5	19.20	1	0	0	14	12	14.29%	0	0	0.00%	15	14	6.67%	85.52%
3-8	7	16.29	2	0	0	23	17	26.09%	2	2	0.00%	20	16	20.00%	80.51%
3	**6.13**	**18.26**	**0.50**	**1.38**	**0.00**	**20.25**	**14.88**	**26.54%**	**2.63**	**1.88**	**28.57%**	**20.13**	**16.00**	**20.50%**	**78.78%**
4-1	40	18.45	8	4	1	63	38	39.68%	1	8	-700.00%	52	37	28.85%	41.54%
4	**40.00**	**18.45**	**8.00**	**4.00**	**1.00**	**63.00**	**38.00**	**39.68%**	**1.00**	**8.00**	**-700.00%**	**52.00**	**37.00**	**28.85%**	**41.54%**
5-1	6	15.50	1	0	0	15	7	53.33%	3	1	66.67%	15	6	60.00%	37.65%
5-2	5	18.00	1	0	2	14	12	14.29%	1	1	0.00%	10	8	20.00%	80.25%
5-3	10	21.30	1	2	2	34	17	50.00%	5	1	80.00%	31	13	58.06%	64.27%
5-4	15	18.00	2	5	3	36	20	44.44%	8	6	25.00%	36	19	47.22%	72.33%
5	**9.00**	**18.20**	**1.25**	**1.75**	**1.75**	**24.75**	**14.00**	**43.43%**	**4.25**	**2.25**	**47.06%**	**23.00**	**11.50**	**50.00%**	**63.63%**
6-1	18	26.16	2	12	1	48	36	25.00%	2	6	-200.00%	40	37	7.50%	42.51%
6-2	5	13.20	1	0	0	10	10	0.00%	0	0	0.00%	8	6	25.00%	78.78%

Continued on next page

Table A.10 – continued from previous page

ID	n	∅l	m	r	j	$\lvert N_{\text{gen}}\rvert$	$\lvert N_{\text{hum}}\rvert$	$\Delta\lvert N\rvert$	$\lvert G_{\text{gen}}\rvert$	$\lvert G_{\text{hum}}\rvert$	$\Delta\lvert G\rvert$	$\lvert E_{\text{gen}}\rvert$	$\lvert E_{\text{hum}}\rvert$	$\Delta\lvert E\rvert$	sim
6-3	9	19.88	3	7	0	25	14	44.00%	1	2	-100.00%	21	10	52.38%	35.74%
6-4	14	14.28	3	13	1	36	36	0.00%	8	9	-12.50%	32	32	0.00%	86.70%
6	**11.50**	**18.38**	**2.25**	**8.00**	**0.50**	**29.75**	**24.00**	**19.33%**	**2.75**	**4.25**	**-54.55%**	**25.25**	**21.25**	**15.84%**	**60.93%**
7-1	7	9.71	0	5	0	14	13	7.14%	2	1	50.00%	11	9	18.18%	74.35%
7	**7.00**	**9.71**	**0.00**	**5.00**	**0.00**	**14.00**	**13.00**	**7.14%**	**2.00**	**1.00**	**50.00%**	**11.00**	**9.00**	**18.18%**	**74.35%**
8-1	3	17.67	0	0	1	10	13	-30.00%	0	0	0.00%	6	7	-16.67%	85.21%
8-2	6	15.83	0	8	0	17.00	13.00	23.53%	1.00	2.00	-100.00%	14.00	19.00	-35.71%	68.42%
8-3	5	17.60	0	7	0	13	13	0.00%	2	2	0.00%	11	18	-63.64%	78.84%
8	**4.67**	**17.03**	**0.00**	**5.00**	**0.33**	**13.33**	**13.00**	**2.50%**	**1.00**	**1.33**	**-33.33%**	**10.33**	**14.67**	**-41.94%**	**77.49%**
9-1	8	20.75	1	4	1	24	29	-20.83%	5	3	40.00%	25	25	0.00%	70.03%
9-2	5	21.00	0	0	1	21	21	0.00%	3	2	33.33%	18	17	5.56%	80.88%
9-3	4	21.75	0	0	0	13	20	-53.85%	0	0	0.00%	9	16	-77.78%	70.75%
9-4	5	21.80	0	0	0	18	22	-22.22%	0	0	0.00%	13	17	-30.77%	75.18%
9-5	7	19.29	0	0	0	24	15	37.50%	4	3	25.00%	24	18	25.00%	69.90%
9-6	13	20.00	4	5	0	34	36	-5.88%	8	10	-25.00%	36	49	-36.11%	63.90%
9	**7.00**	**20.77**	**0.83**	**1.50**	**0.33**	**22.33**	**23.83**	**-6.72%**	**3.33**	**3.00**	**10.00%**	**20.83**	**23.67**	**-13.60%**	**71.77%**
10-1	3	10.66	0	0	0	14	14	0.00%	1	1	0.00%	12	13	-8.33%	96.01%
10-2	14	18.29	0	0	1	61	55	9.84%	17	10	41.18%	70	64	8.57%	85.61%
10-3	11	12.82	0	0	0	33	31	6.06%	5	5	0.00%	39	33	15.38%	91.13%
10-4	9	14.44	0	0	0	36	37	-2.78%	6	6	0.00%	42	44	-4.76%	92.88%
10-5	4	11.75	0	0	0	11	10	9.09%	0	0	0.00%	10	9	10.00%	89.80%
10-6	3	11.00	0	0	0	14	14	0.00%	1	1	0.00%	13	13	0.00%	96.54%
10-7	7	10.86	0	0	0	21	21	0.00%	2	1	50.00%	22	20	9.09%	91.17%
10-8	7	12.00	0	1	0	20	19	5.00%	2	1	50.00%	21	19	9.52%	90.11%
10-9	5	14.20	0	1	0	20	23	-15.00%	3	4	-33.33%	20	25	-25.00%	87.48%
10-10	8	14.50	0	1	1	21	26	-23.81%	1	3	-200.00%	20	27	-35.00%	84.14%
10-11	7	11.29	0	0	0	19	24	-26.32%	4	4	0.00%	18	25	-38.89%	85.29%

Continued on next page

Table A.10 – continued from previous page

ID	n	∅l	m	r	j	$\lvert N_{\text{gen}}\rvert$	$\lvert N_{\text{hum}}\rvert$	$\Delta\lvert N\rvert$	$\lvert G_{\text{gen}}\rvert$	$\lvert G_{\text{hum}}\rvert$	$\Delta\lvert G\rvert$	$\lvert E_{\text{gen}}\rvert$	$\lvert E_{\text{hum}}\rvert$	$\Delta\lvert E\rvert$	sim
10-12	4	13.75	0	0	0	21	18	14.29%	2	1	50.00%	19	16	15.79%	89.72%
10-13	3	14.33	0	0	0	11	11	0.00%	0	0	0.00%	10	9	10.00%	95.04%
10-14	5	25.40	0	0	3	52	39	25.00%	8	7	12.50%	66	46	30.30%	82.42%
10	**6.43**	**13.95**	**0.00**	**0.21**	**0.36**	**25.29**	**24.43**	**3.39%**	**3.71**	**3.14**	**15.38%**	**27.29**	**25.93**	**4.97%**	**89.81%**
All	**9.19**	**17.16**	**1.23**	**2.60**	**0.49**	**27.43**	**23.21**	**15.36%**	**3.72**	**3.38**	**9.14%**	**26.77**	**23.64**	**11.69%**	**76.98%**

Appendix B. Detailed Test Data Sets

Appendix B.1. Models provided by the Humboldt-Universität zu Berlin

A small company manufactures customized bicycles. Whenever the sales department receives an order, a new process instance is created. A member of the sales department can then reject or accept the order for a customized bike. In the former case, the process instance is finished. In the latter case, the storehouse and the engineering department are informed. The storehouse immediately processes the part list of the order and checks the required quantity of each part. If the part is available in-house, it is reserved. If it is not available, it is back-ordered. This procedure is repeated for each item on the part list. In the meantime, the engineering department prepares everything for the assembling of the ordered bicycle. If the storehouse has successfully reserved or back-ordered every item of the part list and the preparation activity has finished, the engineering department assembles the bicycle. Afterwards, the sales department ships the bicycle to the customer and finishes the process instance.

Text 1: Process Description 1-1: Bicycle manufacturing.

A customer brings in a defective computer and the CRS checks the defect and hands out a repair cost calculation back. If the customer decides that the costs are acceptable, the process continues, otherwise she takes her computer home unrepaired. The ongoing repair consists of two activities, which are executed, in an arbitrary order. The first activity is to check and repair the hardware, whereas the second activity checks and configures the software. After each of these activities, the proper system functionality is tested. If an error is detected another arbitrary repair activity is executed, otherwise the repair is finished.

Text 2: Process Description 1-2: Computer repair.

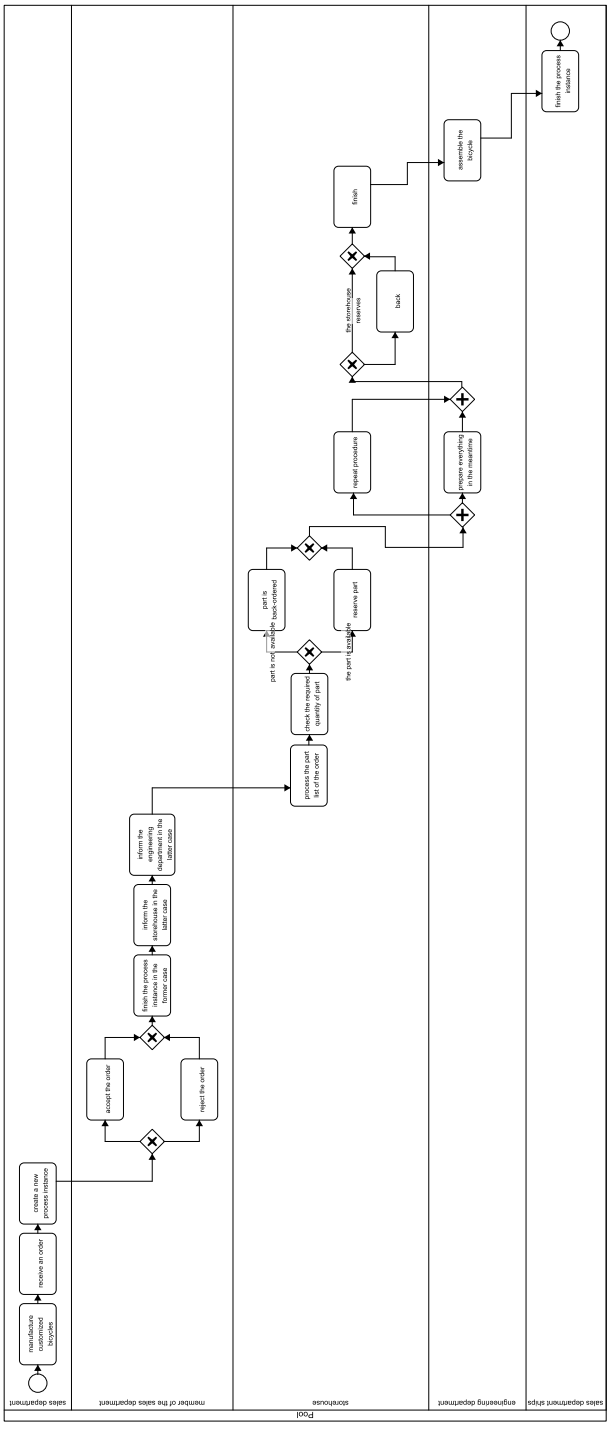

Figure B.21: Model 1-1 as generated by our system.

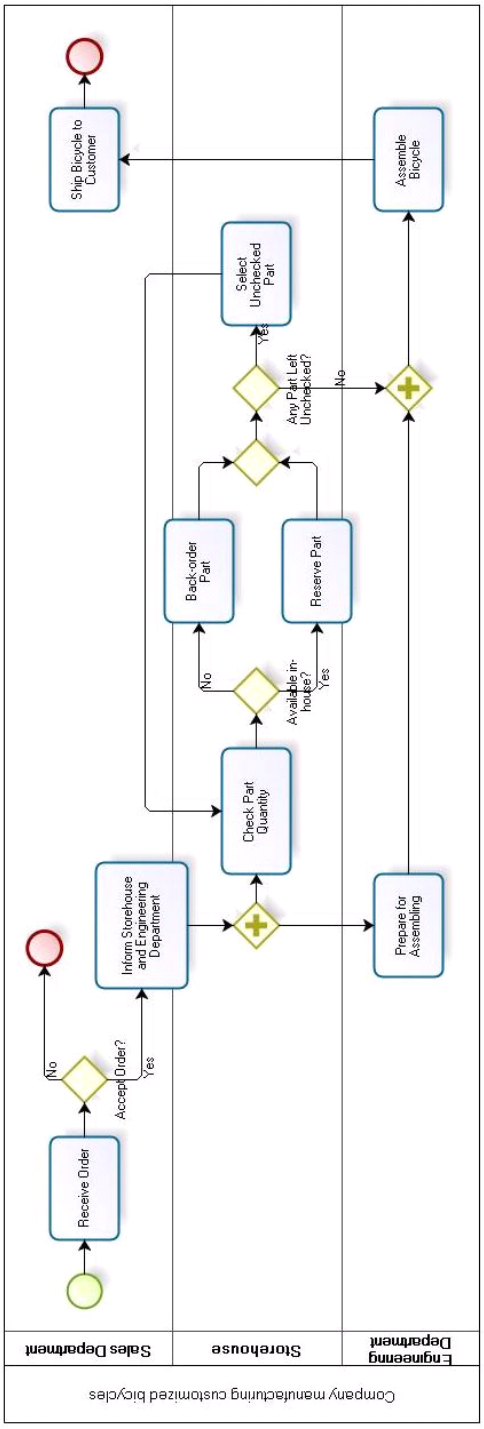

Figure B.22: Model 1-1 as created by a human modeler.

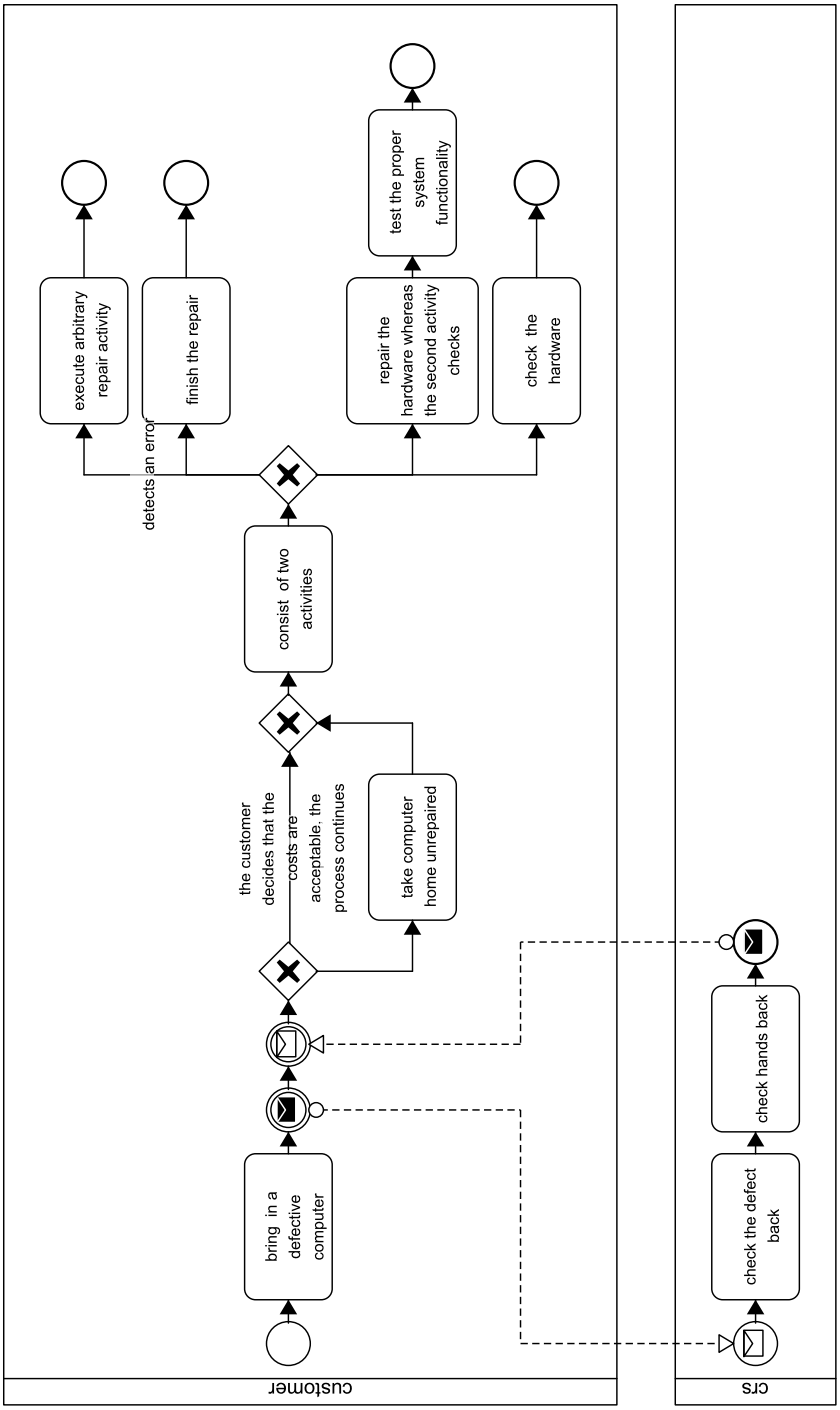

Figure B.23: Model 1-2 as generated by our system.

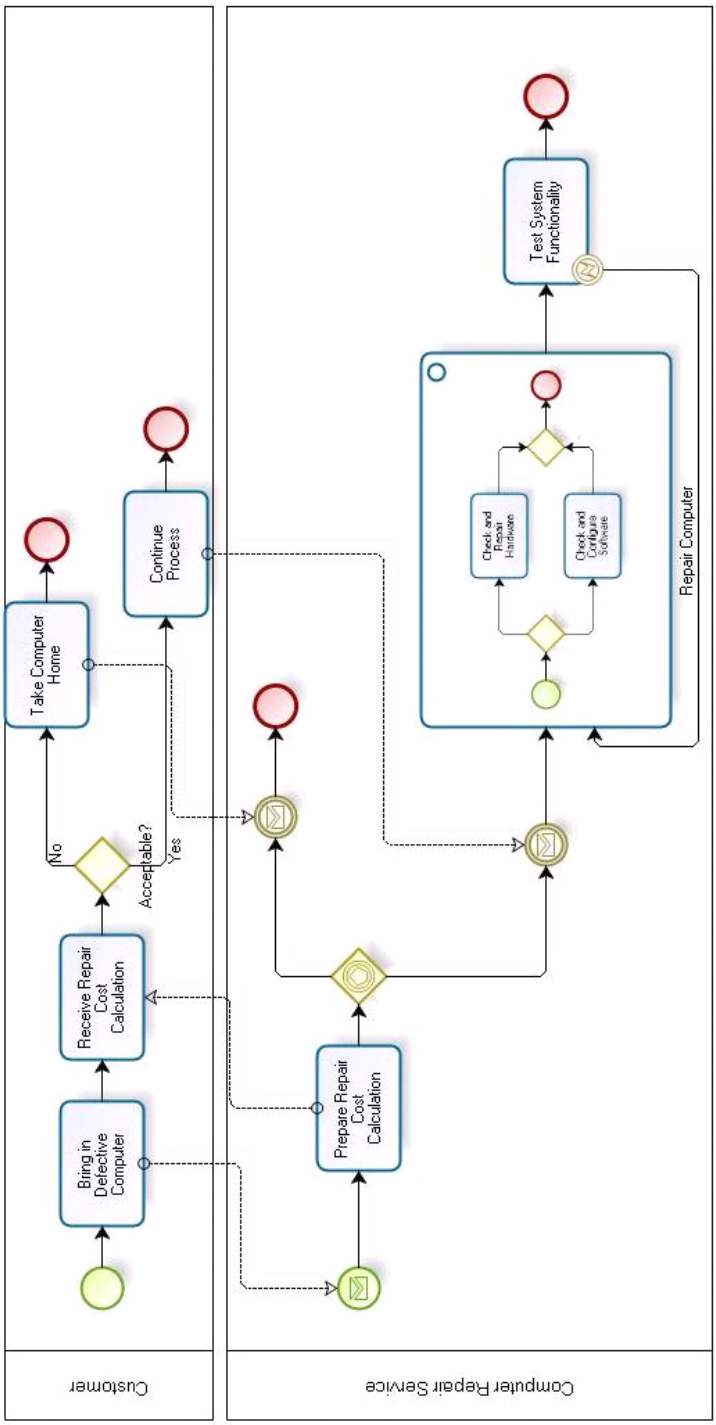

Figure B.24: Model 1-2 as created by a human modeler.

The Evanstonian is an upscale independent hotel. When a guest calls room service at The Evanstonian, the room-service manager takes down the order. She then submits an order ticket to the kitchen to begin preparing the food. She also gives an order to the sommelier (i.e., the wine waiter) to fetch wine from the cellar and to prepare any other alcoholic beverages. Eighty percent of room-service orders include wine or some other alcoholic beverage. Finally, she assigns the order to the waiter. While the kitchen and the sommelier are doing their tasks, the waiter readies a cart (i.e., puts a tablecloth on the cart and gathers silverware). The waiter is also responsible for nonalcoholic drinks. Once the food, wine, and cart are ready, the waiter delivers it to the guests room. After returning to the room-service station, the waiter debits the guests account. The waiter may wait to do the billing if he has another order to prepare or deliver.

Text 3: Process Description 1-3: Hotel Service.

Whenever a company makes the decision to go public, its first task is to select the underwriters. Underwriters act as financial midwives to a new issue. Usually they play a triple role: First they provide the company with procedural and financial advice, then they buy the issue, and finally they resell it to the public. Established underwriters are careful of their reputation and will not handle a new issue unless they believe the facts have been presented fairly. Thus, in addition to handling the sale of a companys issue, the underwriters in effect give their seal of approval to it. They prepare a registration statement for the approval of the Securities and Exchange Commission (SEC). In addition to registering the issue with the SEC, they need to check that the issue complies with the so-called blue-sky laws of each state that regulate sales of securities within the state. While the registration statement is awaiting approval, underwriters begin to firm up the issue price. They arrange a road show to talk to potential investors. Immediately after they receive clearance from the SEC, underwriters fix the issue price. After that they enter into a firm commitment to buy the stock and then offer it to the public, when they havent still found any reason not to do it.

Text 4: Process Description 1-4: Underwriters (original source [8]).

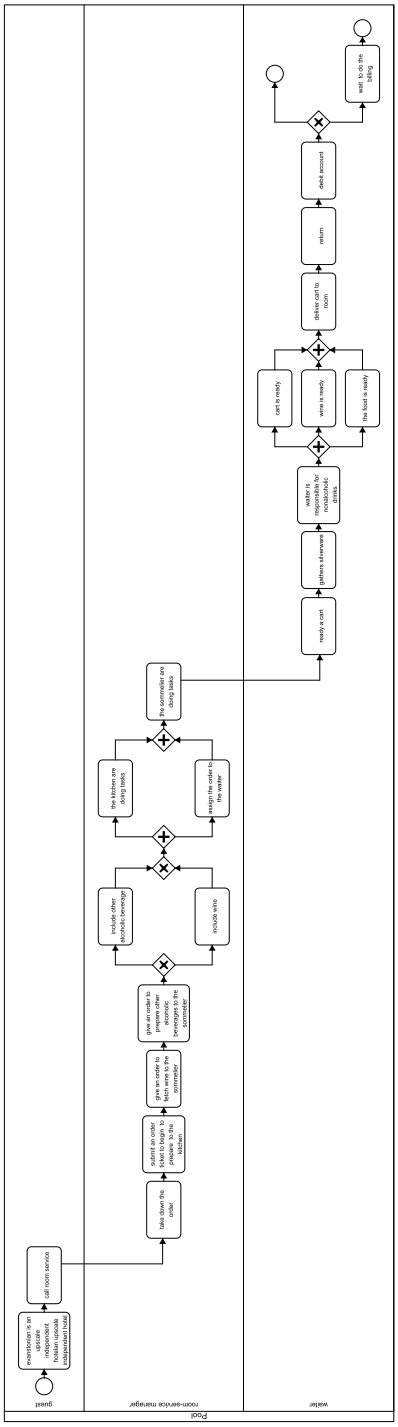

Figure B.25: Model 1-3 as generated by our system.

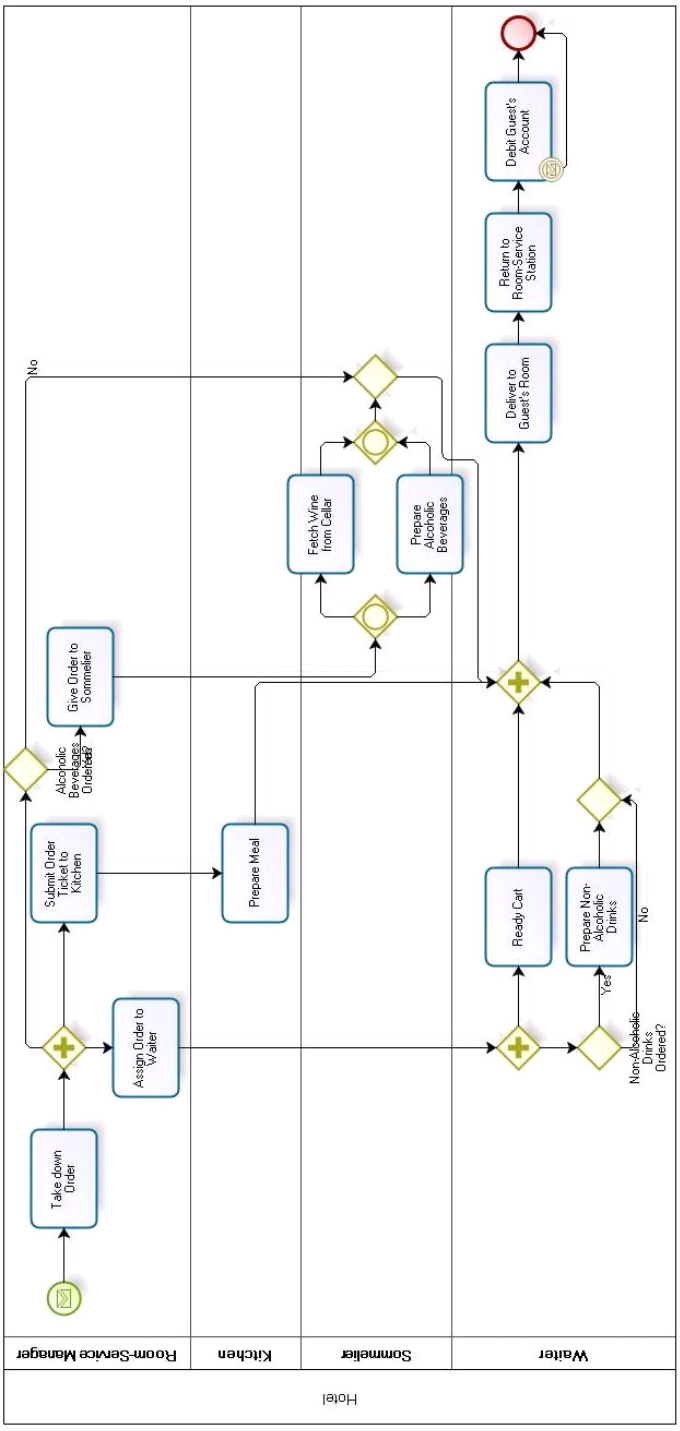

Figure B.26: Model 1-3 as created by a human modeler.

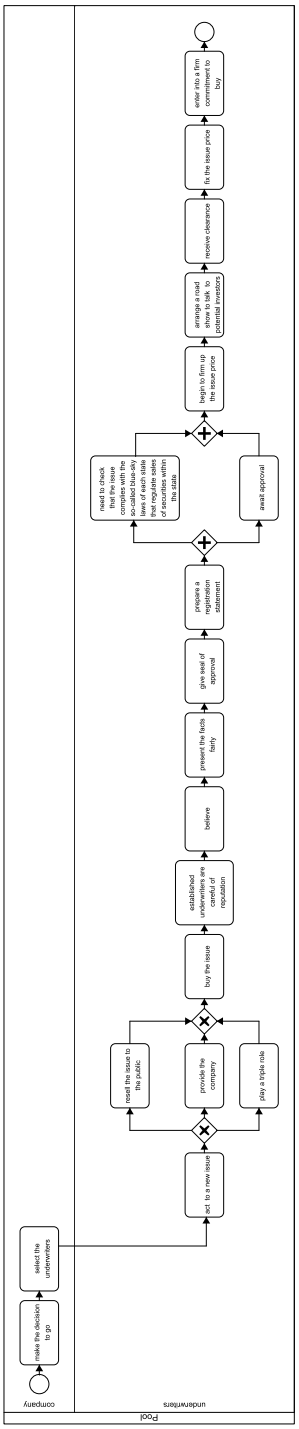

Figure B.27: Model 1-4 as generated by our system.

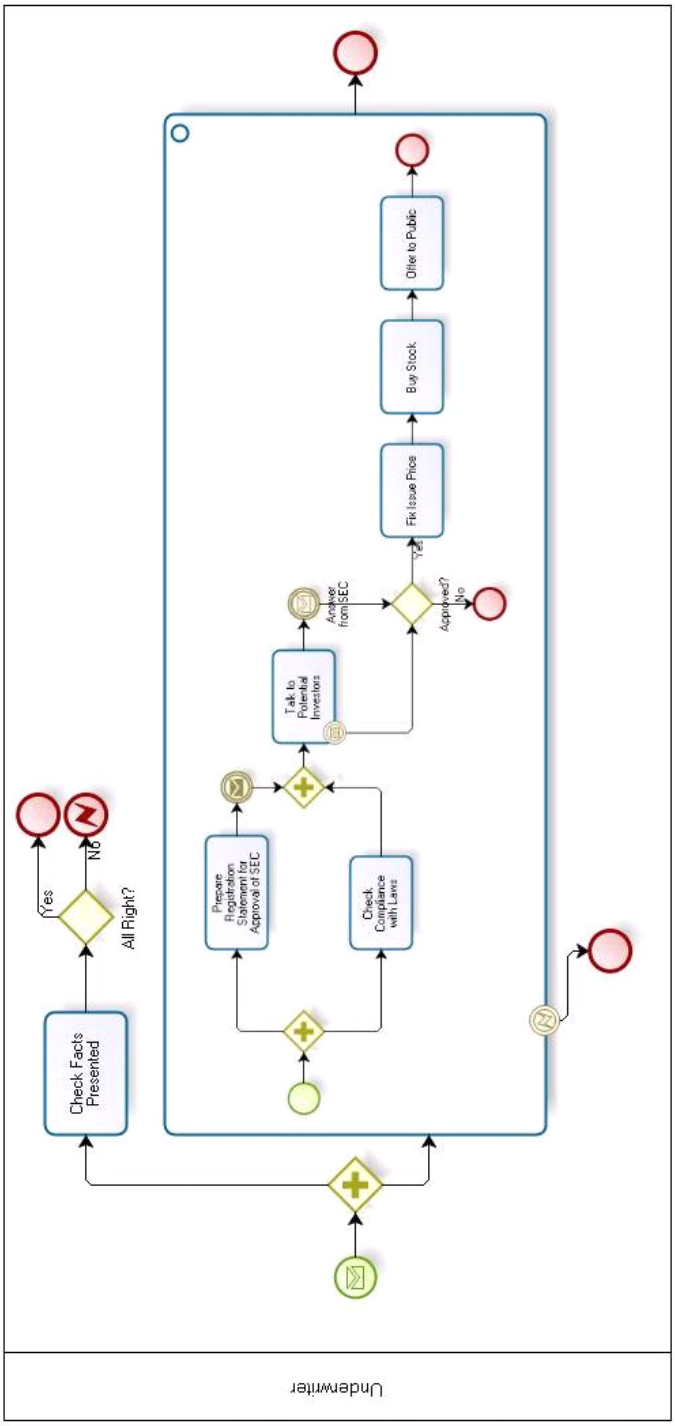

Figure B.28: Model 1-4 as created by a human modeler.

Appendix B.2. Models provided by the Technische Universität Berlin

At the beginning the customer perceives that her subscribed service has degraded. A list with all the problem parameters is then sent to the Customer Service department of TELECO. At the customer service an employee enters (based on the received data) a problem report into system T.. Then the problem report is compared to the customer SLA to identify what the extent and the details of the service degradation are. Based on this, the necessary counter measures are determined including their respective priorities. An electronic service then determines the significance of the customer based on information that has been collected during the history of the contractual relationship. In case the customer is premium, the process will link to an extra problem fix process (this process will not be detailed here). In case the customer is of certain significance which would affect the counter measures previously decided upon, the process goes back to re-prioritize these measures otherwise the process continues. Taking together the information (i.e. contract commitment data + prioritized actions) a detailed problem report is created. The detailed problem report is then sent to Service Management. Service Management deals on a first level with violations of quality in services that are provided to customers. After receiving the detailed problem report, Service management investigates whether the problem is analyzable at the level of their department or whether the problem may be located at Resource Provisioning. In case Service Management assesses the problem to be not analyzable by themselves, the detailed problem report is sent out to Resource Provisioning. If Service Management is sure they can analyze it, they perform the analysis and based on the outcome they create a trouble report that indicates the type of problem. After Resource Provisioning receives the detailed problem report, it is checked whether there are any possible problems. If no problems are detected, a notification about the normal service execution is created. If a problem is detected this will be analyzed by Resource Provisioning and a trouble report is created. Either trouble report or the normal execution notification will be included in a status report and sent back to Service Management. Service Management then prepares the final status report based on the received information. Subsequently it has to be determined what counter measures should be taken depending on the information in the final status report. Three alternative process paths may be taken. For the case that no problem was detected at all, the actual service performance is sent back to the Customer Service. For the case that minor corrective actions are required, Service Management will undertake corrective actions by themselves. Subsequently, the problem resolution report is created and then sent out to Customer Service. After sending, this process path of Service Management ends. For the case that automatic resource restoration from Resource Provisioning is required, Service Management must create a request for automatic resource restoration. This message is then sent to Resource Provisioning. Resource Provisioning has been on-hold and waiting for a restoration request but this must happen within 2 days after the status report was sent out, otherwise Resource Provisioning terminates the process. After the restoration request is received, all possible errors are tracked. Based on the tracked errors, all necessary corrective actions are undertaken by Resource Provisioning. Then a trouble-shooting report is created. This report is sent out to Service Management; then the process ends. The trouble-shooting report is received by Service Management and this information goes then into the creation of the problem resolution report just as described for ii). Customer Service either receives the actual service performance (if there was no problem) or the problem resolution report. Then, two concurrent activities are triggered, i.e. i) a report is created for the customer which details the current service performance and the resolution of the problem, and ii) an SLA violation rebate is reported to Billing & Collections who will adjust the billing. The report for the customer is sent out to her. After all three activities are completed the process ends within Customer Service. After the customer then receives the report about service performance and problem resolution from Customer Service, the process flow at the customer also ends.

Text 5: Process Description 2-1: SLA Violation.

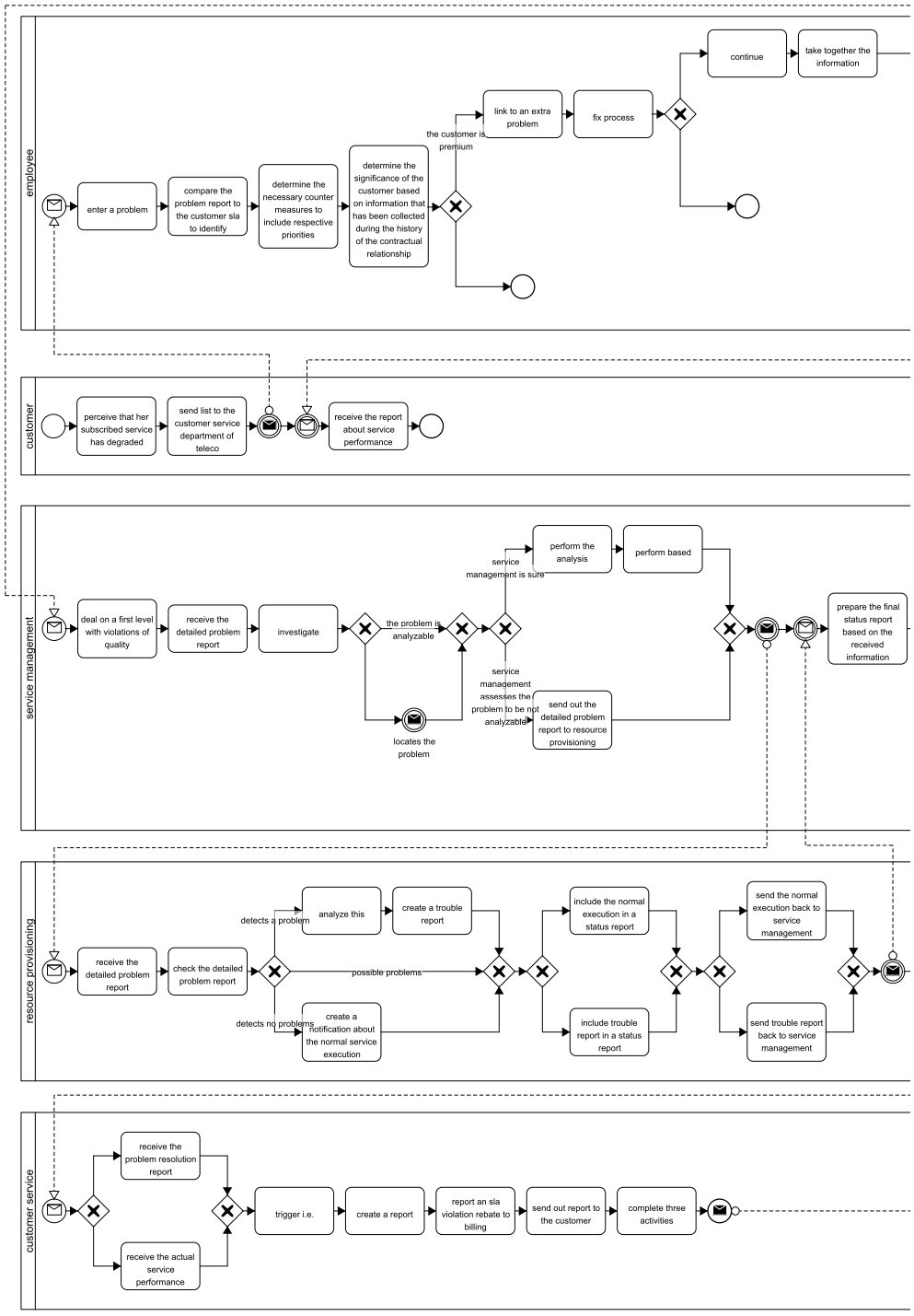

Figure B.29: Model 2-1 as generated by our system (part 1).

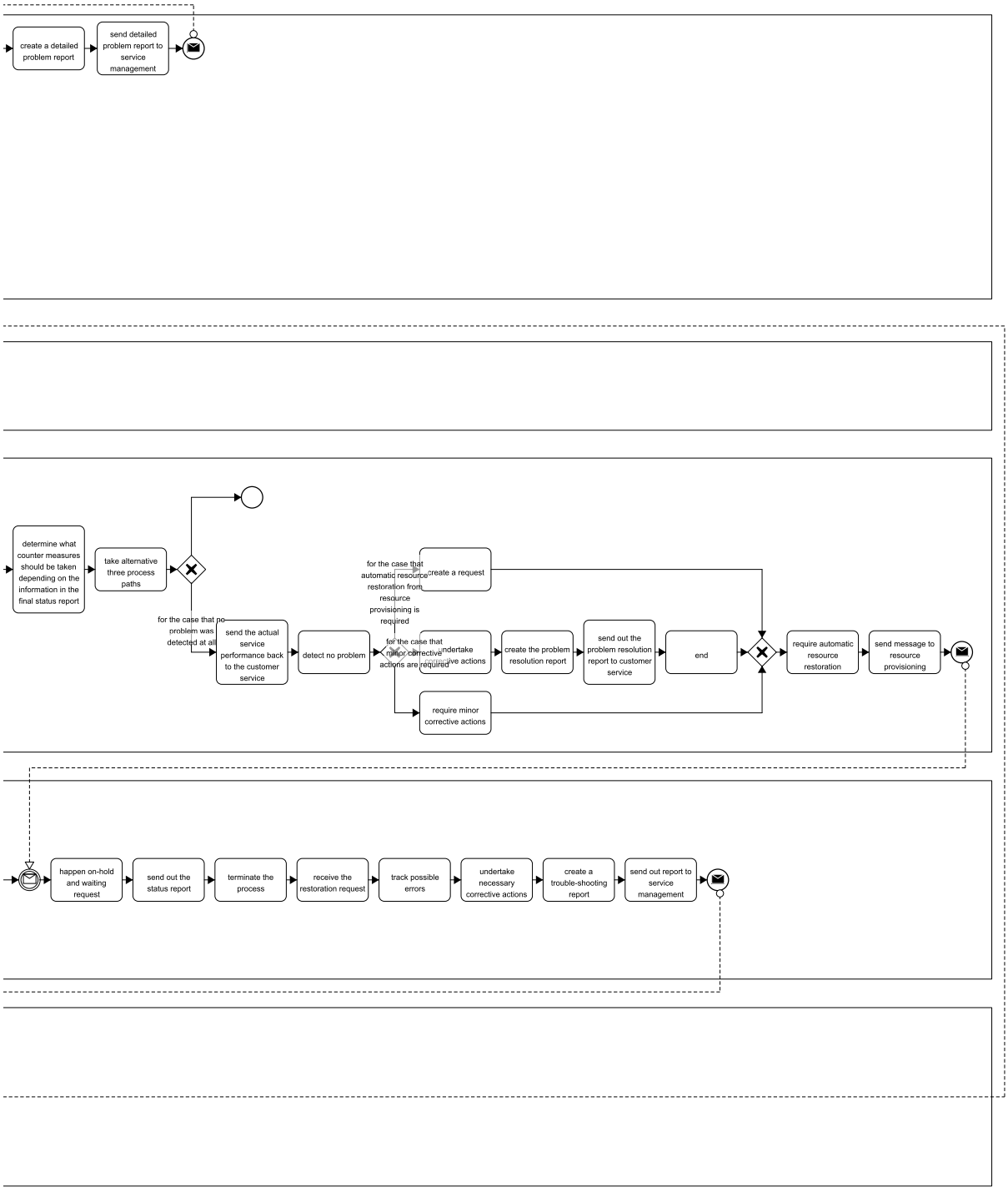

Figure B.30: Model 2-1 as generated by our system (part 2).

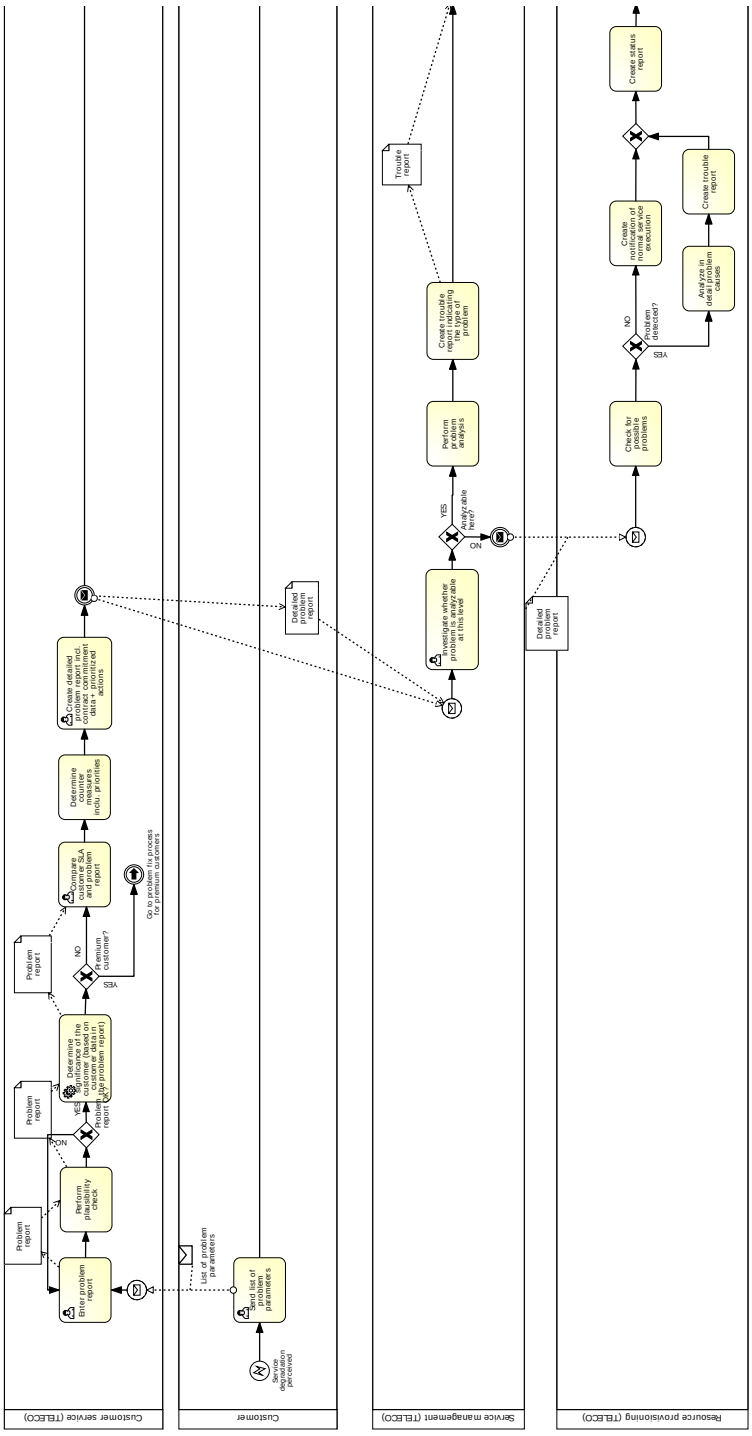

Figure B.31: Model 2-1 as created by a human modeler (part1).

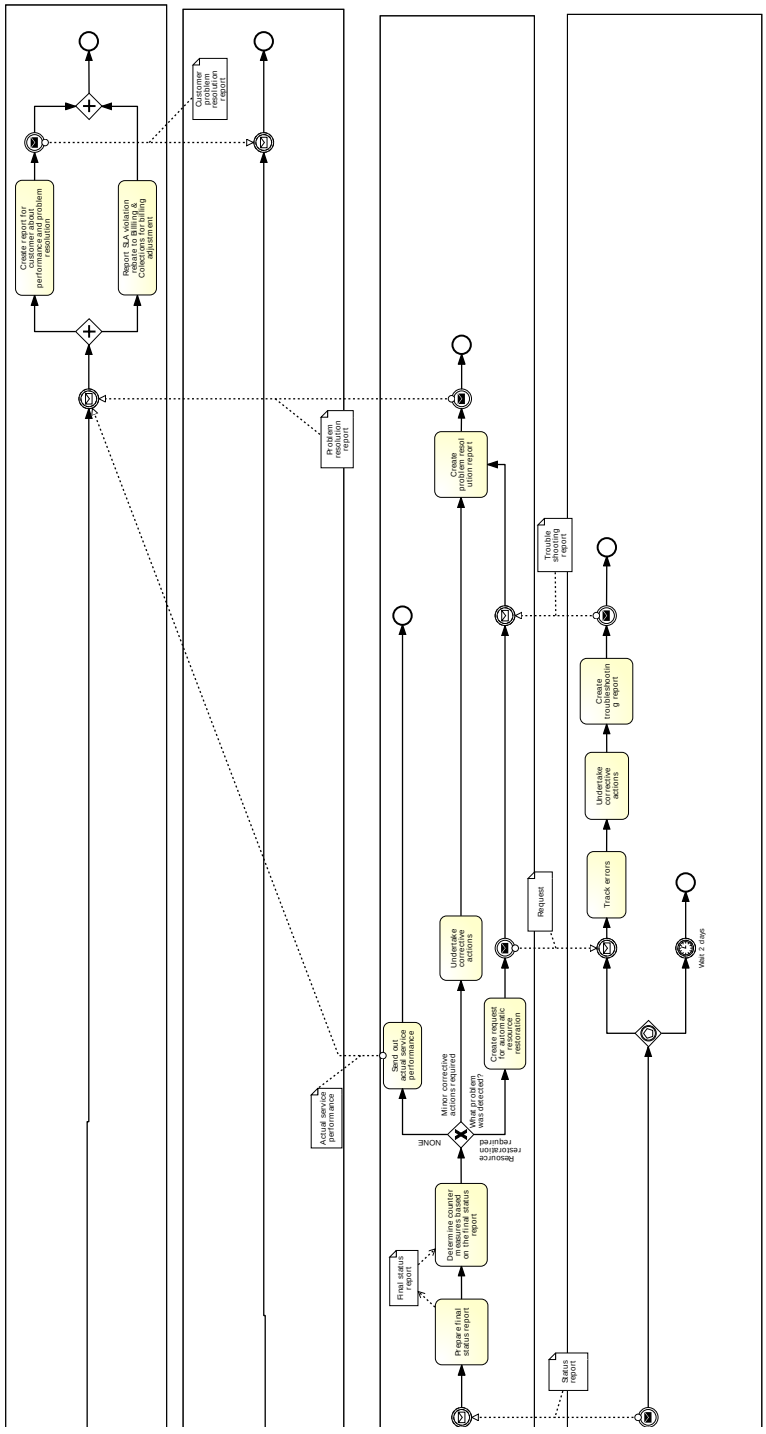

Figure B.32: Model 2-1 as created by a human modeler (part 2).

The process is initiated by a switch-over request. In doing so, the customer transmits his data to the customer service department of the company. Customer service is a shared service center between the departments Sales and Distribution. The customer data is received by customer service and based on this data a customer data object is entered into the CRM system. After customer data has been entered it should then be compared with the internal customer data base and checked for completeness and plausibility. In case of any errors these should be corrected on the basis of a simple error list. The comparison of data is done to prevent individual customer data being stored multiple times. In case the customer does not exist in the customer data base, a new customer object is being created which will remain the data object of interest during the rest of the process flow. This object consists of data elements such as the customers name and address and the assigned power gauge. The generated customer object is then used, in combination with other customer data to prepare the contract documents for the power supplier switch (including data such as bank connection, information on the selected rate, requested date of switch-over). In the following an automated check of the contract documents is carried out within the CIS (customer information system) in order to confirm their successful generation. In case of a negative response, i.e. the contract documents are not (or incorrectly) generated, the causing issues are being analyzed and resolved. Subsequently the contract documents are generated once again. In case of a positive response a confirmation document is sent out to the customer stating that the switch-over to the new supplier can be executed. A request to the grid operator is automatically sent out by the CIS. It puts the question whether the customer may be supplied by the selected supplier in the future. The switch-over request is checked by the grid operator for supplier concurrence, and the grid operator transmits a response comment. In the case of supplier concurrence the grid operator would inform all involved suppliers and demand the resolution of the conflict. The grid operator communicates with the old supplier and carries out the termination of the sales agreement between the customer and the old supplier (i.e. the customer service (of the new supplier) does not have to interact with the old supplier regarding termination). If there are not any objections by the grid operator (i.e. no supplier concurrence), customer service creates a CIS contract. The customer then has the chance to check the contract details and based on this check may decide to either withdraw from the switch contract or confirm it. Depending on the customers acceptance/rejection the process flow at customer service either ends (in case of withdrawal) or continues (in case of a confirmation). An additional constraint is that the customer can only withdraw from the offered contract within 7 days after the 7th day the contract will be regarded as accepted and the process continues. The confirmation message by the customer is therefore not absolutely necessary (as it will count as accepted after 7 days in any way) but it can speed up the switch process. On the switch-date, but no later than 10 days after power supply has begun, the grid operator transmits the power meter data to the customer service and the old supplier via messages containing a services consumption report. At the same time, the grid operator computes the final billing based on the meter data and sends it to the old supplier. Likewise the old supplier creates and sends the final billing to the customer. For the customer as well as the grid operator the process ends then. After receiving the meter data customer service imports the meter data to systems that require the information. The process of winning a new customer ends here.

Text 6: Process Description 2-2: Supplier Switch.

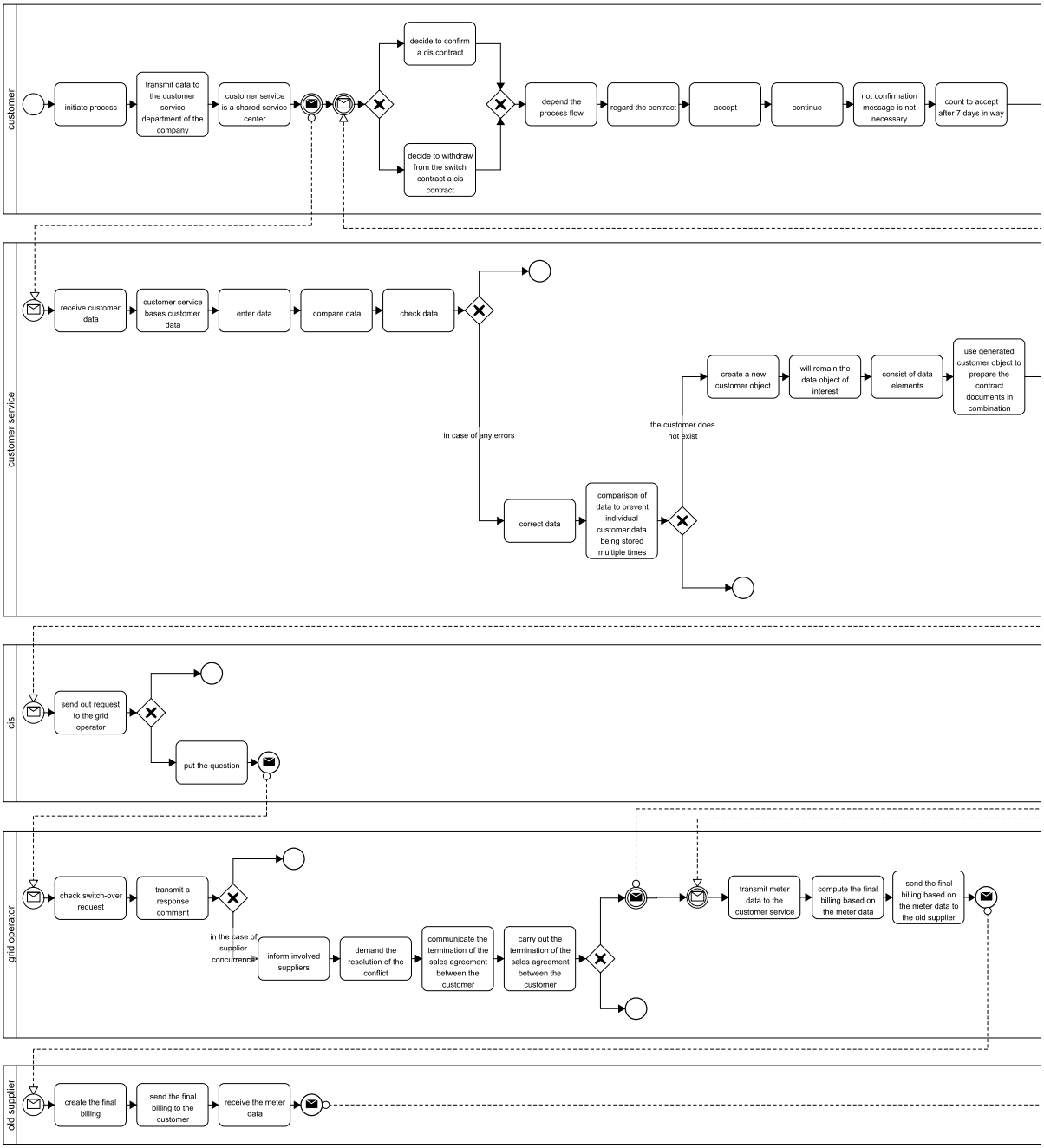

Figure B.33: Model 2-2 as generated by our system (part 1).

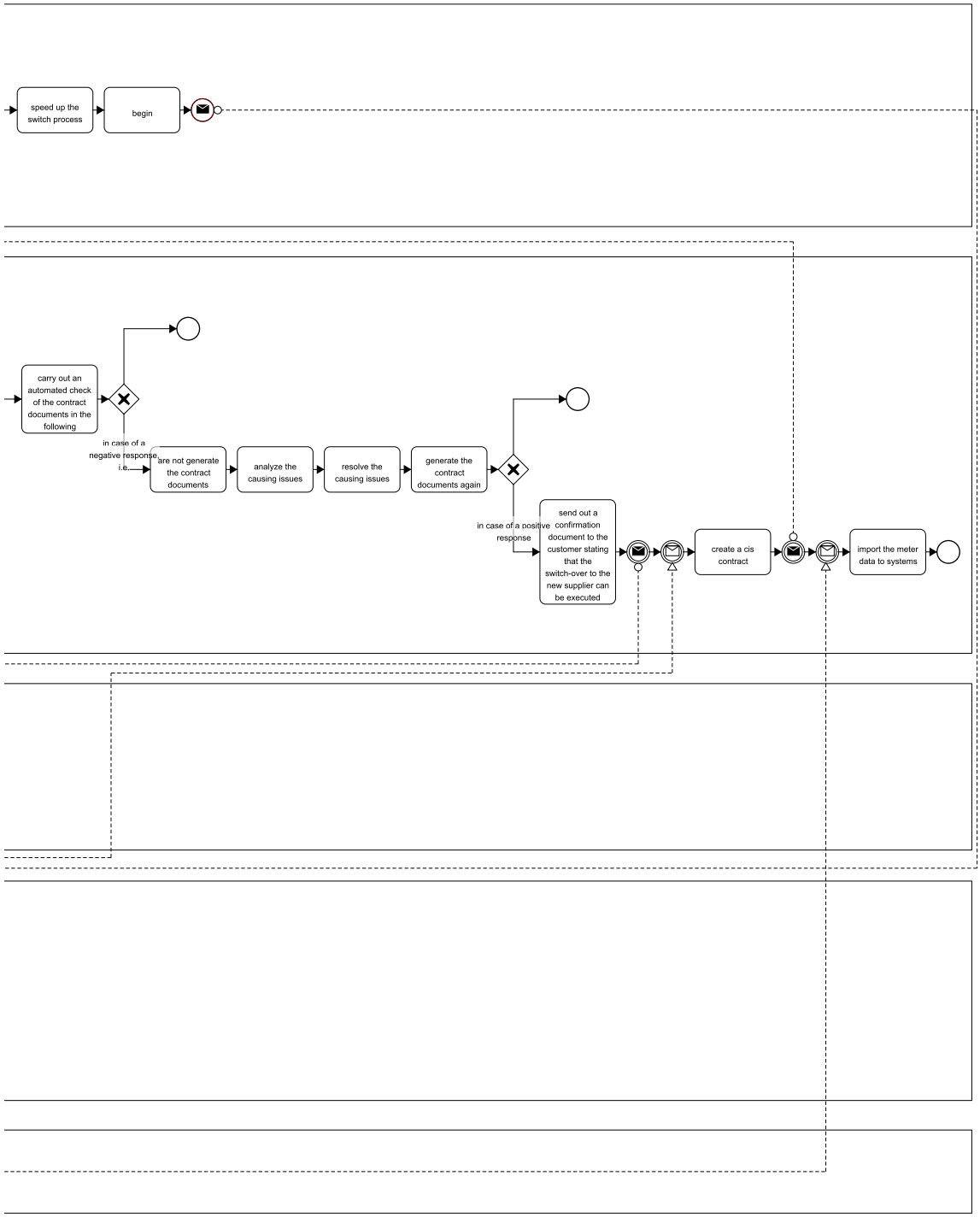

Figure B.34: Model 2-2 as generated by our system (part 2).

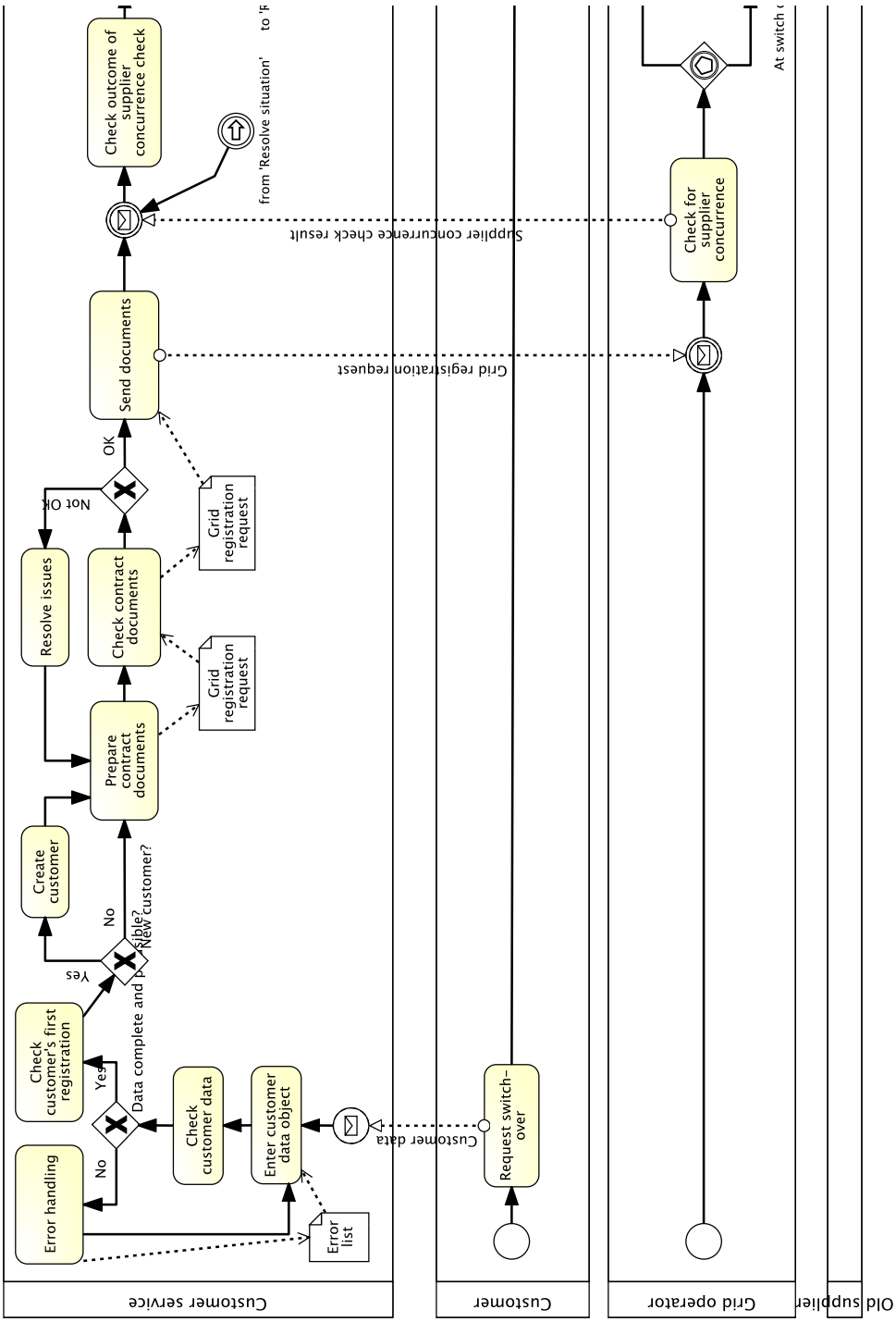

Figure B.35: Model 2-2 as created by a human modeler (part1).

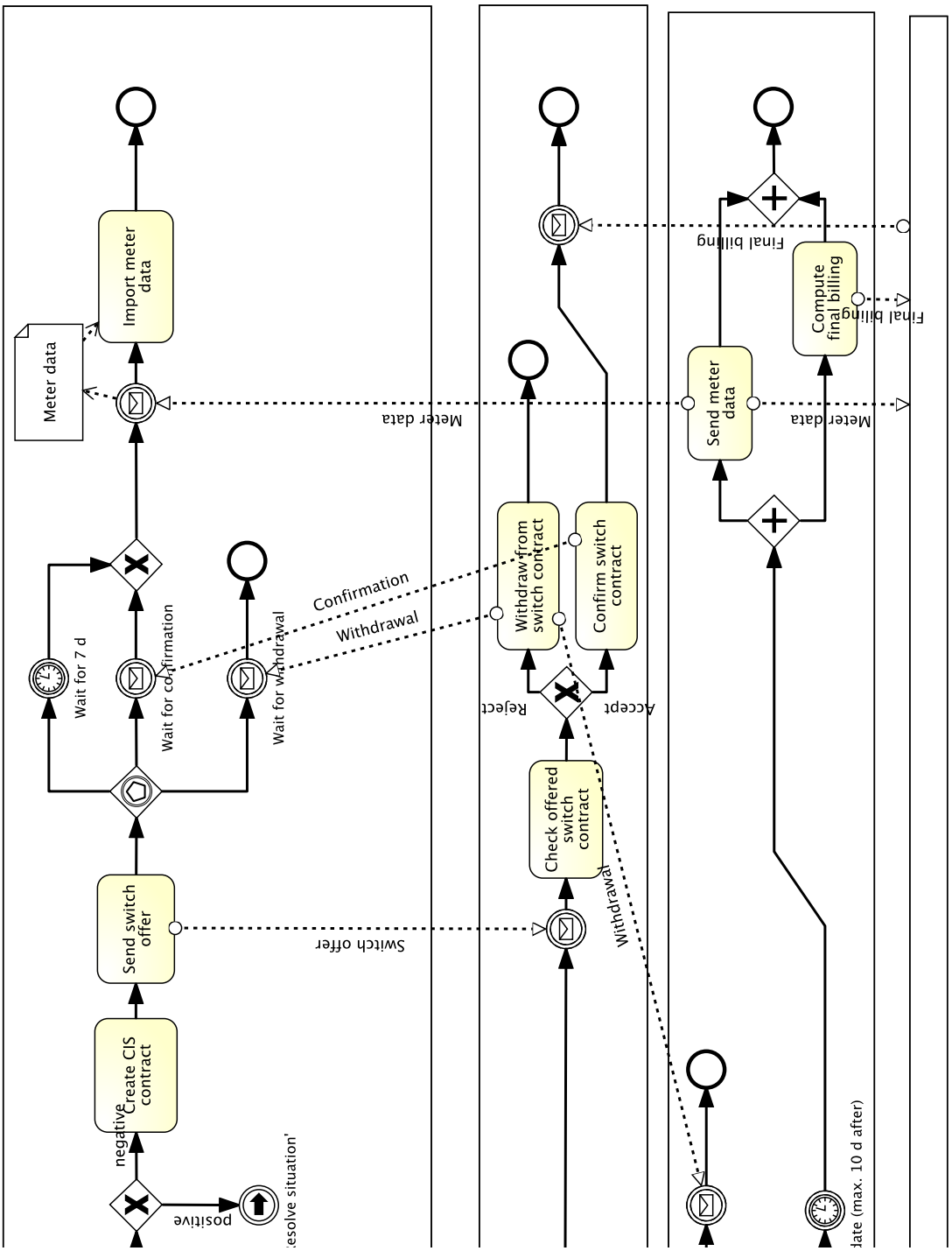

Figure B.36: Model 2-2 as created by a human modeler (part 2).

Appendix B.3. Models provided by the Queensland University of Technology

The party sends a warrant possession request asking a warrant to be released. The Client Service Back Office as part of the Small Claims Registry Operations receives the request and retrieves the SCT file. Then, the SCT Warrant Possession is forwarded to Queensland Police. The SCT physical file is stored by the Back Office awaiting a report to be sent by the Police. When the report is received, the respective SCT file is retrieved. Then, Back Office attaches the new SCT document, and stores the expanded SCT physical file. After that, some other MC internal staff receives the physical SCT file (out of scope).

Text 7: Process Description 3-1: 2009-1 MC Finalise SCT Warrant Posession.

Each morning, the files which have yet to be processed need to be checked, to make sure they are in order for the court hearing that day. If some files are missing, a search is initiated, otherwise the files can be physically tracked to the intended location. Once all the files are ready, these are handed to the Associate, and meantime the Judges Lawlist is distributed to the relevant people. Afterwards, the directions hearings are conducted.

Text 8: Process Description 3-2: 2009-2 Conduct Directions Hearing.

After a claim is registered, it is examined by a claims officer. The claims officer then writes a "settlement recommendation". This recommendation is then checked by a senior claims officer who may mark the claim as "OK" or "Not OK". If the claim is marked as "Not OK", it is sent back to the claims officer and the recommendation is repeated. If the claim is OK, the claim handling process proceeds.

Text 9: Process Description 3-3: 2009-3 Repetition - Cycles.

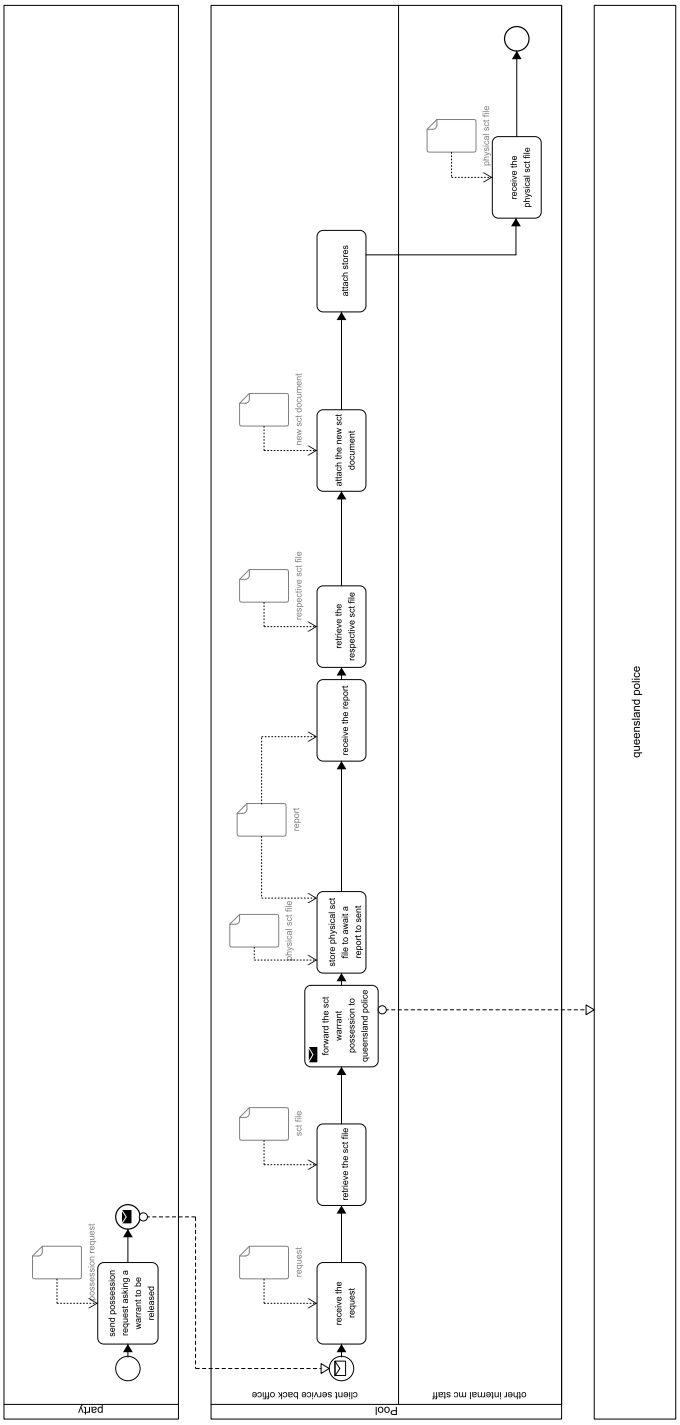

Figure B.37: Model 3-1 as generated by our system.

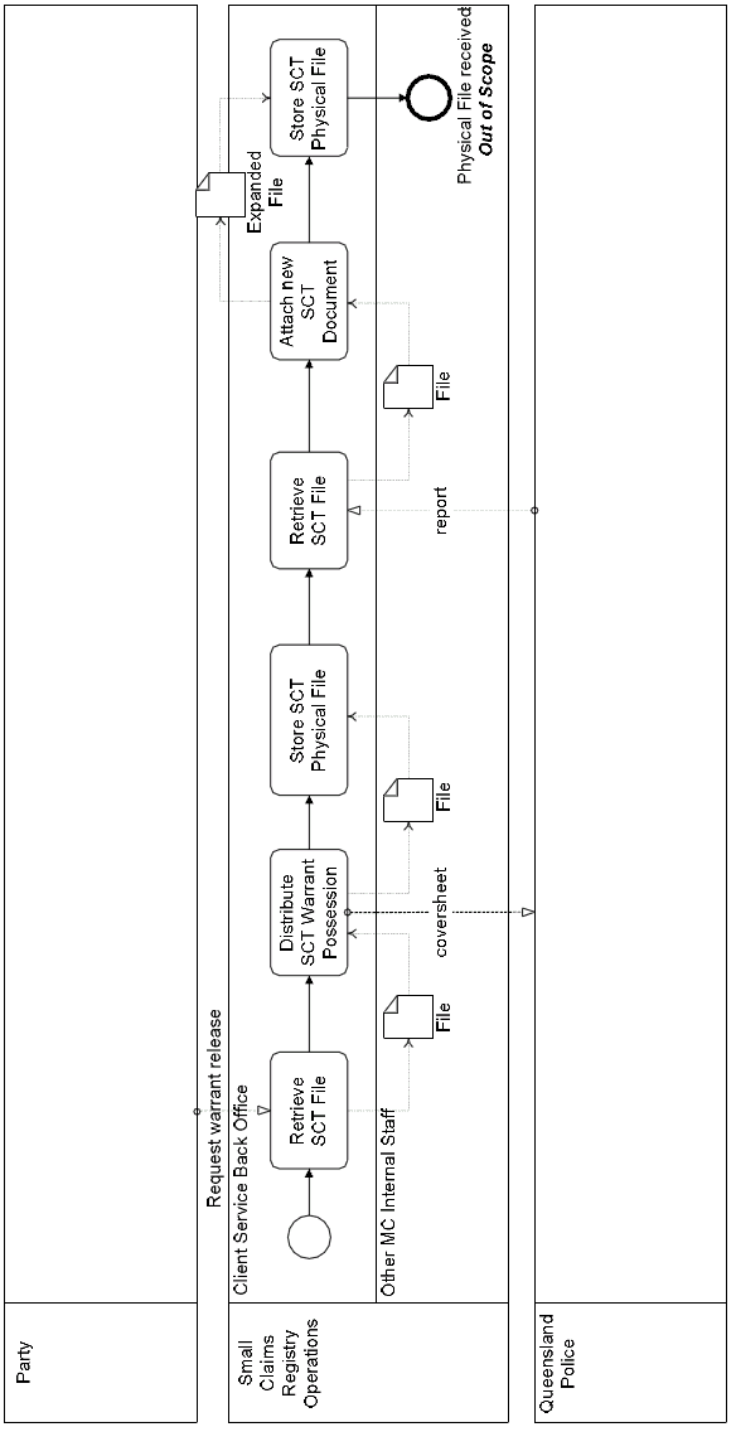

Figure B.38: Model 3-1 as created by a human modeler.

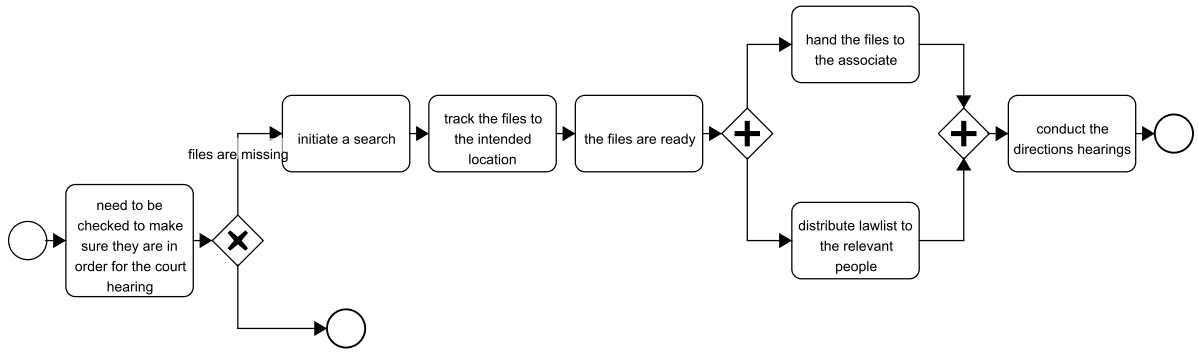

Figure B.39: Model 3-2 as generated by our system.

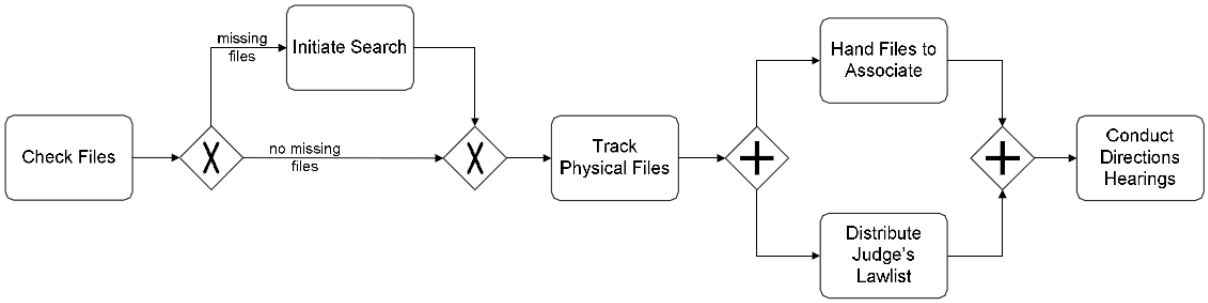

Figure B.40: Model 3-2 as created by a human modeler.

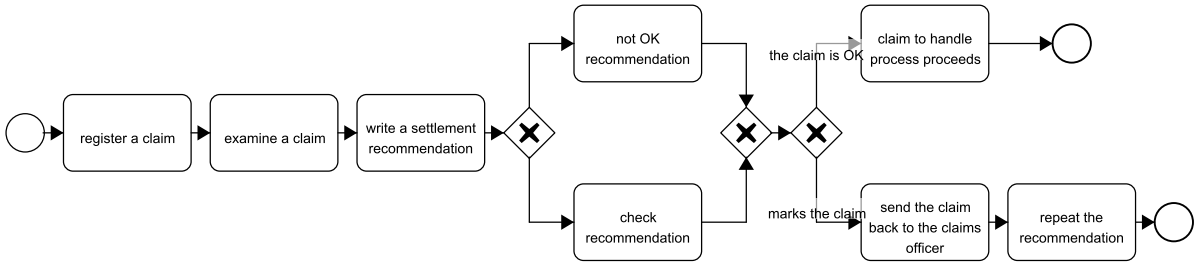

Figure B.41: Model 3-3 as generated by our system.

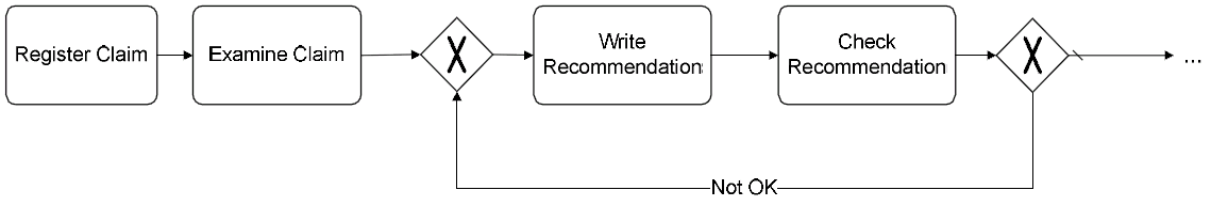

Figure B.42: Model 3-3 as created by a human modeler.

In the context of a claim handling process, it is sometimes necessary to send a questionnaire to the claimant to gather additional information. The claimant is expected to return the questionnaire within five days. If no response is received after five days, a reminder is sent to the claimant. If after another five days there is still no response, another reminder is sent and so on until the completed questionnaire is received.

Text 10: Process Description 3-4: 2009-4 Event-based Gateways.

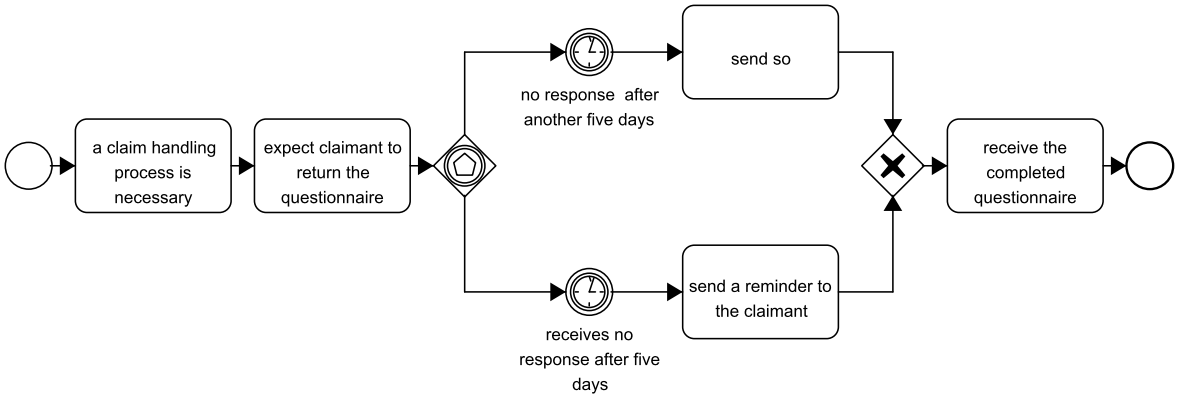

Figure B.43: Model 3-4 as generated by our system.

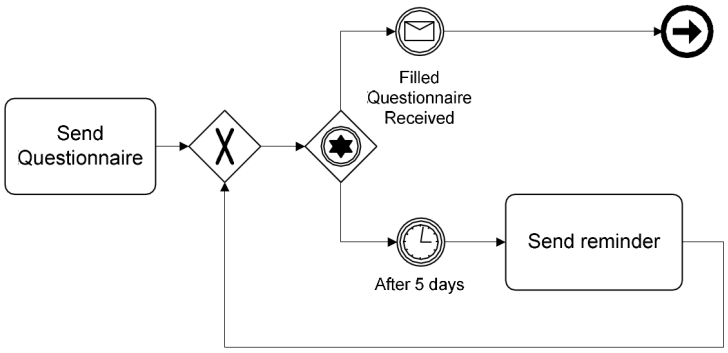

Figure B.44: Model 3-4 as created by a human modeler.

Mail from the party is collected on a daily basis by the Mail Processing Unit. Within this unit, the Mail Clerk sorts the unopened mail into the various business areas. The mail is then distributed. When the mail is received by the Registry, it is opened and sorted into groups for distribution, and thus registered in a manual incoming Mail Register. Afterwards, the Assistant Registry Manager within the Registry performs a quality check. If the mail is not compliant, a list of requisition explaining the reason for rejection is compiled and sent back to the party. Otherwise, the matter details (types of action) are captured and provided to the Cashier, who takes the applicable fees attached to the mail. At this point, the Assistant Registry Manager puts the receipt and copied documents into an envelope and posts it to the party. Meantime, the Cashier captures the Party Details and prints the Physical Court File.

Text 11: Process Description 3-5: 2009-5 P&E - Lodge Originating Document by Post.

When a claim is received, it is first checked whether the claimant is insured by the organization. If not, the claimant is informed that the claim must be rejected. Otherwise, the severity of the claim is evaluated. Based on the outcome (simple or complex claims), relevant forms are sent to the claimant. Once the forms are returned, they are checked for completeness. If the forms provide all relevant details, the claim is registered in the Claims Management system, which ends the Claims Notification process. Otherwise, the claimant is informed to update the forms. Upon reception of the updated forms, they are checked again.

Text 12: Process Description 3-6: 2010-1 Claims Notification.

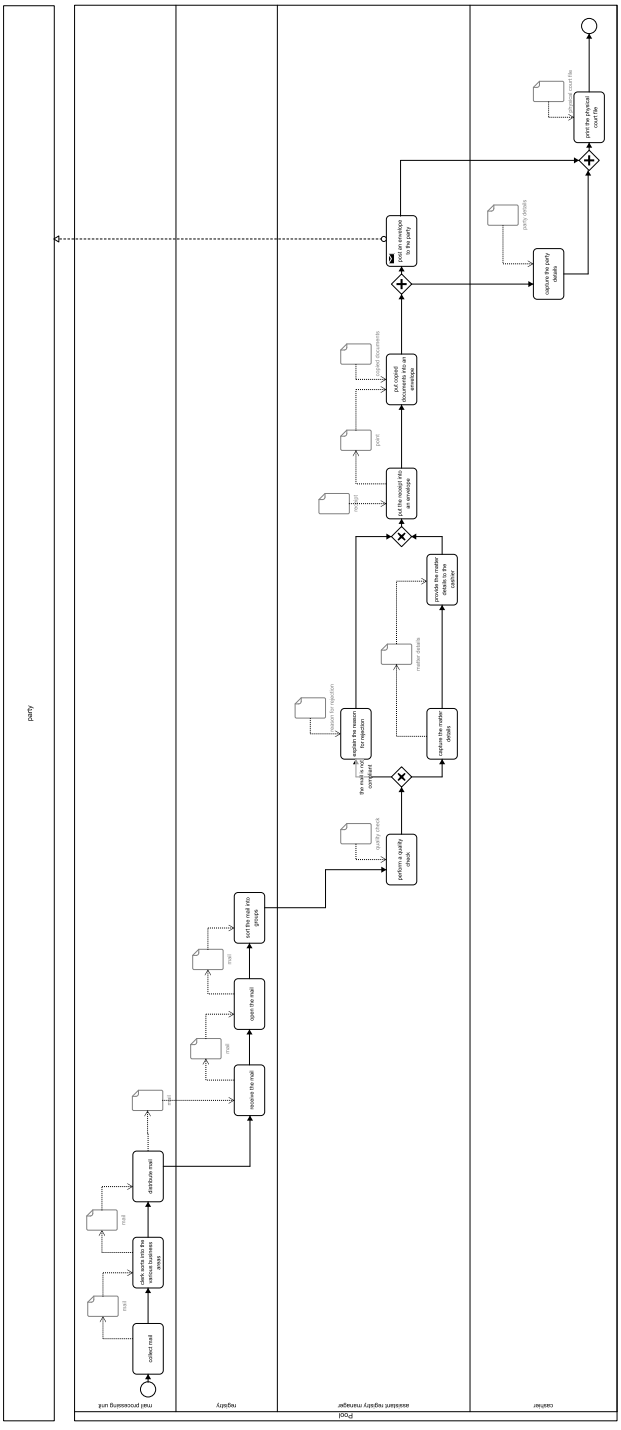

Figure B.45: Model 3-5 as generated by our system.

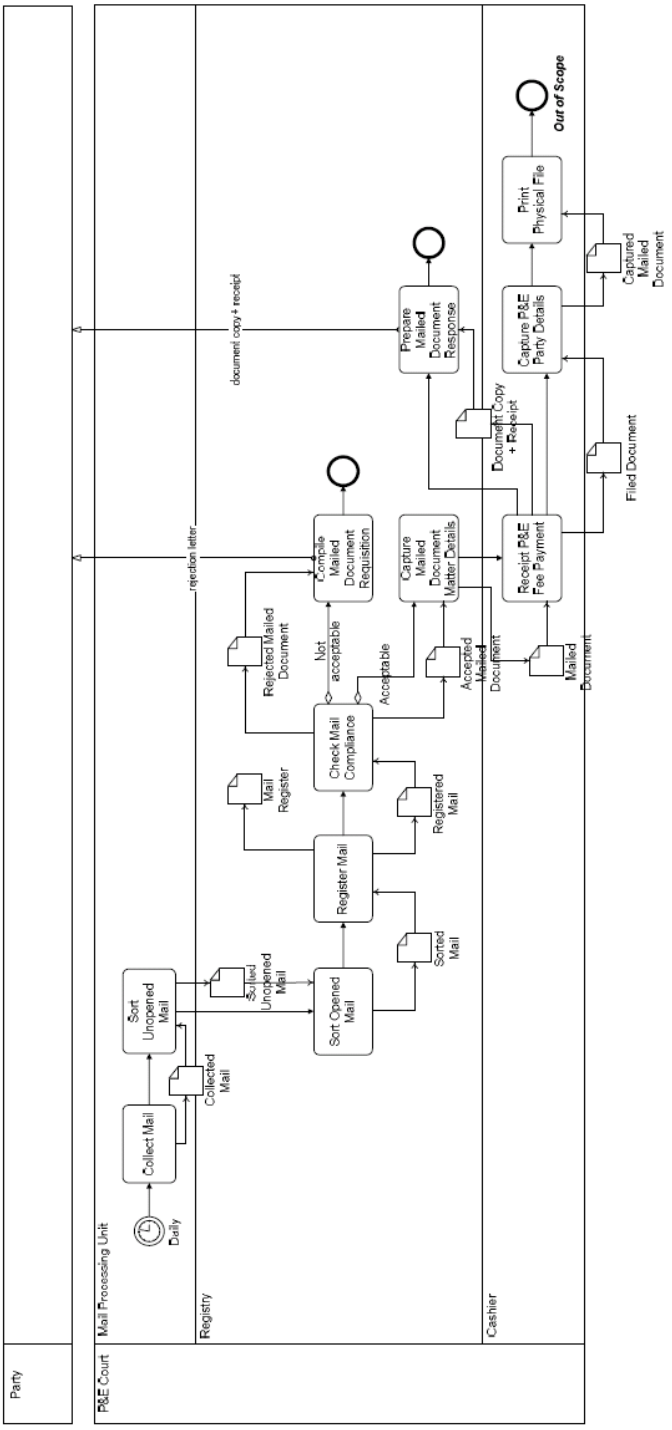

Figure B.46: Model 3-5 as created by a human modeler.

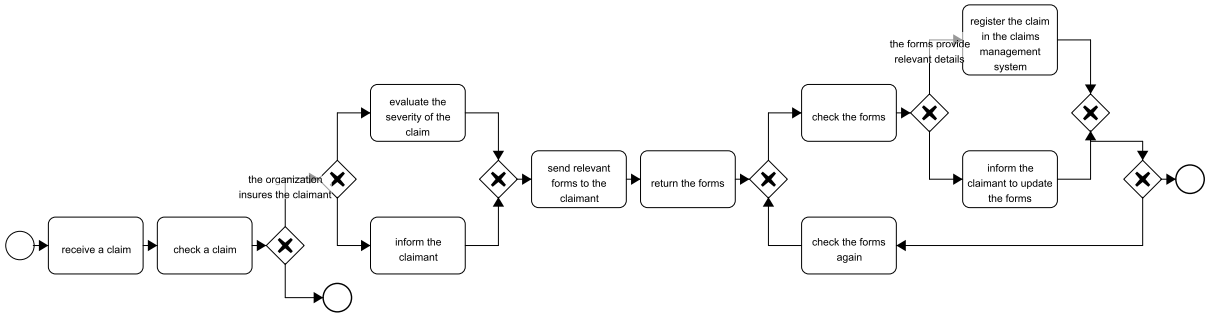

Figure B.47: Model 3-6 as generated by our system.

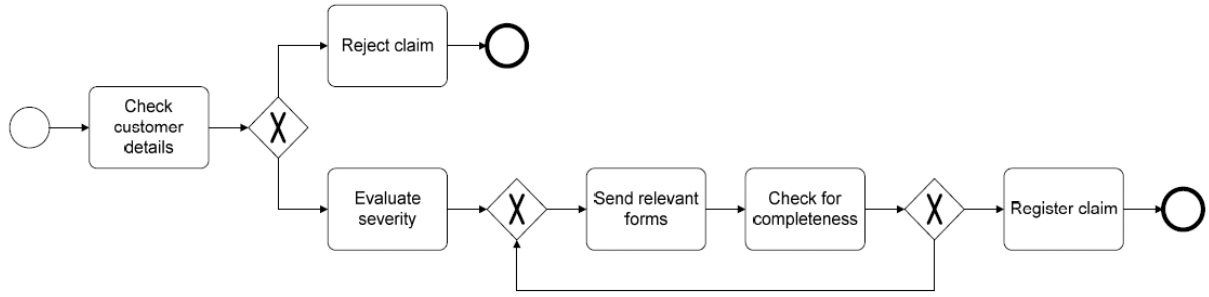

Figure B.48: Model 3-6 as created by a human modeler.

The Police Report related to the car accident is searched within the Police Report database and put in a file together with the Claim Documentation. This file serves as input to a claims handler who calculates an initial claim estimate. Then, the claims handler creates an Action Plan based on an Action Plan Checklist available in the Document Management system. Based on the Action Plan, a claims manager tries to negotiate a settlement on the claim estimate. The claimant is informed of the outcome, which ends the process.

Text 13: Process Description 3-7: 2010-2 Claims Creation.

The Police Report related to the car accident is searched within the Police Report database and put in a file together with the Claim Documentation. This file serves as input to a claims handler who calculates an initial claim estimate. Then, the claims handler creates an Action Plan based on an Action Plan Checklist available in the Document Management system. Based on the Action Plan, a claims manager tries to negotiate a settlement on the claim estimate. The claimant is informed of the outcome, which ends the process.

Text 14: Process Description 3-8: 2010-3 Claims Handling Process.

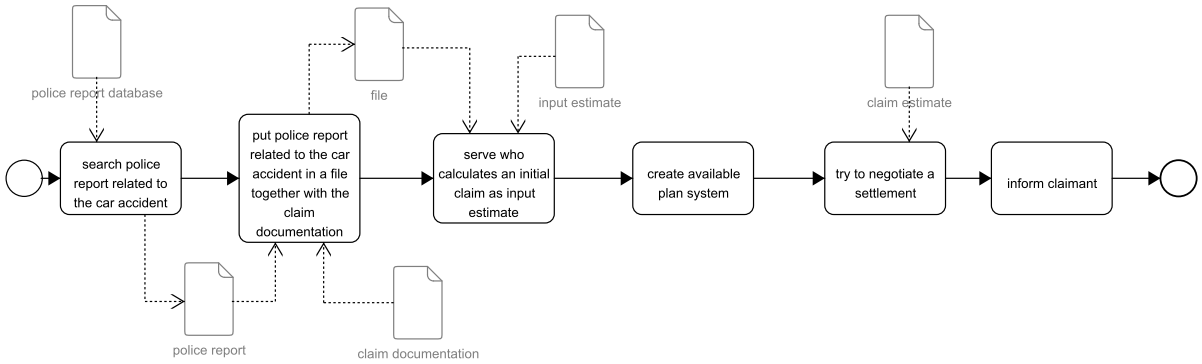

Figure B.49: Model 3-7 as generated by our system.

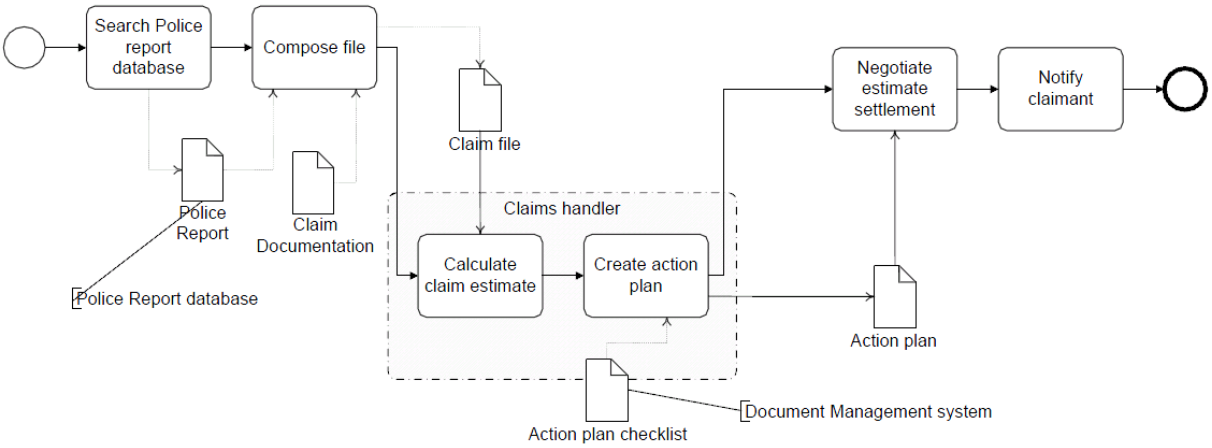

Figure B.50: Model 3-7 as created by a human modeler.

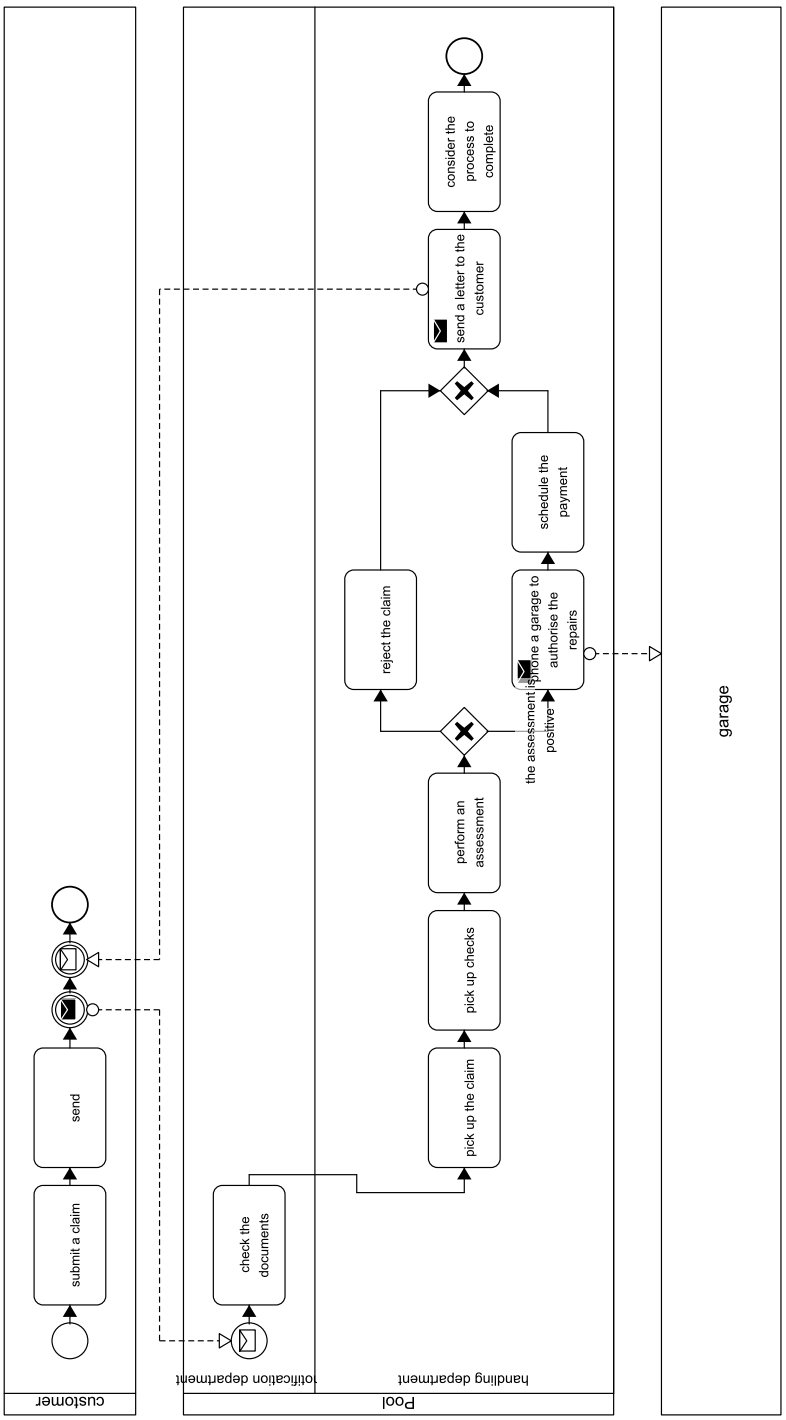

Figure B.51: Model 3-8 as generated by our system.

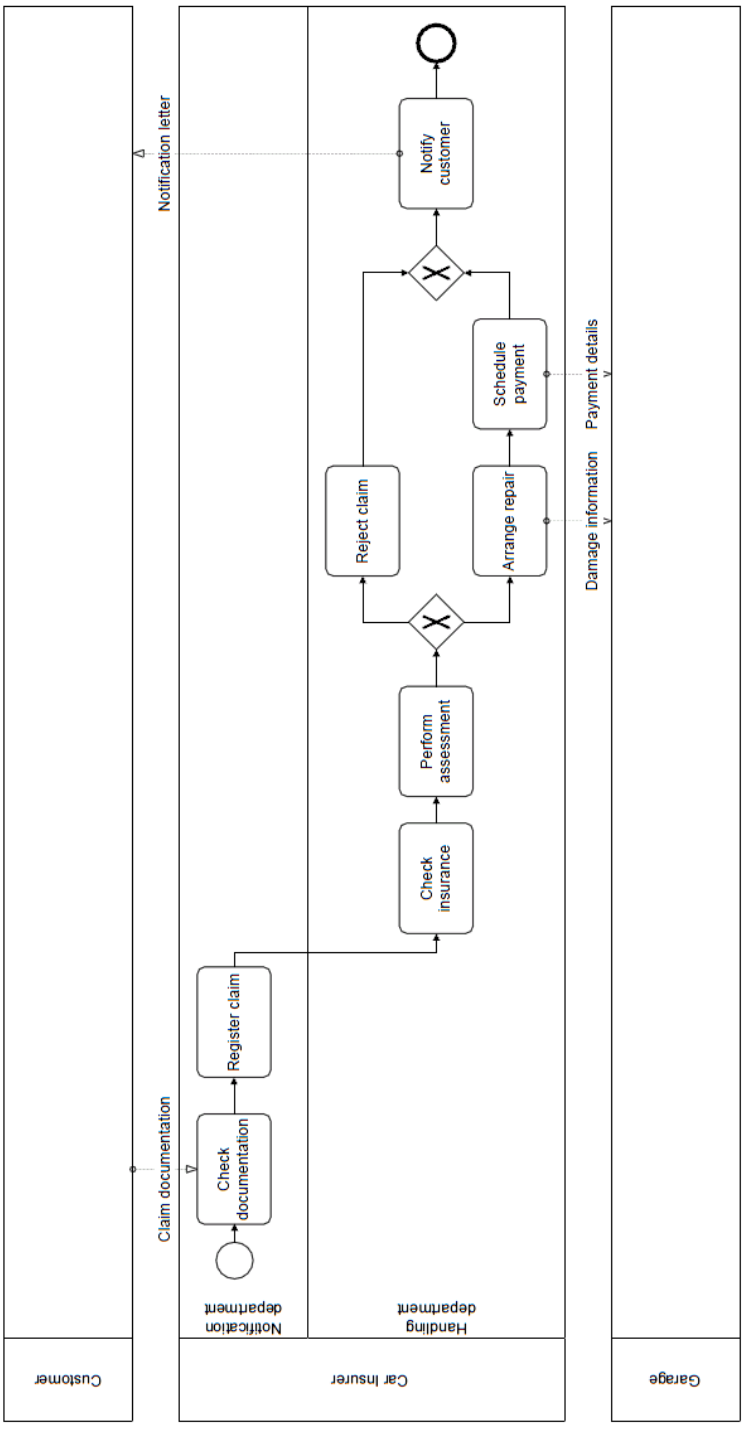

Figure B.52: Model 3-8 as created by a human modeler.

Appendix B.4. Models provided by the Technische Universiteit Eindhoven

The intake workflow starts with a notice by telephone at the secretarial office of the mental health care institute. This notice is done by the family doctor of somebody who is in need of mental treatment. The secretarial worker inquires after the name and residence of the patient. On the basis of this information, the doctor is put through to the nursing officer responsible for the part of the region that the patient lives in. The nursing officer makes a full inquiry into the mental, health, and social state of the patient in question. This information is recorded on a registration form. At the end of the conversation, this form is handed in at the secretarial office of the institute. Here, the information on the form is stored in the information system and subsequently printed. For new patients, a patient file is created. The registration form as well as the print from the information system are stored in the patient file. Patient files are kept at the secretarial office and may not leave the building. At the secretarial office, two registration cards are produced for respectively the future first and second intaker of the patient. The registration card contains a set of basic patient data. The new patient is added on the list of new notices. Halfway the week, at Wednesday, a staff meeting of the entire medical team takes place. The medical team consists of social-medical workers, physicians, and a psychiatrist. At this meeting, the team-leader assigns all new patients on the list of new notices to members of the team. Each patient will be assigned to a social-medical worker, who will act as the first intaker of the patient. One of the physicians will act as the second intaker. In assigning intakers, the teamleader takes into account their expertise, the region they are responsible for, earlier contacts they might have had with the patient, and their case load. The assignments are recorded on an assignment list which is handed to the secretarial office. For each new assignment, it is also determined whether the medical file of the patient is required. This information is added to the assignment list. The secretarial office stores the assignment of each patient of the assignment list in the information system. It passes the produced registration cards to the first and second intaker of each newly assigned patient. An intaker keeps this registration with him at times when visiting the patient and in his close proximity when he is at the office. For each patient for which the medical file is required, the secretarial office prepares and sends a letter to the family doctor of the patient, requesting for a copy of the medical file. As soon as this copy is received, the secretarial office will inform the second intaker and add the copy to the patient file. The first intaker plans a meeting with the patient as soon as this is possible. During the first meeting, the patient is examined using a standard checklist which is filled out. Additional observations are registered in a personal notebook. After a visit, the first intaker puts a copy of these notes in the file of a patient. The standard checklist is also added to the patient's file. The second intaker plans the first meeting only after the medical information of the physician if required has been received. Physicians use dictaphones to record their observations made during meetings with patients. The secretarial office types out these tapes, after which the information is added to the patient file. As soon as the meetings of the first and second intaker with the patient have taken place, the secretarial office puts the patient on the list of patients that reach this status. For the staff meeting on Wednesday, they provide the team-leader with a list of these patients. For each of these patients, the first and second intaker together with the team-leader and the attending psychiatrist formulate a treatment plan. This treatment plan formally ends the intake procedure.

Text 15: Process Description 4-1: Intaker Workflow.

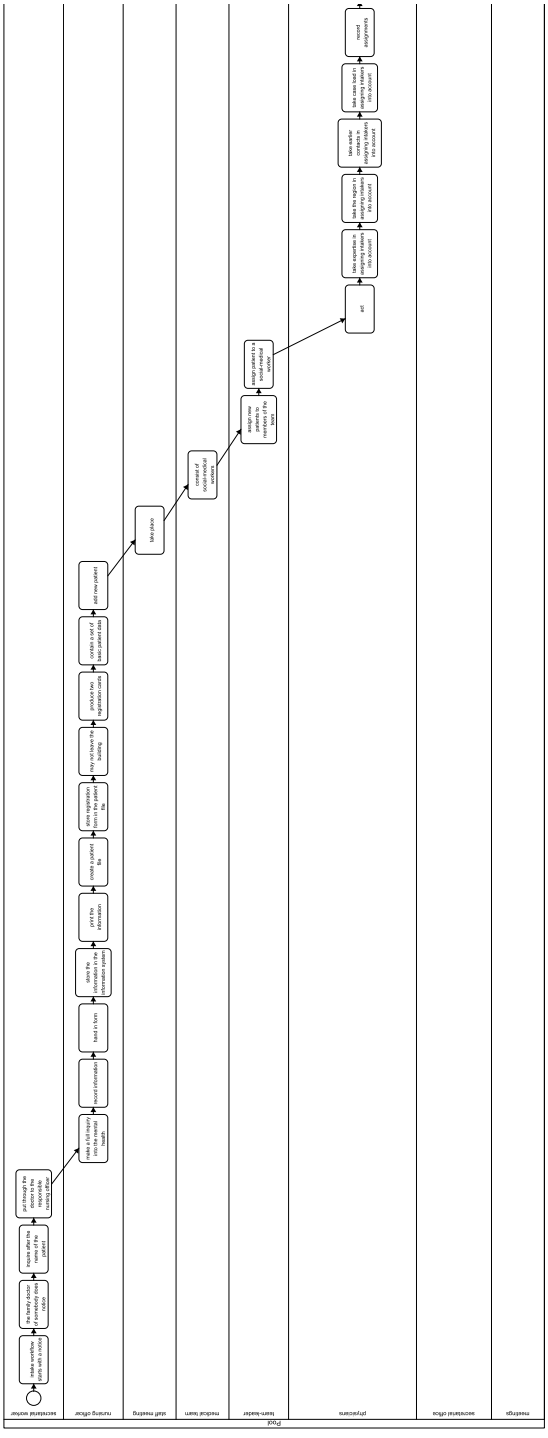

Figure B.53: Model 4-1 as generated by our system (part 1).

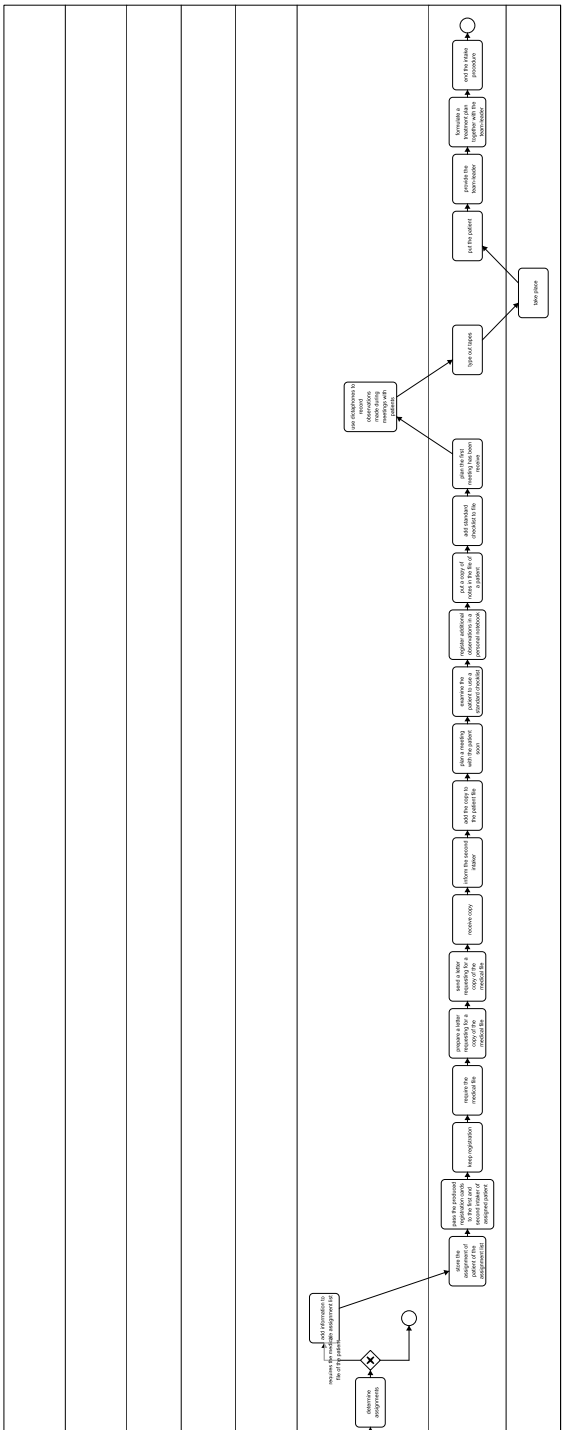

Figure B.54: Model 4-1 as generated by our system (part 2).

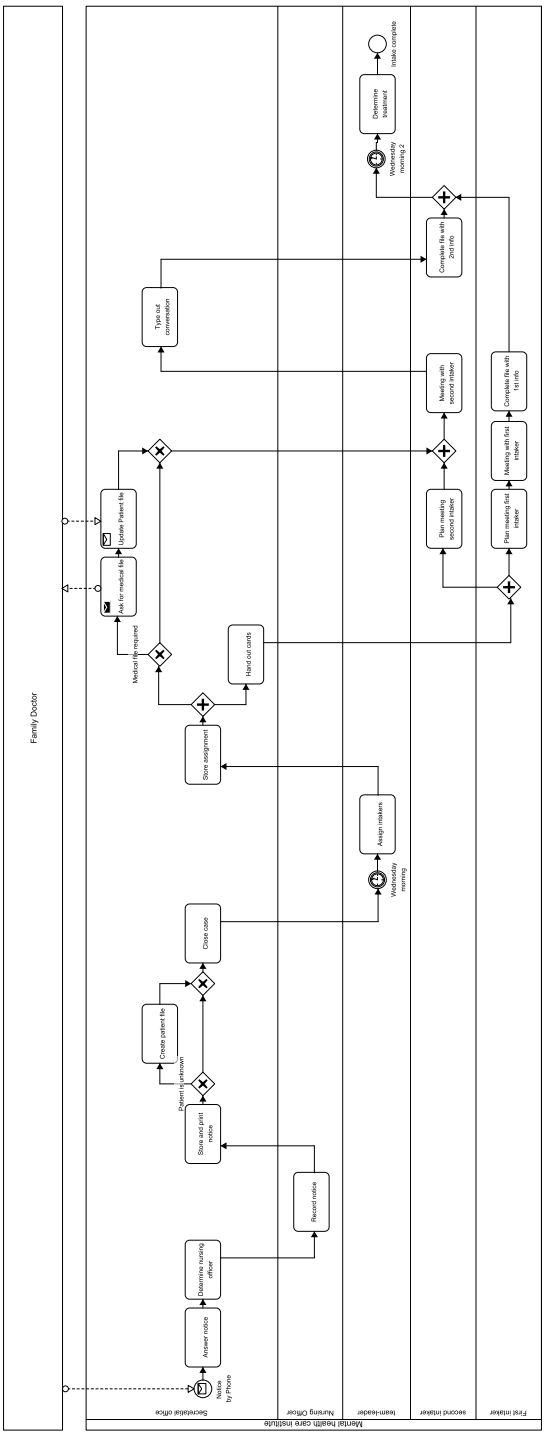

Figure B.55: Model 4-1 as created by a human modeler.

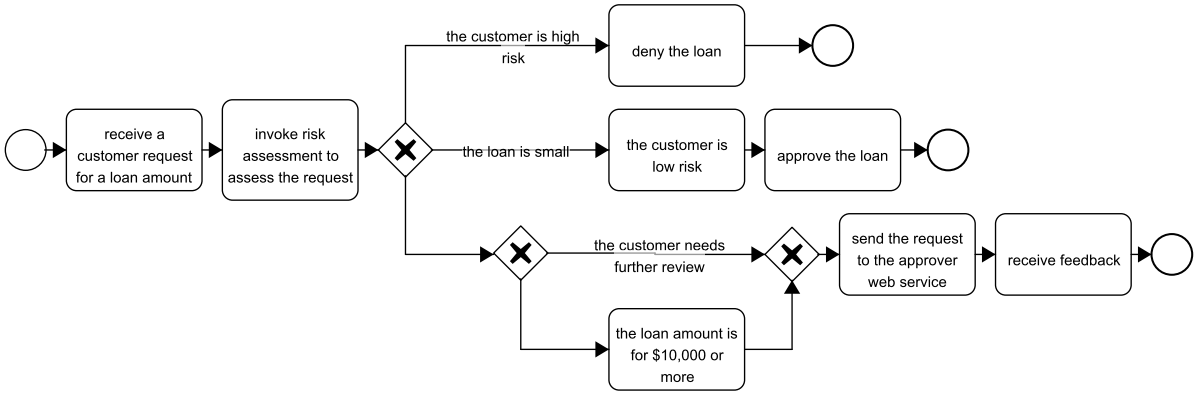

Figure B.56: Model 5-1 as generated by our system.

Appendix B.5. Models taken from BPM Vendor Tutorials

The loan approval process starts by receiving a customer request for a loan amount. The risk assessment Web service is invoked to assess the request. If the loan is small and the customer is low risk, the loan is approved. If the customer is high risk, the loan is denied. If the customer needs further review or the loan amount is for $10,000 or more, the request is sent to the approver Web service. The customer receives feedback from the assessor or approver.

Text 16: Process Description 5-1: Active VOS Tutorial.

The process of Vacations Request starts when any employee of the organization submits a vacation request. Once the requirement is registered, the request is received by the immediate supervisor of the employee requesting the vacation. The supervisor must approve or reject the request. If the request is rejected, the application is returned to the applicant/employee who can review the rejection reasons. If the request is approved a notification is generated to the Human Resources Representative, who must complete the respective management procedures.

Text 17: Process Description 5-2: BizAgi Tutorial 1.

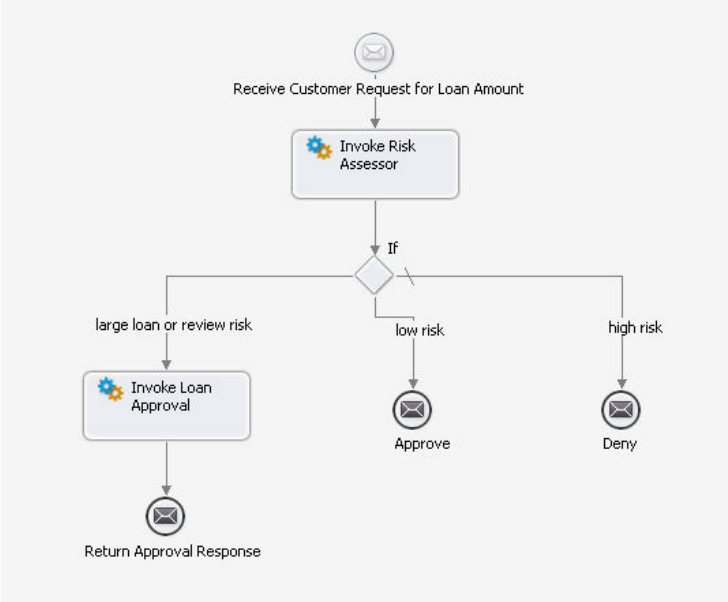

Figure B.57: Model 5-1 as created by a human modeler.

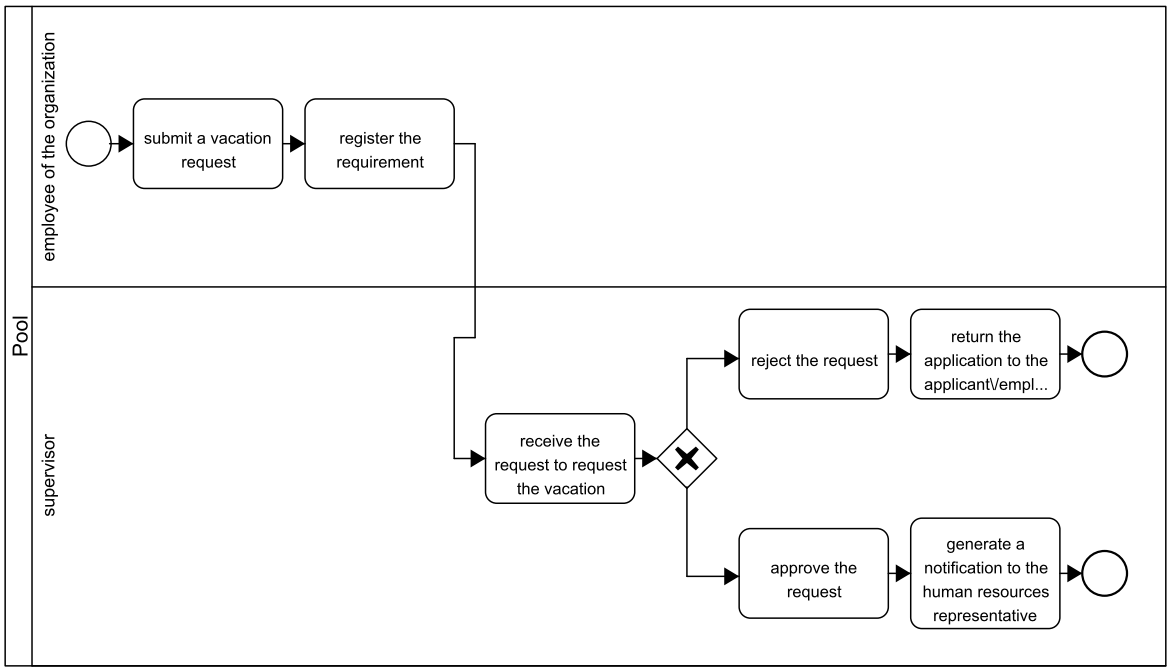

Figure B.58: Model 5-2 as generated by our system.

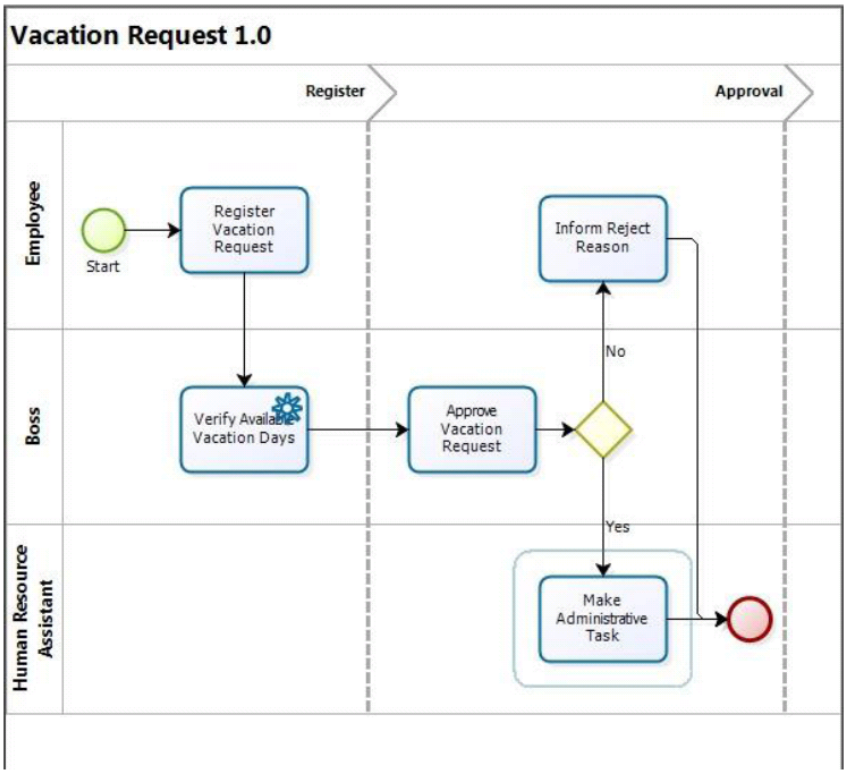

Figure B.59: Model 5-2 as created by a human modeler.

The process of an Office Supply Request starts when any employee of the organization submits an office supply request. Once the requirement is registered, the request is received by the immediate supervisor of the employee requesting the office supplies. The supervisor must approve or ask for a change, or reject the request. If the request is rejected the process will end. If the request is asked to make a change then it is returned to the petitioner/employee who can review the comments for the change request. If the request is approved it will go to the purchase department that will be in charge of making quotations (using a subprocess) and select a vendor. If the vendor is not valid in the system the purchase department will have to choose a different vendor. After a vendor is selected and confirmed, the system will generate and send a purchase order and wait for the product to be delivered and the invoice received. In any case the system will send a notification to let the user know what the result was. In any of the cases, approval, rejection or change required the system will send the user a notification.

Text 18: Process Description 5-3: BizAgi Tutorial 2.

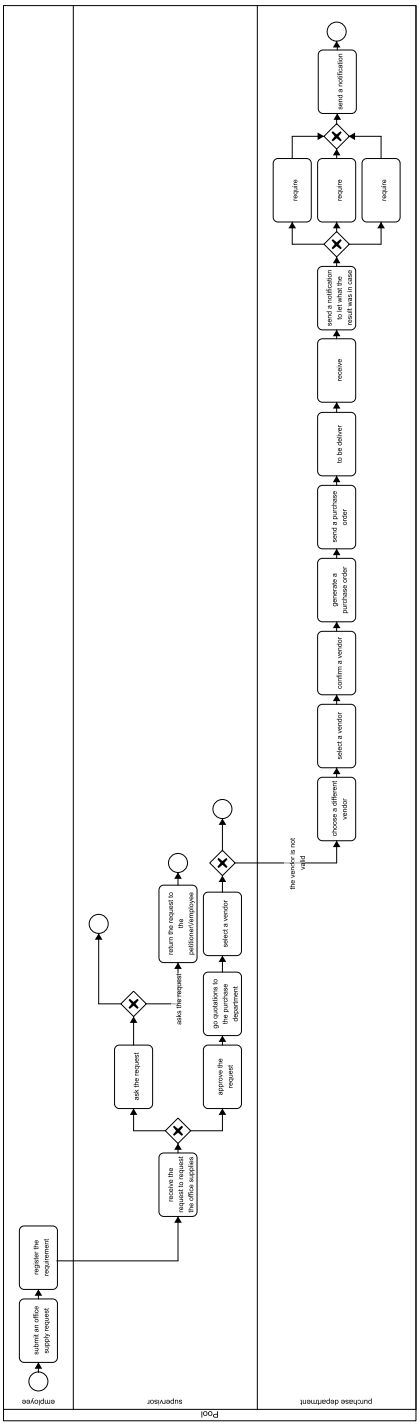

Figure B.60: Model 5-3 as generated by our system.

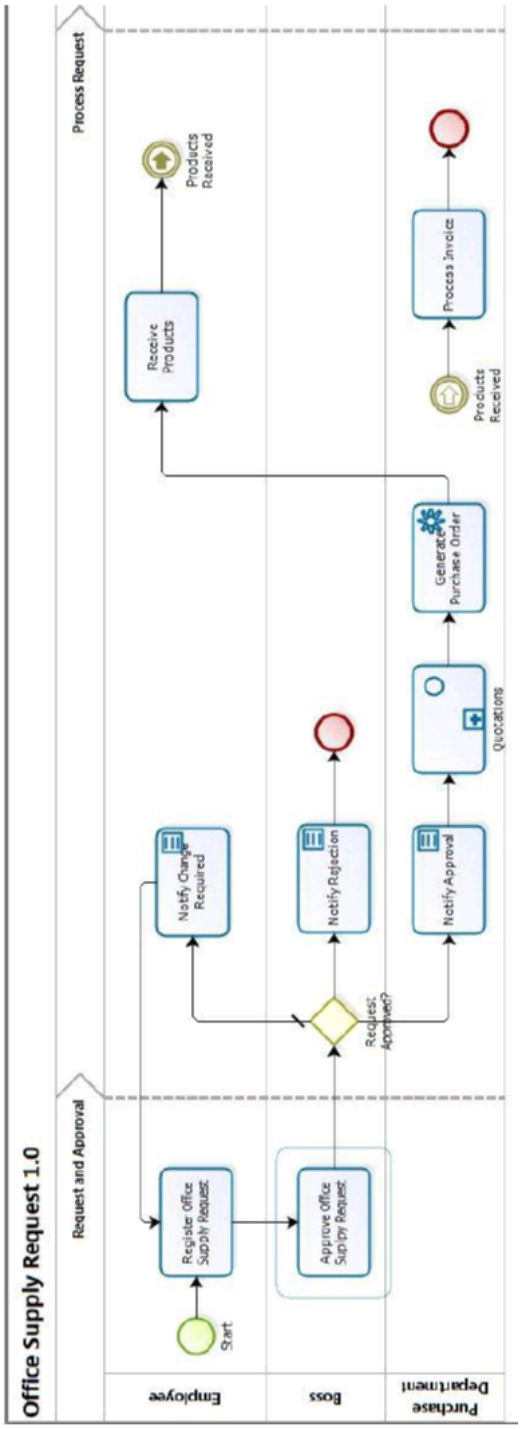

Figure B.61: Model 5-3 as created by a human modeler.

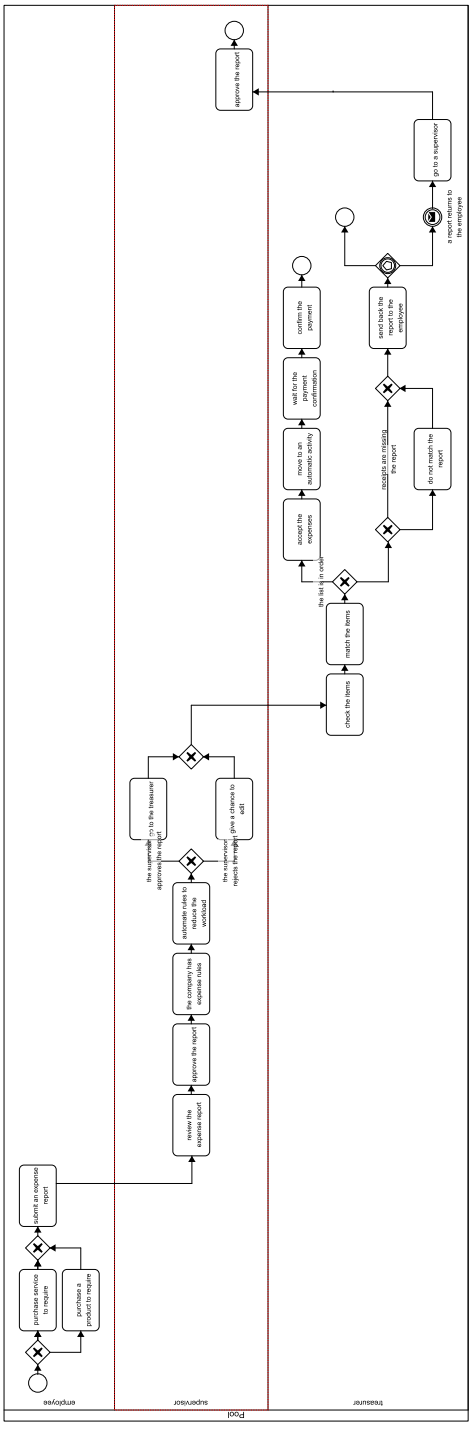

Figure B.62: Model 5-4 as generated by our system.

An employee purchases a product or service he requires. For instance, a sales person on a trip rents a car. The employee submits an expense report with a list of items, along with the receipts for each item. A supervisor reviews the expense report and approves or rejects the report. Since the company has expense rules, there are circumstances where the supervisor can accept or reject the report upon first inspection. These rules could be automated, to reduce the workload on the supervisor. If the supervisor rejects the report, the employee, who submitted it, is given a chance to edit it, for example to correct errors or better describe an expense. If the supervisor approves the report, it goes to the treasurer. The treasurer checks that all the receipts have been submitted and match the items on the list. If all is in order, the treasurer accepts the expenses for processing (including, e.g. , payment or refund, and accounting). If receipts are missing or do not match the report, he sends it back to the employee. If a report returns to the employee for corrections, it must again go to a supervisor, even if the supervisor previously approved the report. If the treasurer accepts the expenses for processing, the report moves to an automatic activity that links to a payment system. The process waits for the payment confirmation. After the payment is confirmed, the process ends.

Text 19: Process Description 5-4: Oracle Tutorial.

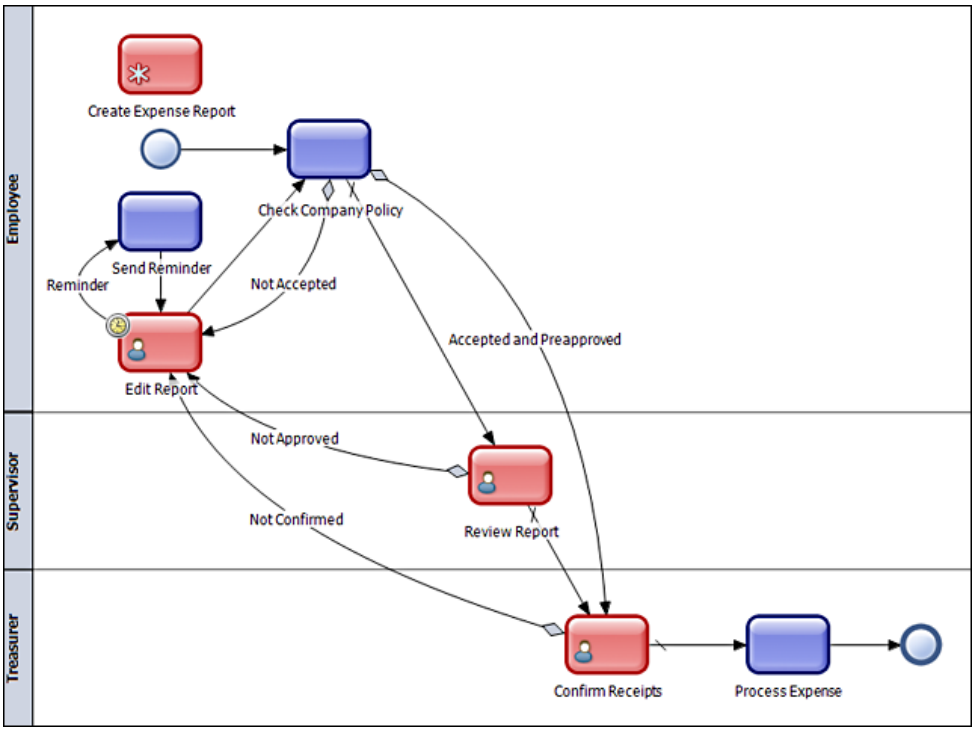

Figure B.63: Model 5-4 as created by a human modeler.

Appendix B.6. Models provided by the inubit AG

As a basic principle, ACME AG receives invoices on paper or fax. These are received by the Secretariat in the central inbox and forwarded after a short visual inspection to an accounting employee. In "ACME Financial Accounting", a software specially developed for the ACME AG, she identifies the charging suppliers and creates a new instance (invoice). She then checks the invoice items and notes the corresponding cost center at the ACME AG and the related cost center managers for each position on a separate form ("docket").The docket and the copy of the invoice go to the internal mail together and are sent to the first cost center manager to the list. He reviews the content for accuracy after receiving the copy of the invoice. Should everything be in order, he notes his code one on the docket ("accurate position - AP") and returns the copy of the invoice to the internal mail. From it, the copy of the invoice is passed on to the next cost center manager, based on the docket, or if all items are marked correct, sent back to accounting. Therefore, the copy of invoice and the docket gradually move through the hands of all cost center managers until all positions are marked as completely accurate. However, if inconsistencies exist, e.g. because the ordered product is not of the expected quantity or quality, the cost center manager rejects the AP with a note and explanatory statement on the docket, and the copy of the invoice is sent back to accounting directly. Based on the statements of the cost center managers, she will proceede with the clarification with the vendor, but, if necessary, she consults the cost center managers by telephone or e-mail again. When all inconsistencies are resolved, the copy of the invoice is sent to the cost center managers again, and the process continues. After all invoice items are AP, the accounting employee forwards the copy of the invoice to the commercial manager. He makes the commercial audit and issues the approval for payment. If the bill amount exceeds EUR 20,000, the Board wants to check it again (4-eyes-principle). The copy of the invoice including the docket moves back to the accounting employee in the appropriate signature file. Should there be a complaint during the commercial audit, it will be resolved by the accounting employee with the supplier. After the commercial audit is successfully completed, the accounting employee gives payment instructions and closes the instance in "ACME financial accounting".

Text 20: Process Description 6-1: ACME.

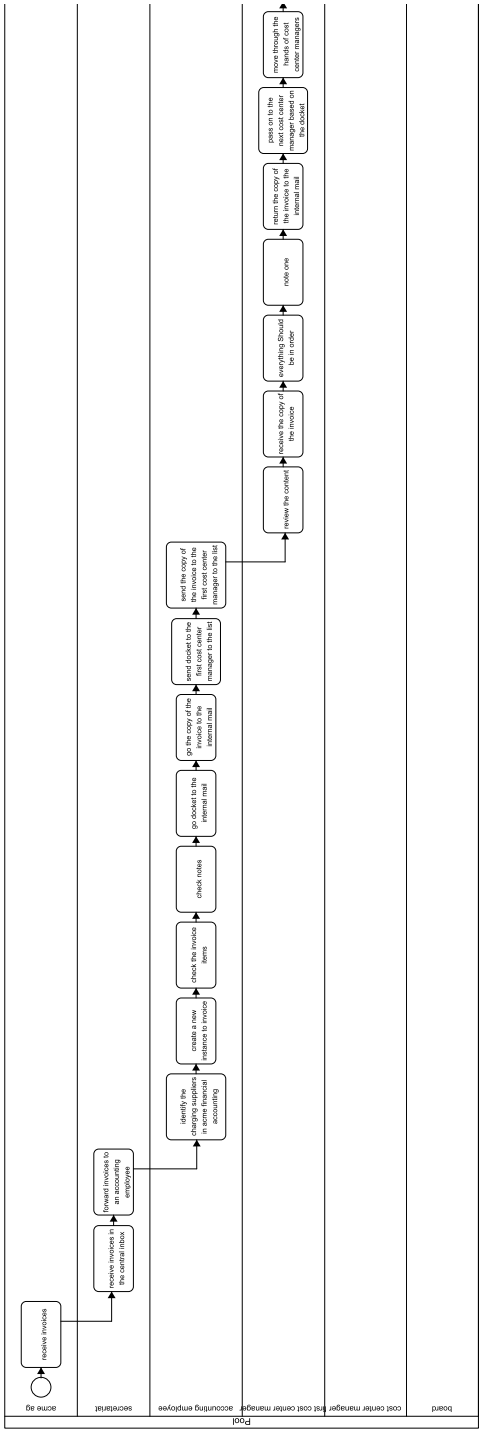

Figure B.64: Model 6-1 as generated by our system (part 1).

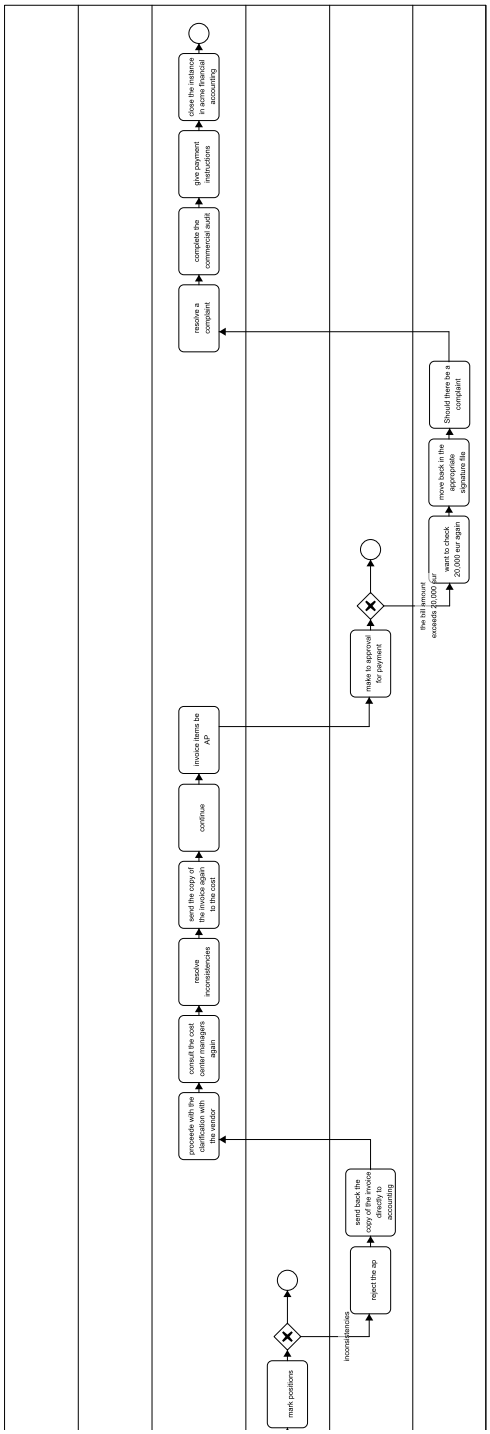

Figure B.65: Model 6-1 as generated by our system (part 2).

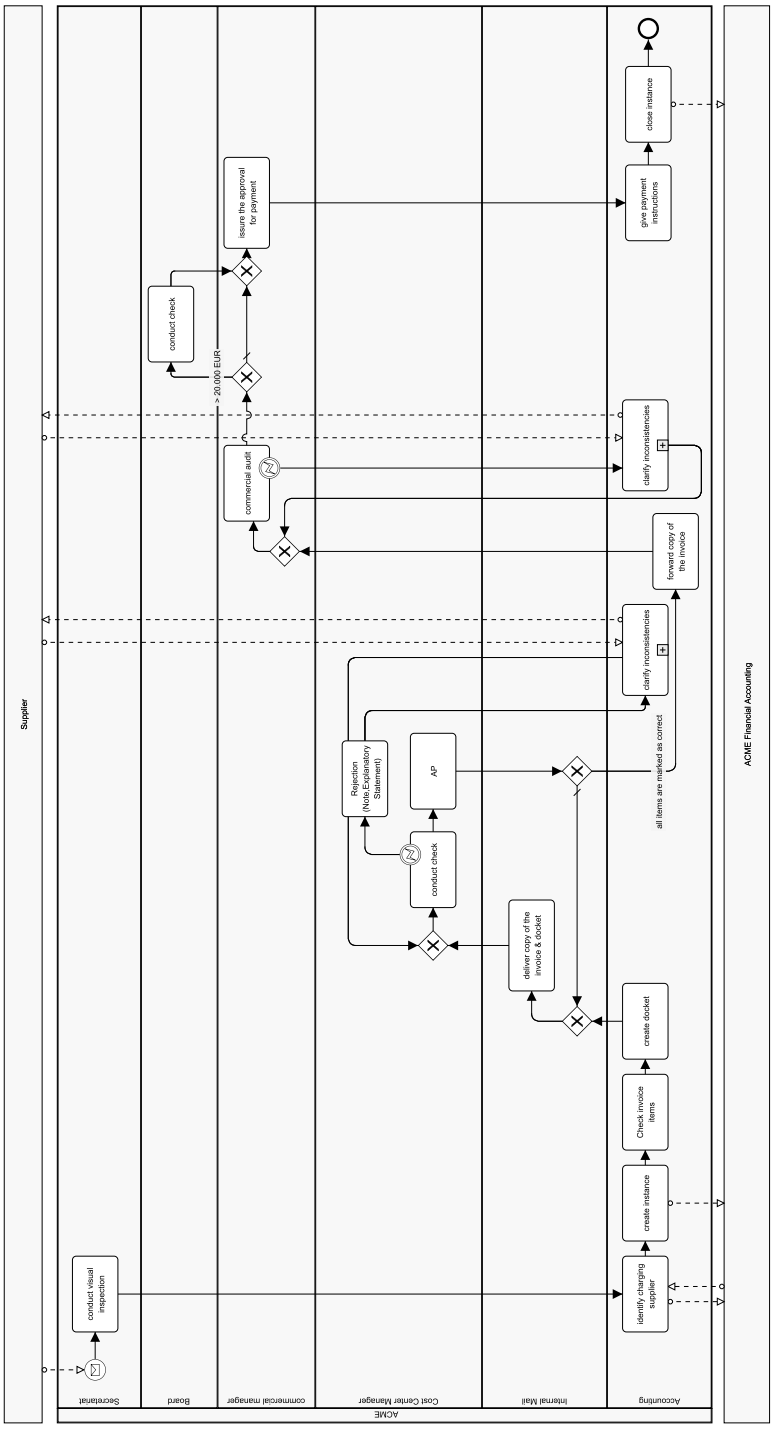

Figure B.66: Model 6-1 as created by a human modeler.

The process starts periodically on the first of each month, when Assembler AG places an order with the supplier in order to request more product parts. a) Assembler AG sends the order to the supplier. b) The supplier processes the order. c) The supplier sends an invoice to Assembler AG. d) Assembler AG receives the invoice.

Text 21: Process Description 6-2: inubit AG Tutorial.

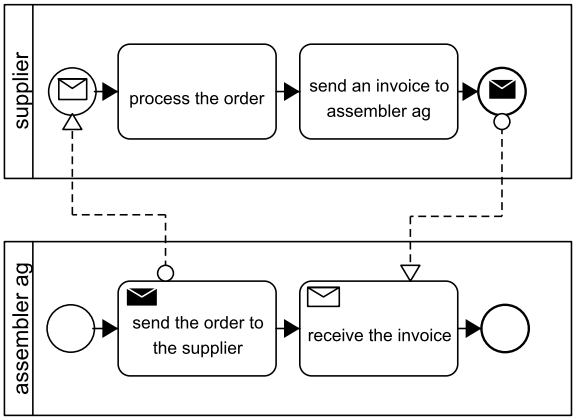

Figure B.67: Model 6-2 as generated by our system.

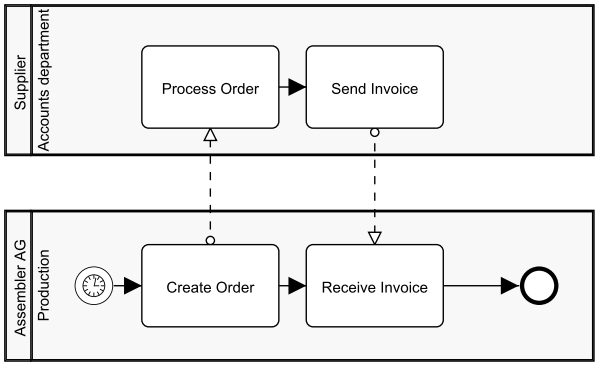

Figure B.68: Model 6-2 as created by a human modeler.

Every time we get a new order from the sales department, first, one of my masters determines the necessary parts and quantities as well as the delivery date. Once that information is present, it has to be entered into our production planning system (PPS). It optimizes our production processes and creates possibly uniform work packages so that the setup times are minimized. Besides, it creates a list of parts to be procured. Unfortunately it is not coupled correctly to our Enterprise Resource Planning system (ERP), so the data must be transferred manually. By the way, that is the second step. Once all the data is present, we need to decide whether any parts are missing and must be procured or if this is not necessary. Once production is scheduled to start, we receive a notice from the system and an employee takes care of the implementation.Finally, the order will be checked again for its quality.

The first step is to determine contact details of potential customers. This can be achieved in several ways. Sometimes, we buy details for cold calls, sometimes, our marketing staff participates in exhibitions and sometimes, you just happen to know somebody, who is interested in the product. Then we start calling the customer. That is done by the call center staff. They are determining the contact person and the budget which would be available for the project. Of course, asking the customer whether he is generally interested is also important. If this is not the case, we leave him alone, except if the potential project budget is huge. Then the head of development personally tries to acquire the customer. If the customer is interested in the end, the next step is a detailed online presentation. It is given either by a sales representative or by a pre-sales employee in case of a more technical presentation. Afterwards we are waiting for the customer to come back to us. If we are not contacted within 2 weeks, a sales representative is calling the customer. The last phase is the creation of a quotation.

Text 22: Process Description 6-4: Turbopixel.

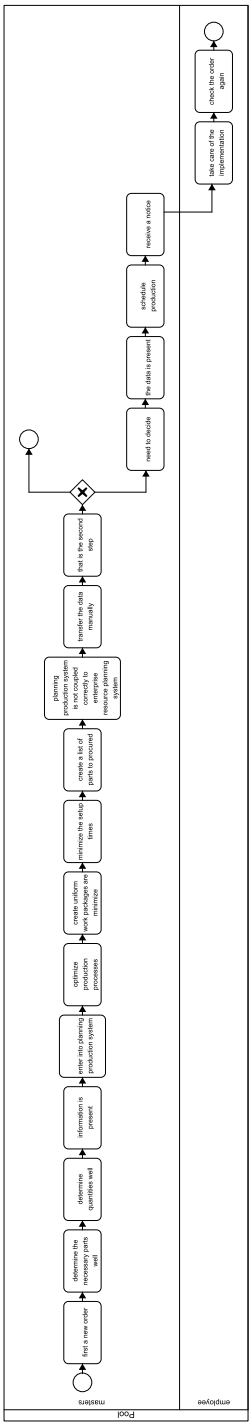

Figure B.69: Model 6-3 as generated by our system.

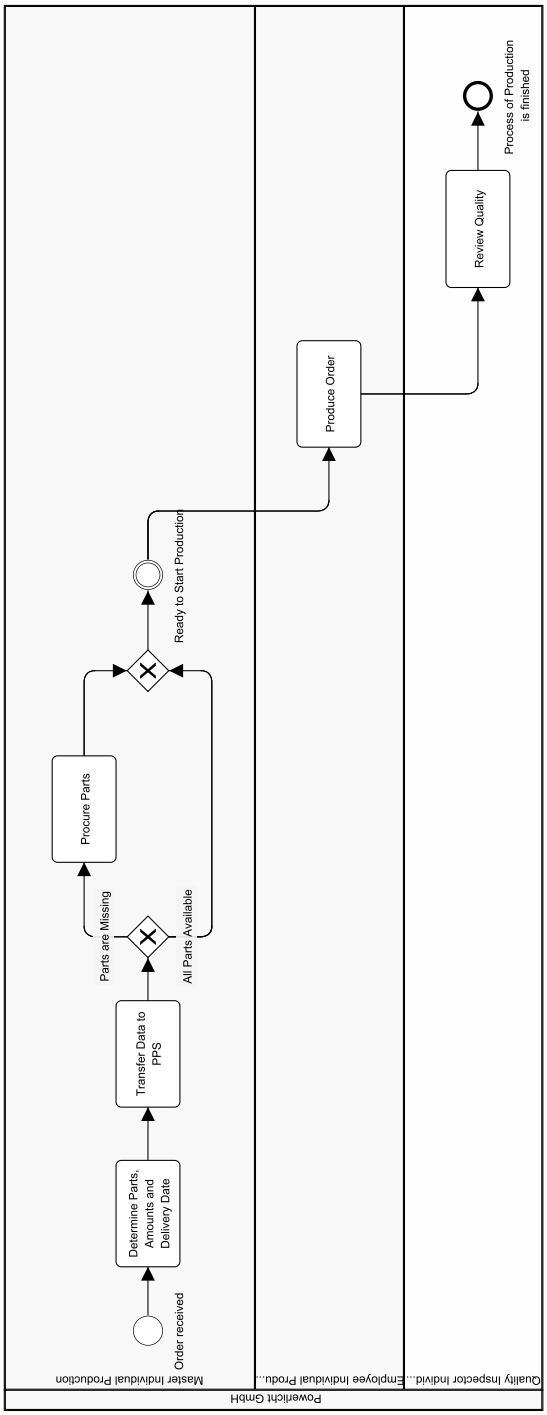

Figure B.70: Model 6-3 as created by a human modeler.

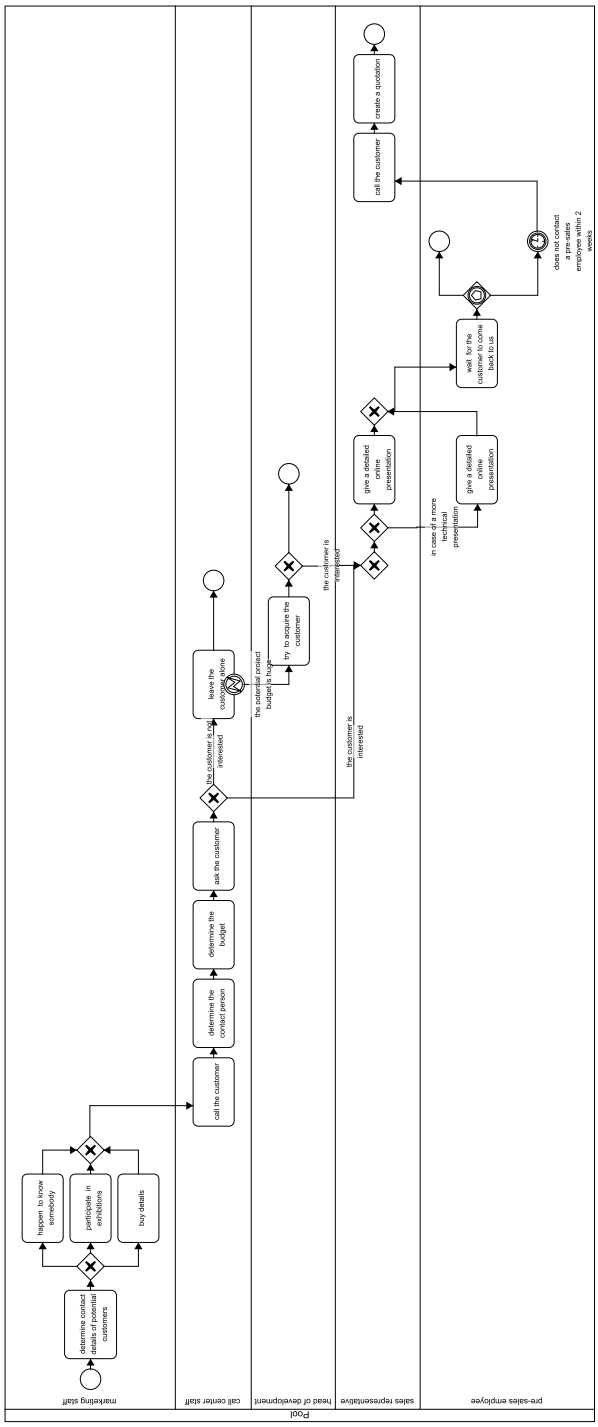

Figure B.71: Model 6-4 as generated by our system.

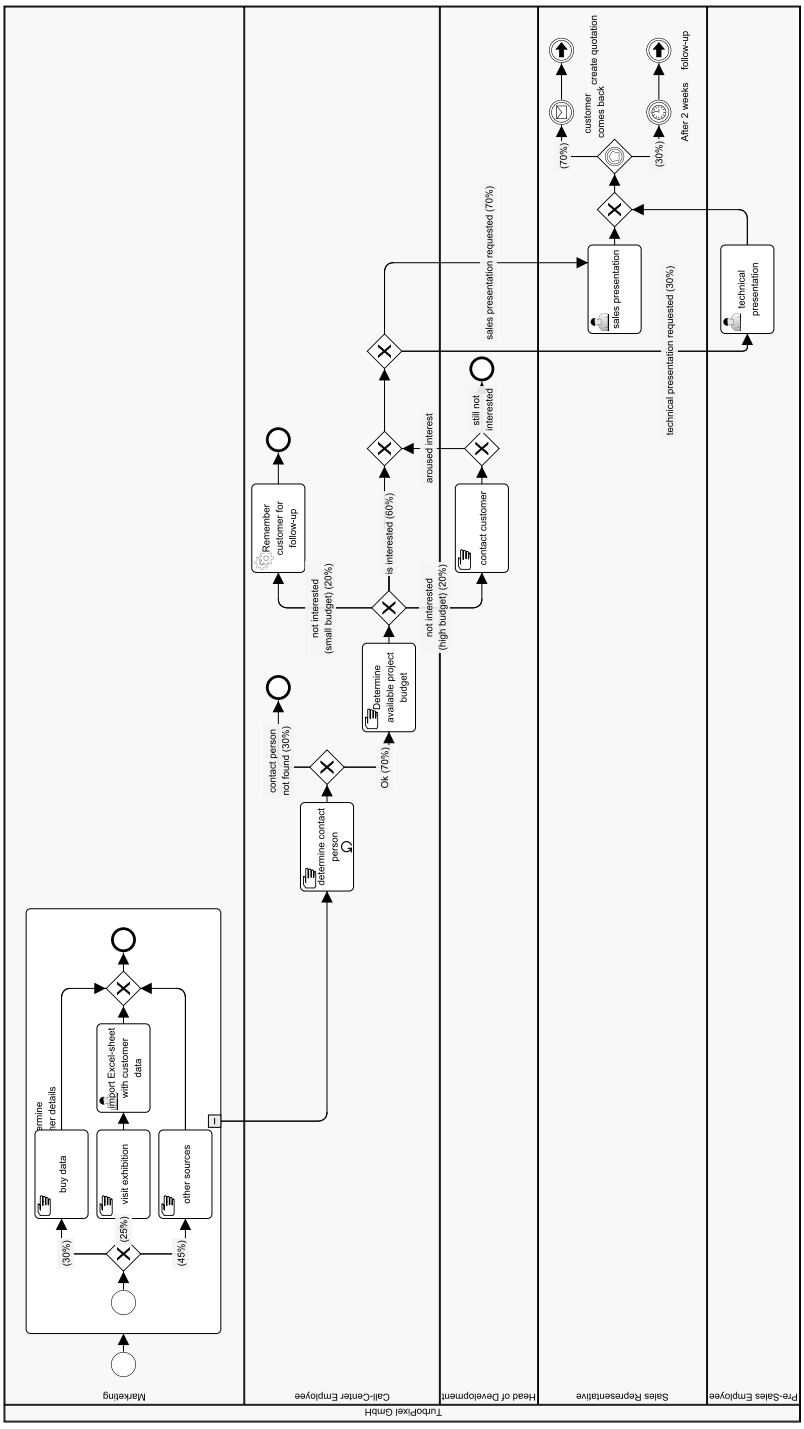

Figure B.72: Model 6-4 as created by a human modeler.

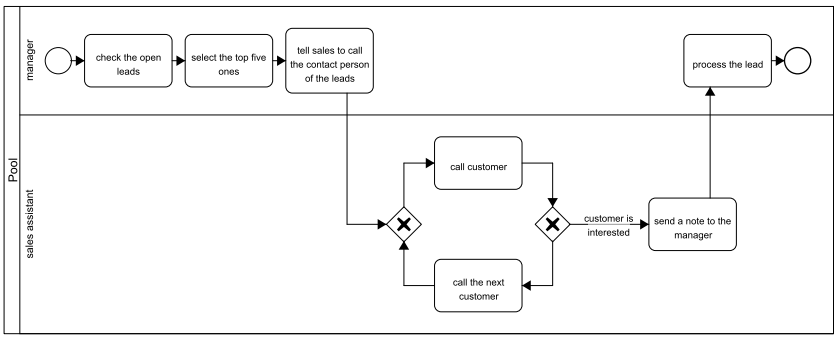

Figure B.73: Model 7-1 as generated by our system.

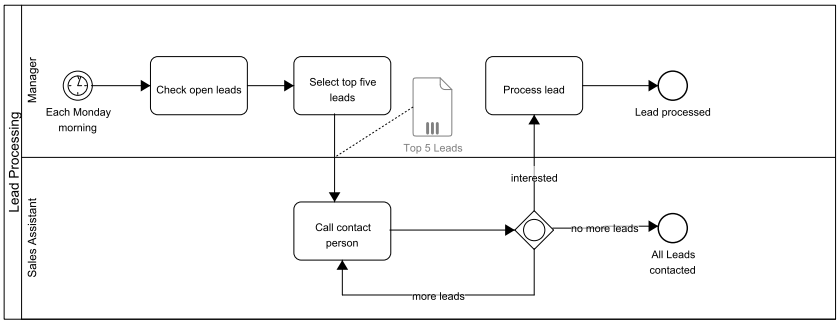

Figure B.74: Model 7-1 as created by a human modeler.

Appendix B.7. Models provided by BPM Practitioners

First, the Manager checks the open leads. Afterwards, he selects the top five ones. He then tells his Sales Assistant to call the contact person of the leads. The Sales Assistant calls each customer. If someone is interested, he sends a note to the Manager. The Manager then processes the lead. Otherwise, he calls the next customer.

Text 23: Process Description 7-1: Calling Leads.

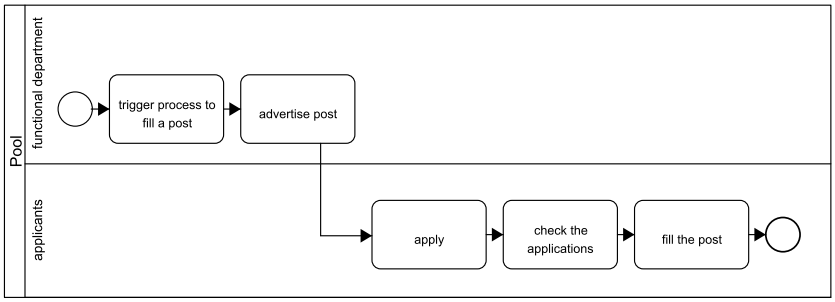

Figure B.75: Model 8-1 as generated by our system.

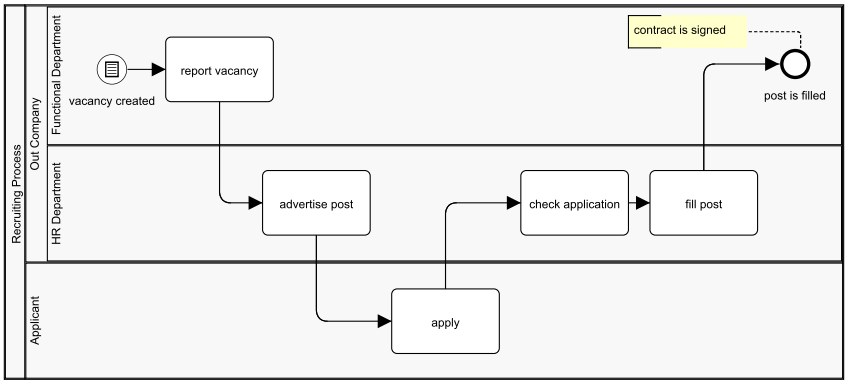

Figure B.76: Model 8-1 as created by a human modeler.

Appendix B.8. Models taken from the BPMN practical handbook

The process is triggered by the demand of a functional department to fill a post. The post is advertised, applicants apply, the applications are checked and the post is filled. The process finishes when the post was filled, precisely through the conclusion of a contract of employment.

Text 24: Process Description 8-1: HR Process - Simple.

When a vacancy is reported to me, I create a job description from the information. Sometimes there is still confusion in the message, then I must ask the Department again. I am submitting the job description for consideration and waiting for the approval. But, it can also happen that the department does not approve it, but rejects it, and requests a correction. Then I correct the description and submit it again for consideration. If the description is finally approved, I post the job.

Text 25: Process Description 8-2: HR Process - HR Department.

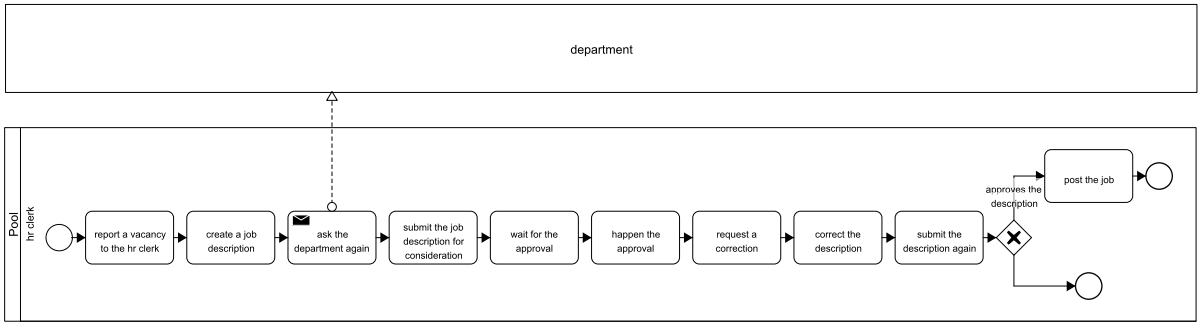

Figure B.77: Model 8-2 as generated by our system.

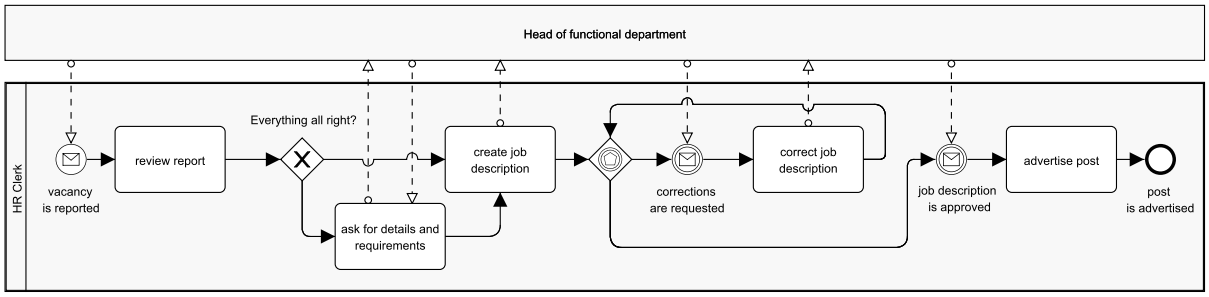

Figure B.78: Model 8-2 as created by a human modeler.

When I have detected a number of personnel requirements, I report the vacancy to the Personnel Department. Then I wait to get the job description for review before it is advertized. Under certain circumstances, I must ask for corrections again, otherwise I approve the job description. Sometimes it also happens that the colleague from the HR department still has questions about the tasks and requirements before he can describe the job. Then I am available for clarifications, of course.

Text 26: Process Description 8-3: HR Process - Functional Department.

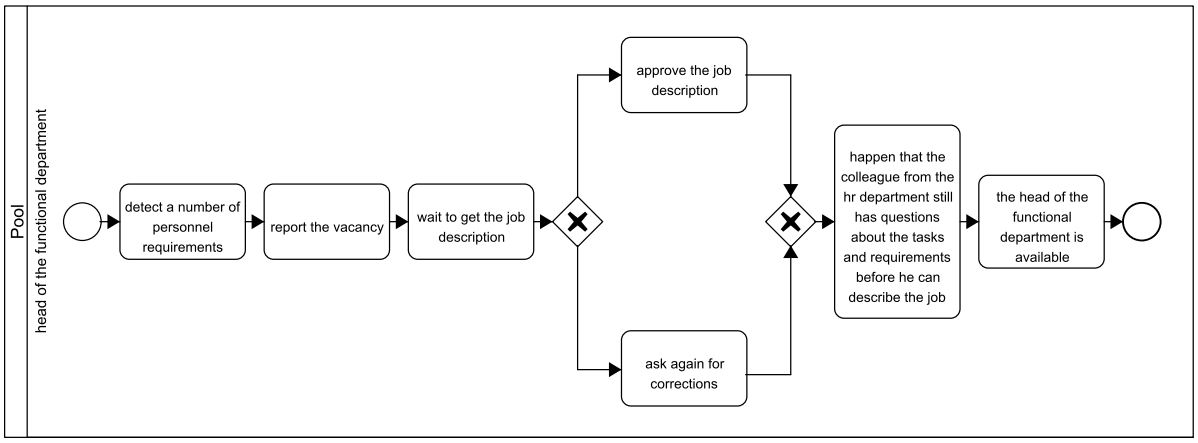

Figure B.79: Model 8-3 as generated by our system.

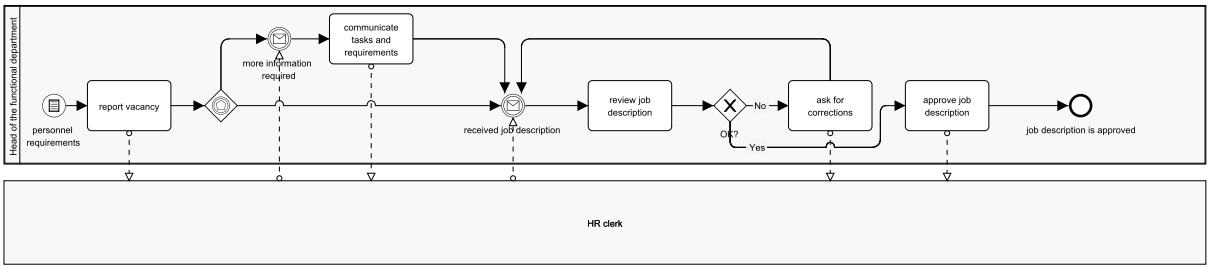

Figure B.80: Model 8-3 as created by a human modeler.

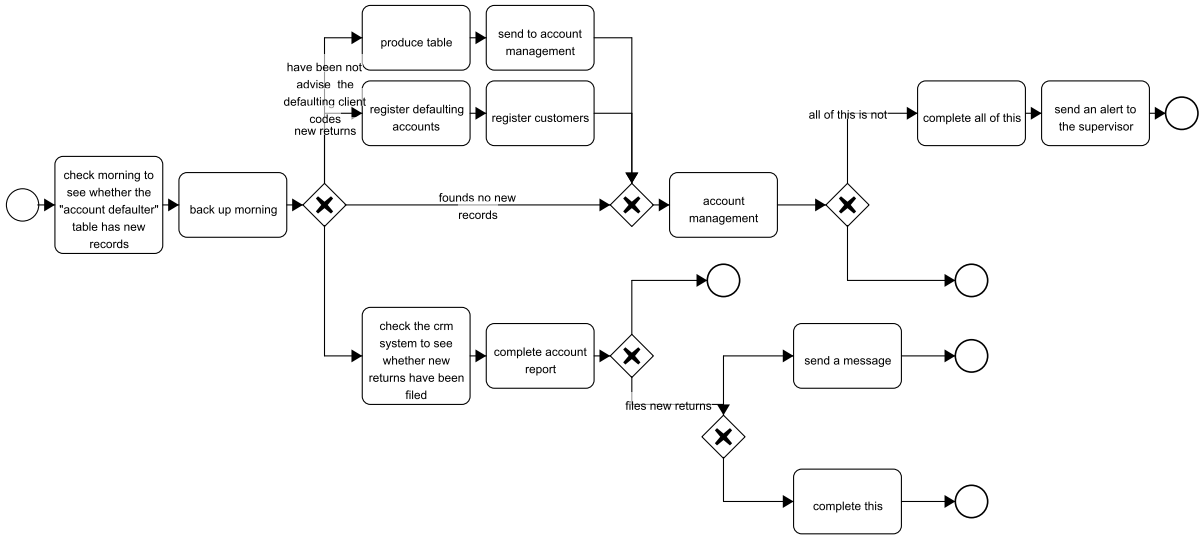

Figure B.81: Model 9-1 as generated by our system.

Appendix B.9. Models taken from the BPMN Modeling an Reference Guide

When I have detected a number of personnel requirements, I report the vacancy to the Personnel Department. Then I wait to get the job description for review before it is advertized. Under certain circumstances, I must ask for corrections again, otherwise I approve the job description. Sometimes it also happens that the colleague from the HR department still has questions about the tasks and requirements before he can describe the job. Then I am available for clarifications, of course.

Text 27: Process Description 9-1: Exercise 1.

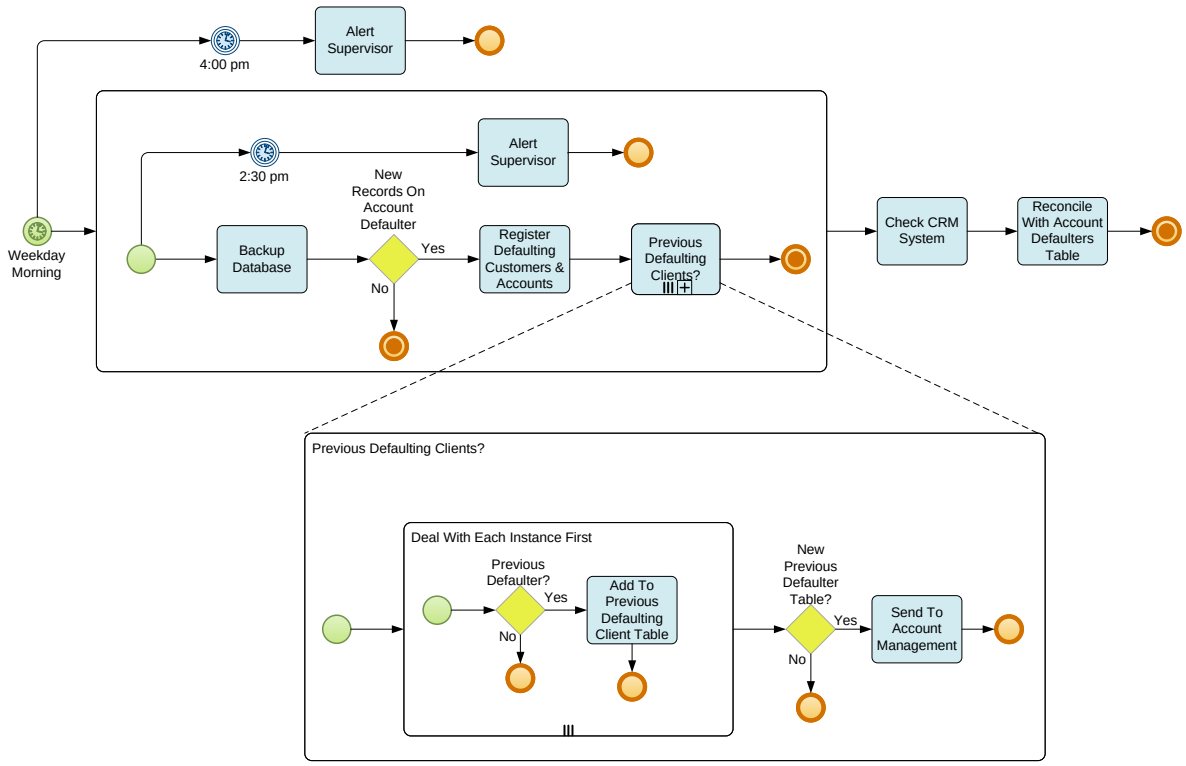

Figure B.82: Model 9-1 as created by a human modeler.

The Customer Service Representative sends a Mortgage offer to the customer and waits for a reply. If the customer calls or writes back declining the mortgage, the case details are updated and the work is then archived prior to cancellation. If the customer sends back the completed offer documents and attaches all prerequisite documents then the case is moved to administration for completion. If all pre-requisite documents are not provided a message is generated to the customer requesting outstanding documents. If no answer is received after 2 weeks, the case details are updated prior to archive and cancellation.

Text 28: Process Description 9-2: Exercise 2.

In November of each year, the Coordination Unit at the Town Planning Authority drafts a schedule of meetings for the next calendar year and adds draft dates to all calendars. The Support Officer then checks the dates and suggests modifications. The Coordination Unit then rechecks all dates and looks for potential conflicts. The final schedule of meeting dates is sent to all the independent Committee Members by email, who then check their diaries and advise the Coordination Unit of any conflicts.

Text 29: Process Description 9-3: Exercise 3a.

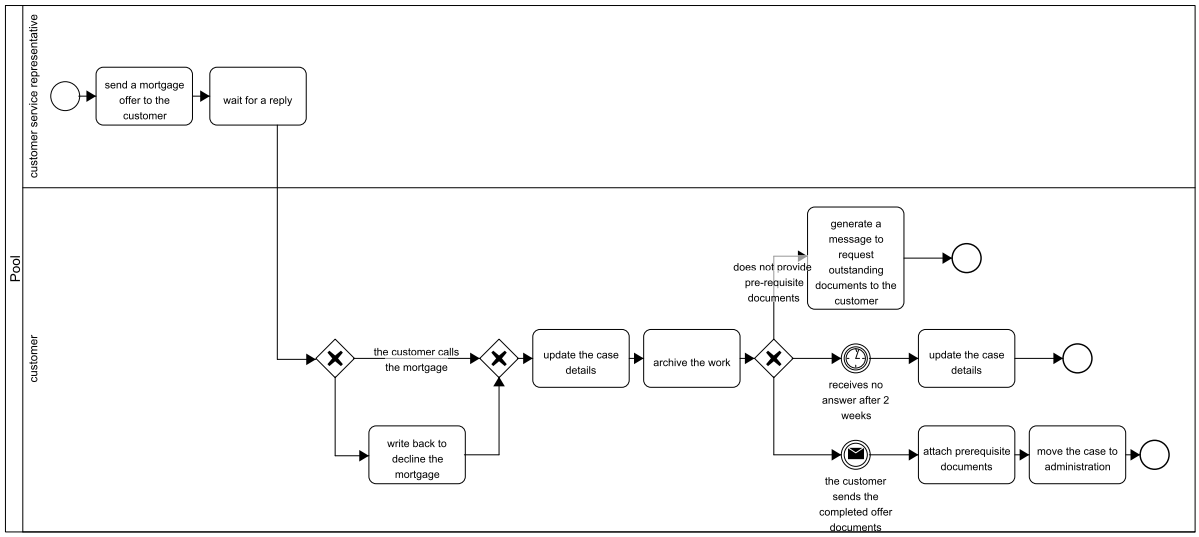

Figure B.83: Model 9-2 as generated by our system.

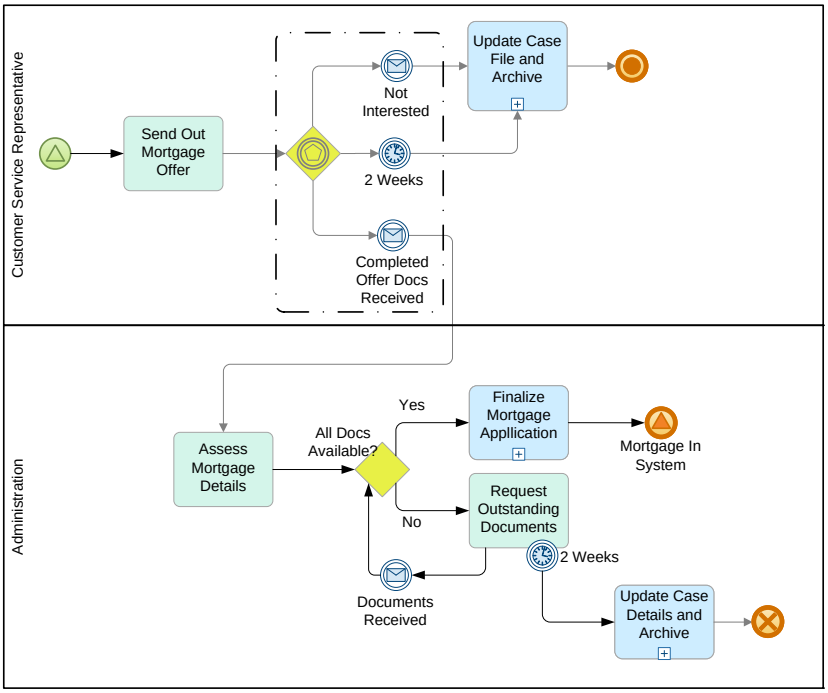

Figure B.84: Model 9-2 as created by a human modeler.

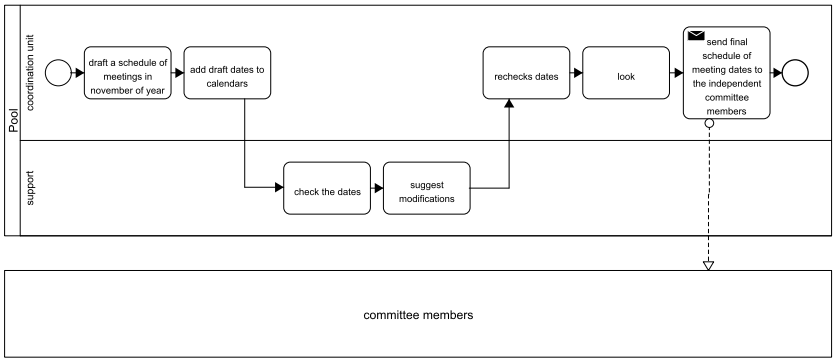

Figure B.85: Model 9-3 as generated by our system.

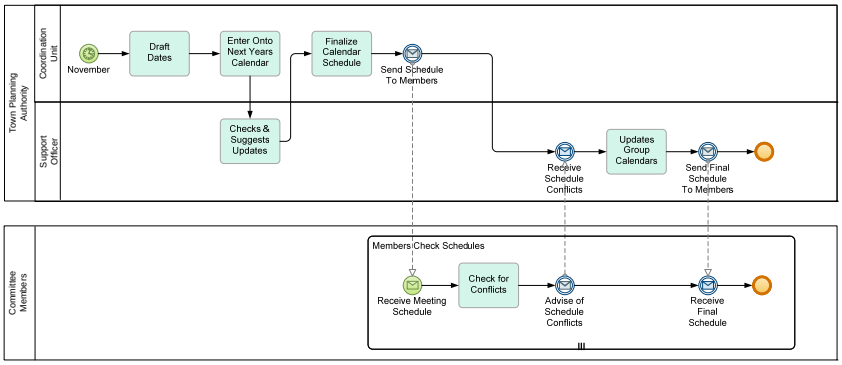

Figure B.86: Model 9-3 as created by a human modeler.

Once the dates are finalized (by the Coordination Unit), the Support Officer updates all group calendars and creates meeting folders for each meeting and ensures all appropriate documents are uploaded to system. Committee Members are advised a week before each meeting to read all related documents. The Committee Members hold their meeting, and the Support Office then produces minutes including any Action Points for each Committee Member. Within 5 working days, the Coordination Unit must conduct a QA check on the minutes, which are then sent to all Committee Members. The Support Officer then updates all departmental records.

Text 30: Process Description 9-4: Exercise 3b.

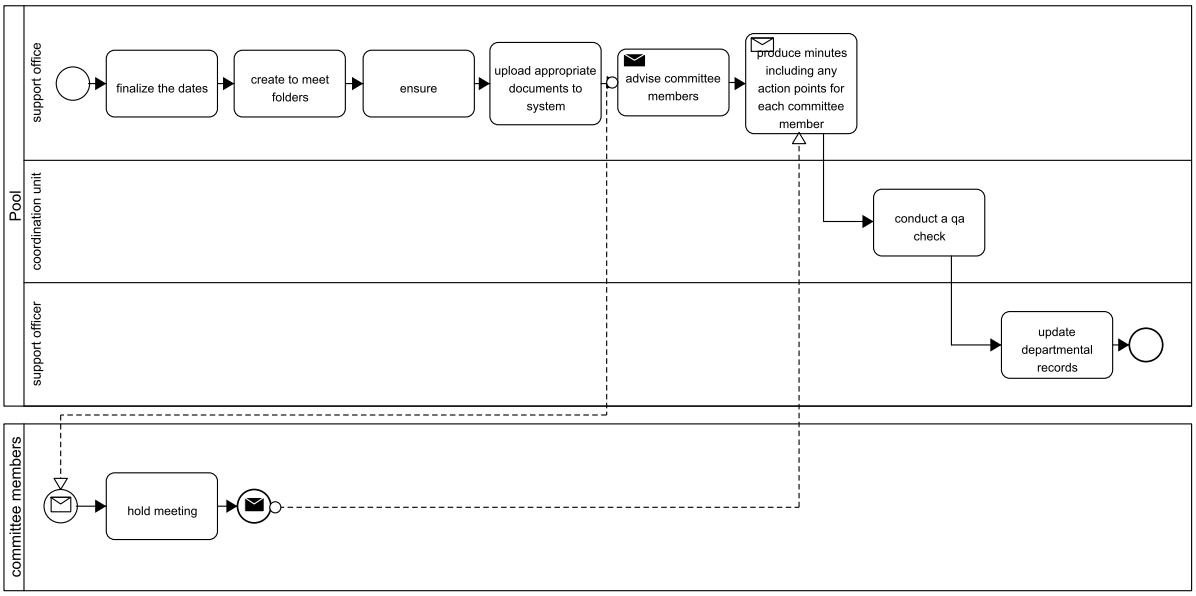

Figure B.87: Model 9-4 as generated by our system.

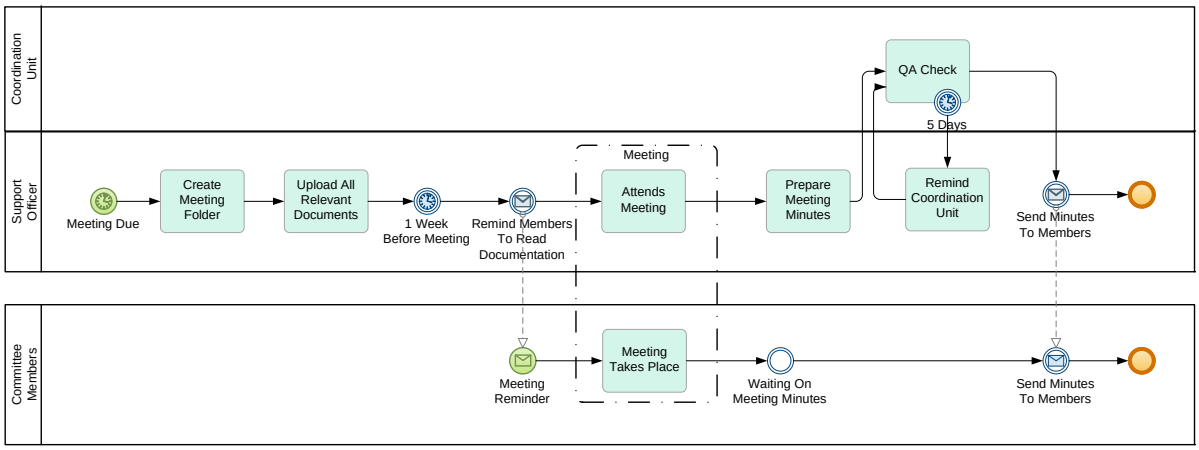

Figure B.88: Model 9-4 as created by a human modeler.

Once the dates are finalized (by the Coordination Unit), the Support Officer updates all group calendars and creates meeting folders for each meeting and ensures all appropriate documents are uploaded to system. Committee Members are advised a week before each meeting to read all related documents. The Committee Members hold their meeting, and the Support Office then produces minutes including any Action Points for each Committee Member. Within 5 working days, the Coordination Unit must conduct a QA check on the minutes, which are then sent to all Committee Members. The Support Officer then updates all departmental records.

Text 31: Process Description 9-5: Exercise 4.

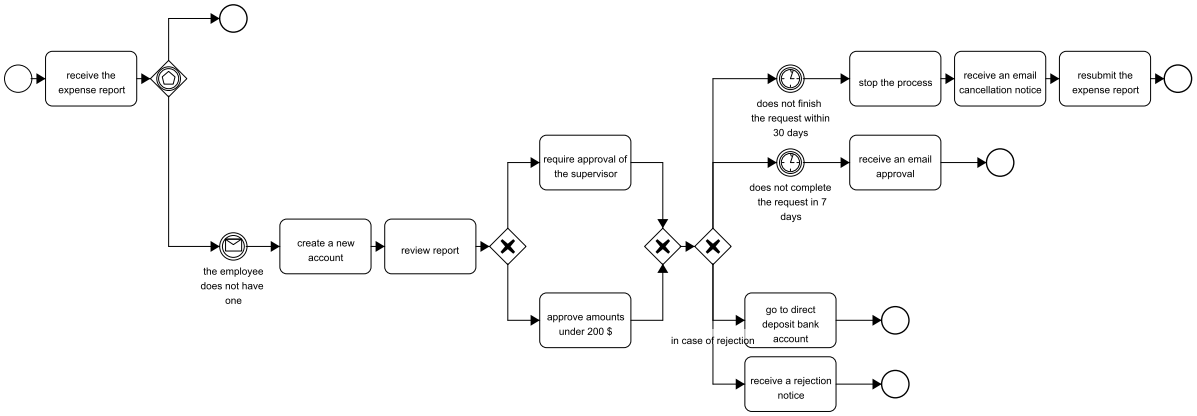

Figure B.89: Model 9-5 as generated by our system.

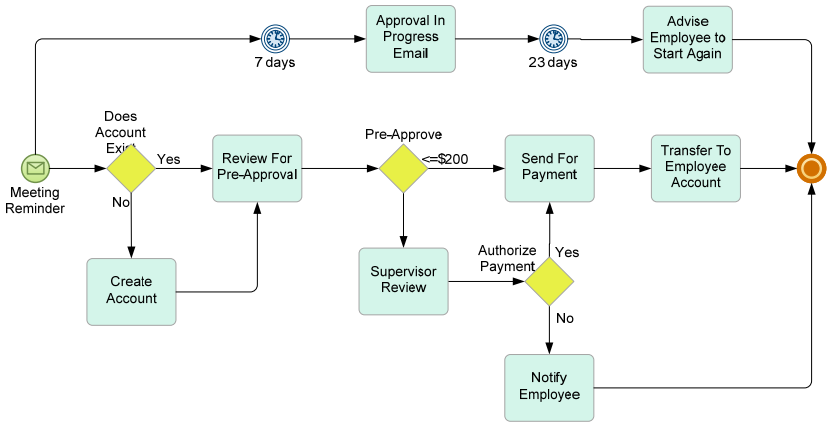

Figure B.90: Model 9-5 as created by a human modeler.

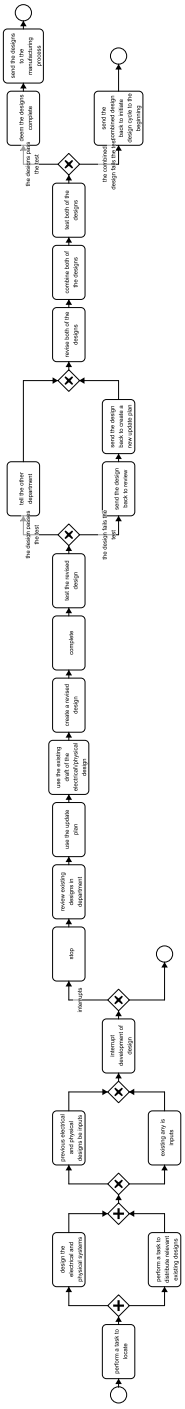

Figure B.91: Model 9-6 as generated by our system.

After the Process starts, a Task is performed to locate and distribute any relevant existing designs, both electrical and physical. Next, the design of the electrical and physical systems starts in parallel. Any existing or previous Electrical and Physical Designs are inputs to both Activities. Development of either design is interrupted by a successful update of the other design. If interrupted, then all current work is stopped and that design must restart. In each department (Electrical Design and Physical Design), any existing designs are reviewed, resulting in an Update Plan for their respective designs (i.e. one in Electrical and another in Physical). Using the Update Plan and the existing Draft of the Electrical/Physical Design, a revised design is created. Once completed the revised design is tested. If the design fails the test, then it is sent back to the first Activity (in the department) to review and create a new Update Plan. If the design passes the test, then it tells the other department that they need to restart their work. When both of the designs have been revised, they are combined and tested. If the combined design fails the test, then they are both sent back to the beginning to initiate another design cycle. If the designs pass the test, then they are deemed complete and are then sent to the manufacturing Process [a separate Process].

Text 32: Process Description 9-5: Exercise 4.

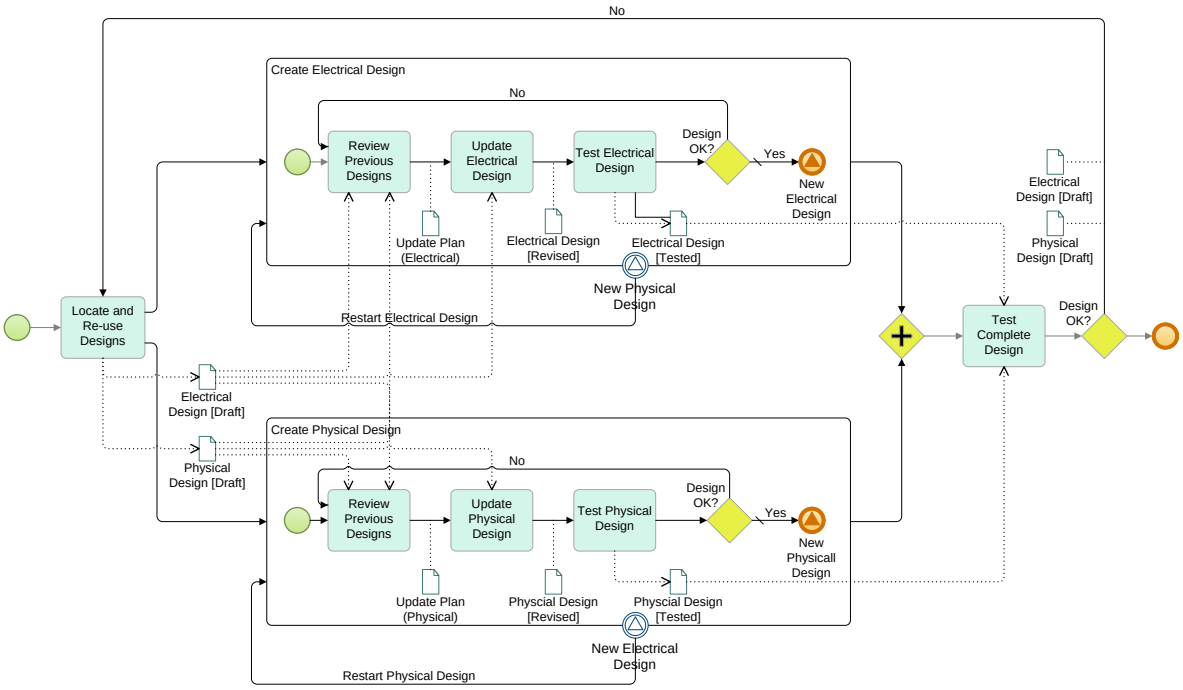

Figure B.92: Model 9-6 as created by a human modeler.

Abbr. Ger.	Amplification Ger.	Amplification Eng.	Abbr. Eng.
L	Letztverbraucher	End consumer	EC
LF	Lieferant	Supplier	SP
MSB	Messstellenbetreiber	Metering point operator	MPO
MSBA	Messstellenbetreiber alt	Metering point operator old	MPOO
MSBN	Messstellenbetreiber neu	Metering point operator new	MPO
MDL	Messdienstleister	Metering service provider	MSP
MDLA	Messdienstleister alt	Metering service provider old	MSPO
MDLN	Messdienstleister neu	Metering service provider new	MSPN
NB	Netzbetreiber	Grid operator	GO
AN	Anschlussnutzer	Power supply user	PU
AG	Angefragter	Inquired person	IP
AF	Anfragender	Inquirer	INQ

Table B.11: List of abbreviations and translations used in the FNA Test Data Set.

Appendix B.10. Models taken from a Federal Network Agency Enactment

The texts contained in this test data set were translated from German using Google translate and correcting grammatical errors. Additionally, the original texts made of several abbreviations specific to the domain of electric power supply. In order to be consistent, these were also translated. The translation and the meaning of the abbreviations in English and German are given in table B.11.

The MPON sents the dismissal to the MPOO. The MPOO reviews the dismissal. The MPOO opposes the dismissal of MPON or the MPOO confirmes the dismissal of the MPON.

Text 33: Process Description 10-1: Process B2.

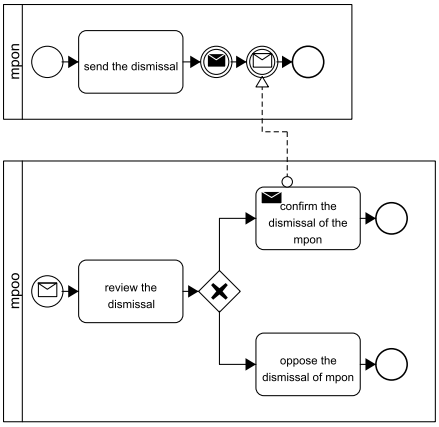

Figure B.93: Model 10-1 as generated by our system.

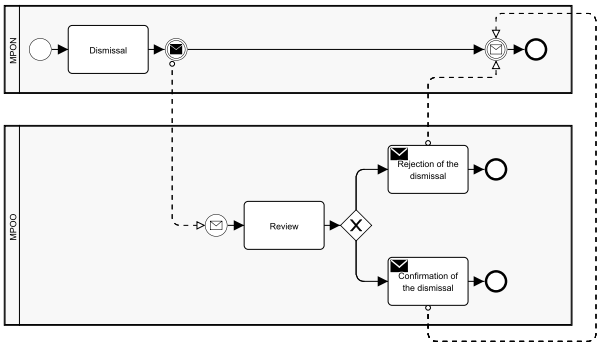

Figure B.94: Model 10-1 Sequence Diagram transformed to BPMN.

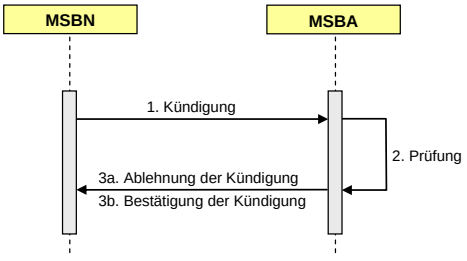

Figure B.95: Model 10-1 originally provided Sequence Diagram.

The MPON reports the meter operation to the GO. The GO examines the application of the MPON. The GO rejects the application of the MPON or the GO confirmes the application of the MPON. The GO informs the MPOO about the registration confirmation of the MPON. The GO informs the MSPO about the registration confirmation of the MPON. The MPON and the MPOO perform the equipment acquisition and/or equipment changes. The MPON informs the GO about the failure of the entire process or the MPON informs the GO about the successful completion of the entire process. The GO informs the MPON about the failure of the overall transaction by deadline if after a maximum time limit no message of the MPON is present at the GO. If the MPON informs the GO about the failure of the entire process, the GO confirms the failure of the assignment to the MPON. If the MPON informs the GO about the successful completion of the overall process, the GO assigns the MPON. The GO confirms the assignment to the MPON. The GO informs the MPOO about the failure of the assignment of the MPON or the GO informs the MPOO about the assignment of the MPON. The GO informs the MSPO about the failure of the assignment of the MPON or the GO informs the MSPO about the assignment of the MPON. The GO informs the SP about the assignment of the MPON.

Text 34: Process Description 10-2: Process B3.

The MPOO deregisters at the GO. The GO verifies the deregistration. The GO rejects the deregistration of the MPOO or the GO preliminarily confirms the deregistration of the MPOO. The GO prepares the readmission of the measuring point. Optionally, the GO may oblige the MPOO to continue the operations. If the GO binds the MPOO to continue the operation, the MPOO confirms the continuation to the MPOO. The GO performs the equipment acquisition and/or equipment changes. The GO assigns the GO as MPO. The GO informs the MPOO about the end of the assignment of the MPOO and the beginning of the assignment of the GO. The GO informs the MSPO about the assignment of the GO. The GO informs the SP about the assignment of the GO.

Text 35: Process Description 10-3: Process B4.

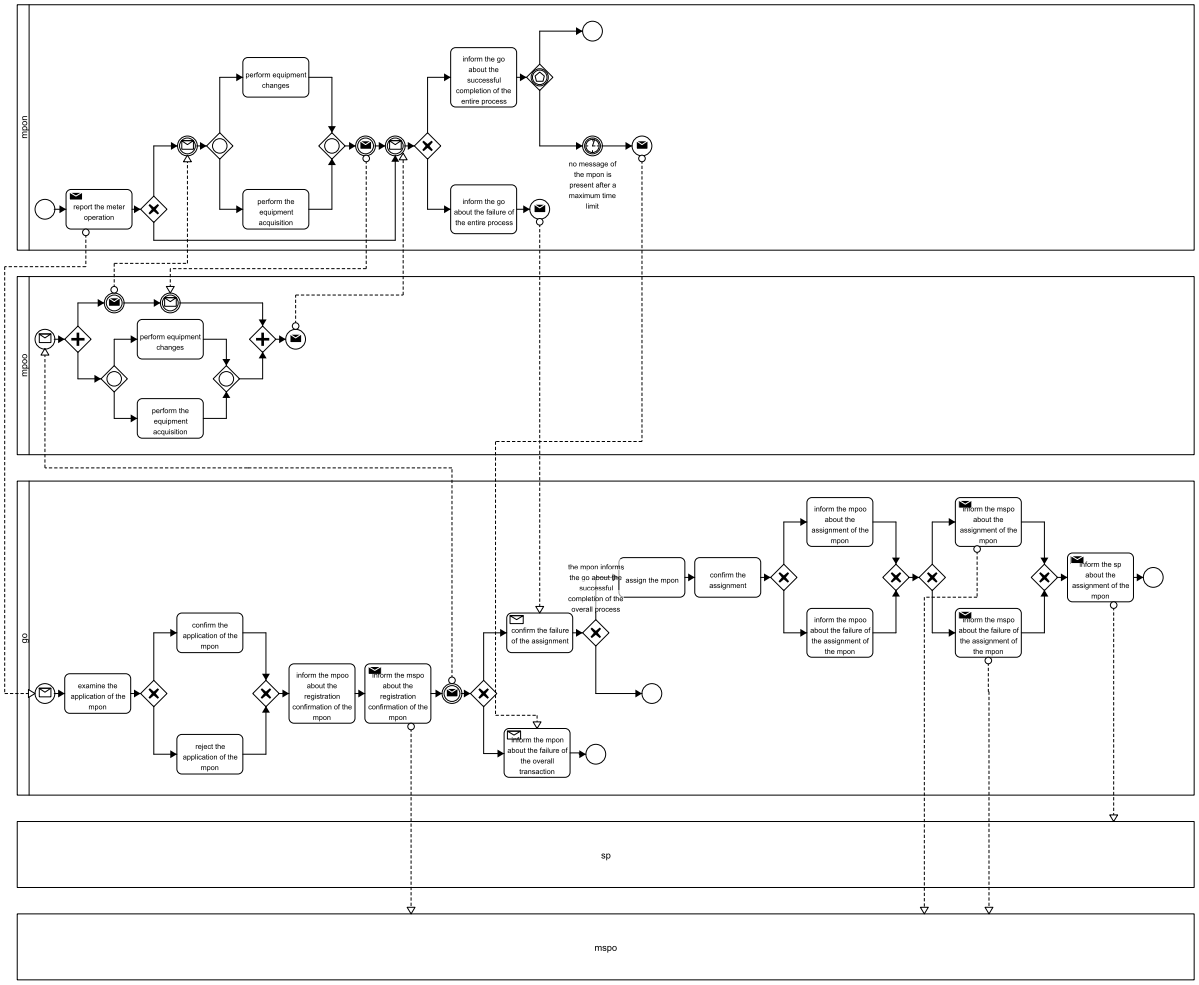

Figure B.96: Model 10-2 as generated by our system.

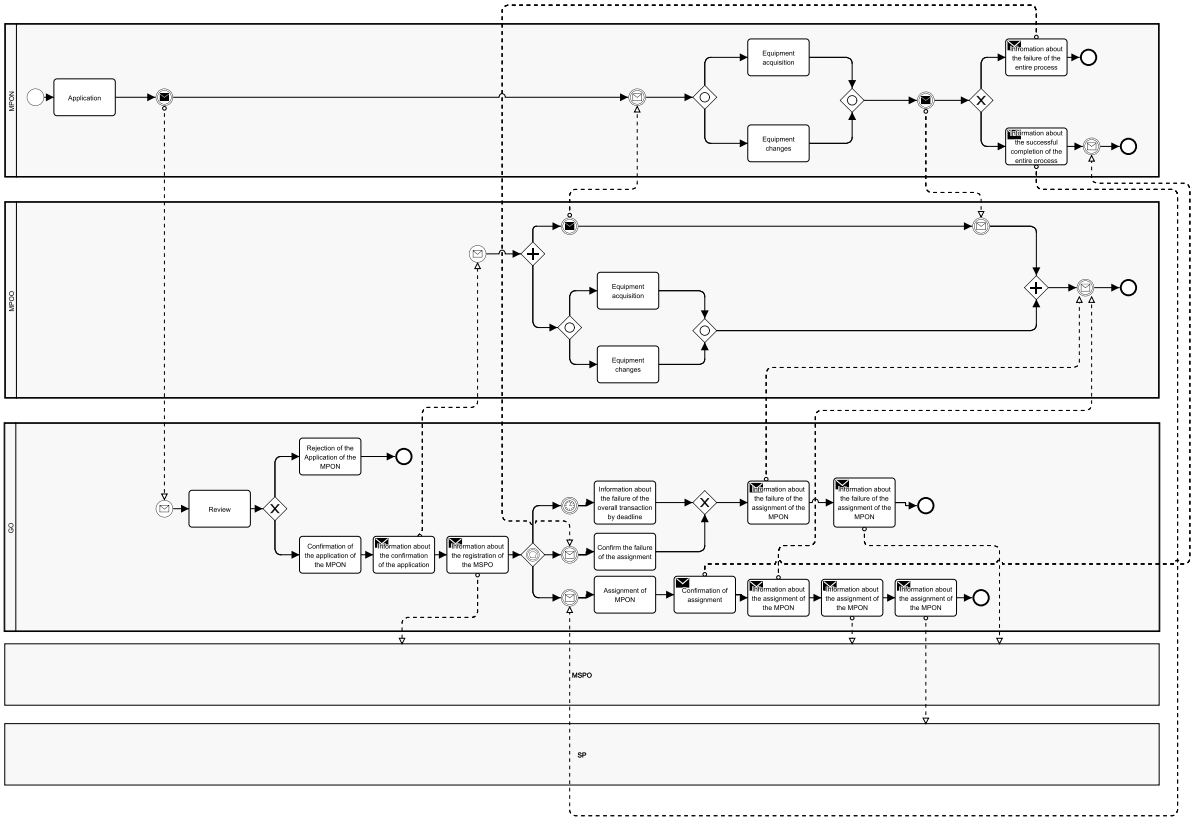

Figure B.97: Model 10-2 Sequence Diagram transformed to BPMN.

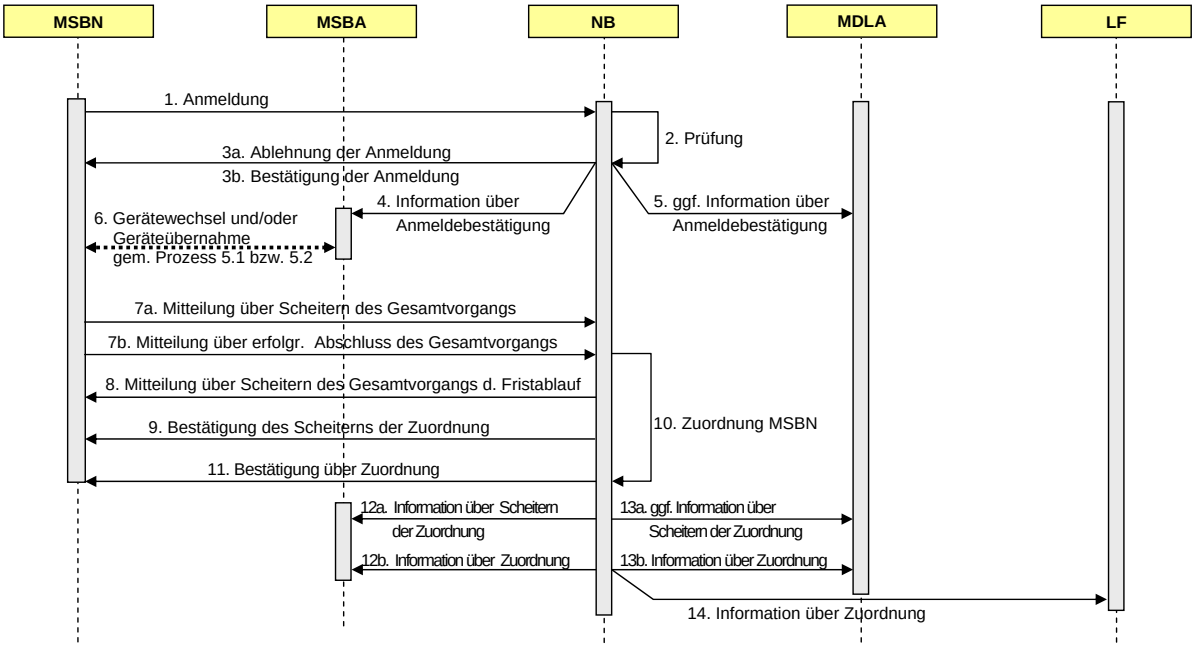

Figure B.98: Model 10-2 originally provided Sequence Diagram.

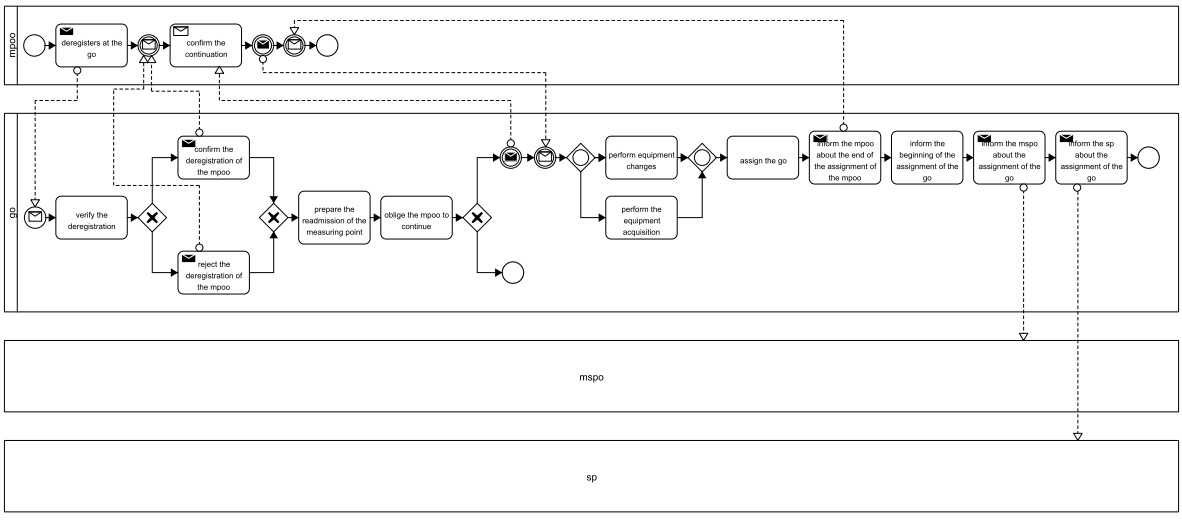

Figure B.99: Model 10-3 as generated by our system.

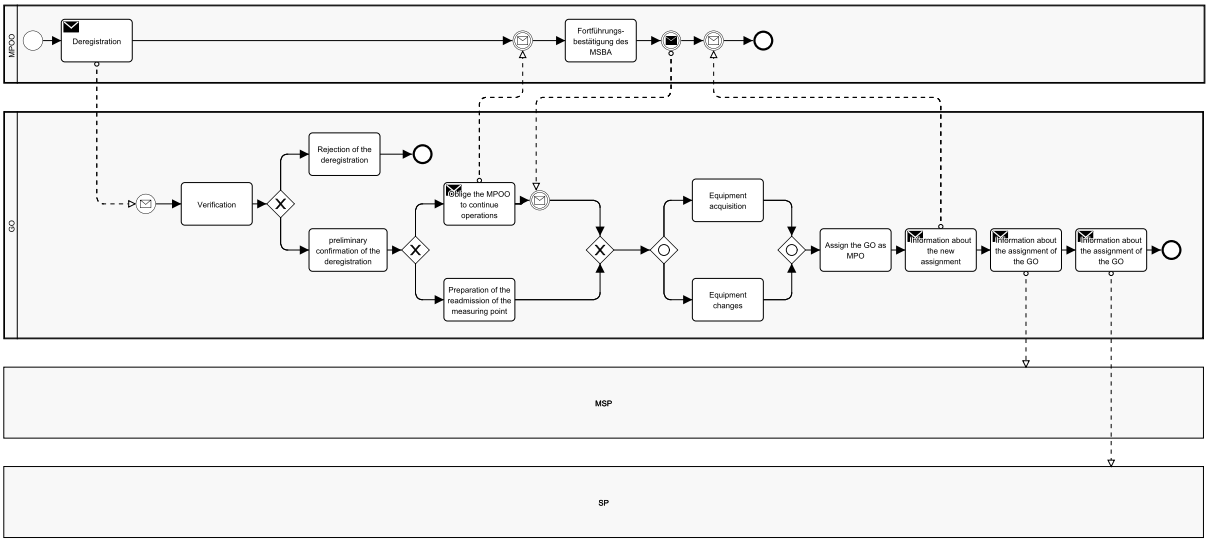

Figure B.100: Model 10-3 Sequence Diagram transformed to BPMN.

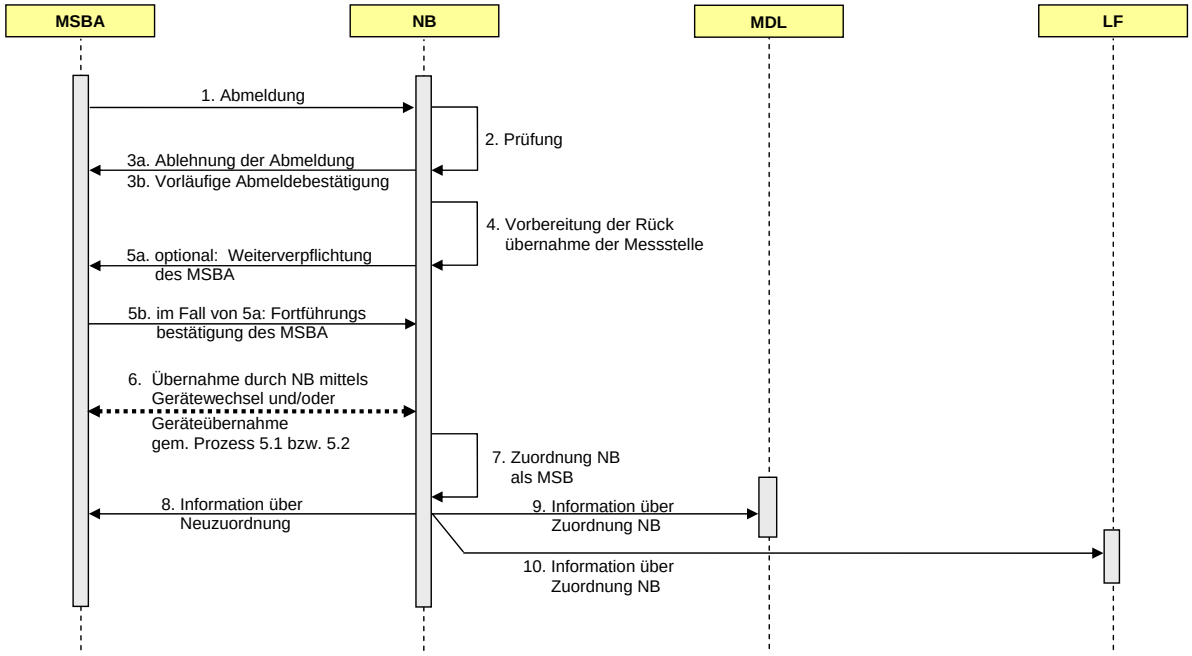

Figure B.101: Model 10-3 originally provided Sequence Diagram.

The MPON notifies the MPOO about equipment change intentions. The MPOO announces self dismounting to the MPON or the MPOO shall notify the MPON about no self-dismounting of the MPOO. The MPON or the MPOO perform the final reading. The MPON or the MPOO dismount the old equipment. The MPON mounts the new device. The MPON reads the meter count from the installed meter. The MPON sents the values of the final reading to the GO. The MPON tells the GO about the device changes, the master data and the meter count at installation. The GO shall notify the MSP about the device changes, the master data, the meter count at dismounting, and the meter count at installation.

Text 36: Process Description 10-4: Process B5.1.

The MPON requests a device takeover bid of the MPOO. The MPOO sends a tender for the equipment takeover to the MPON. The MPON places an order at the MPOO. The MPOO confirms the order of the MPON and sends the master data.

Text 37: Process Description 10-5: Process B5.2.

The MSPN sents a dismissal to the MSPO. The MSPO reviews the dismissal. The MSPO rejects the dismissal of the MSPN or The MSPO confirms the dismissal of the MSPN.

Text 38: Process Description 10-6: Process B6.

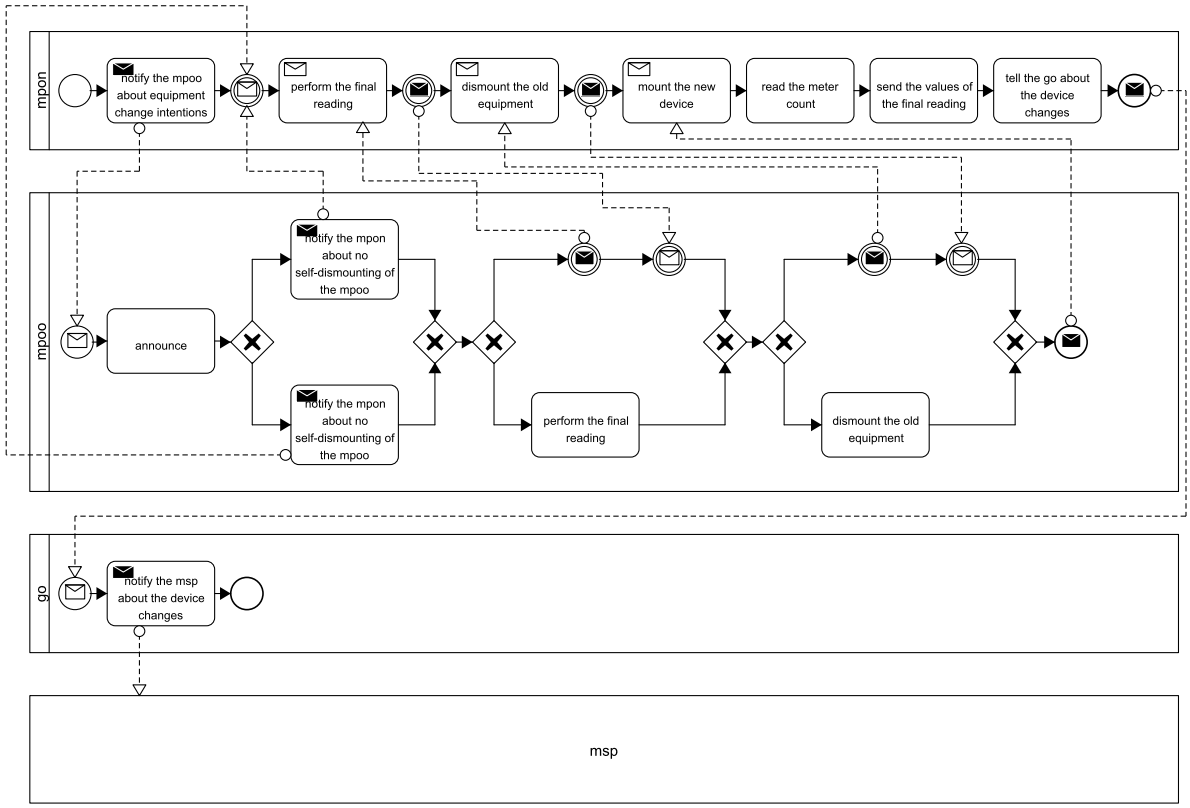

Figure B.102: Model 10-4 as generated by our system.

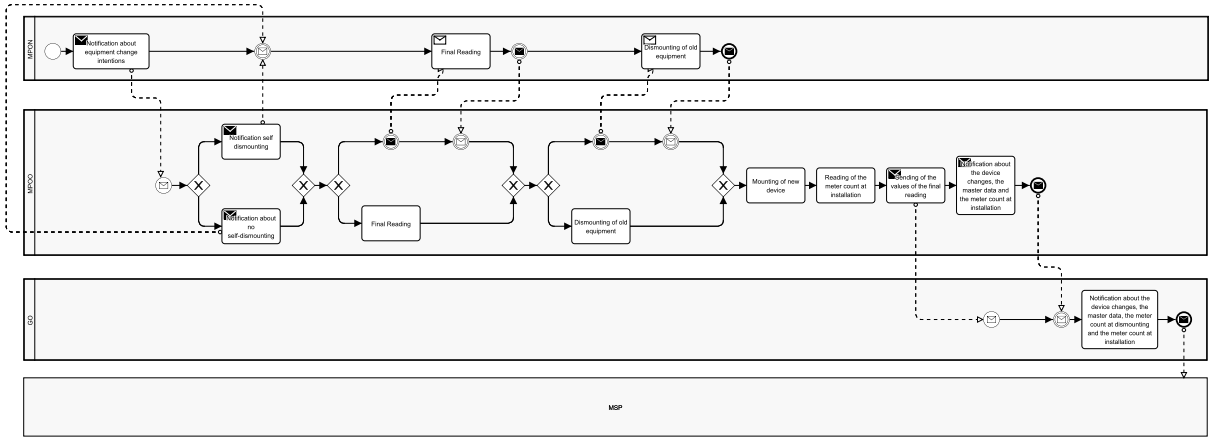

Figure B.103: Model 10-4 Sequence Diagram transformed to BPMN.

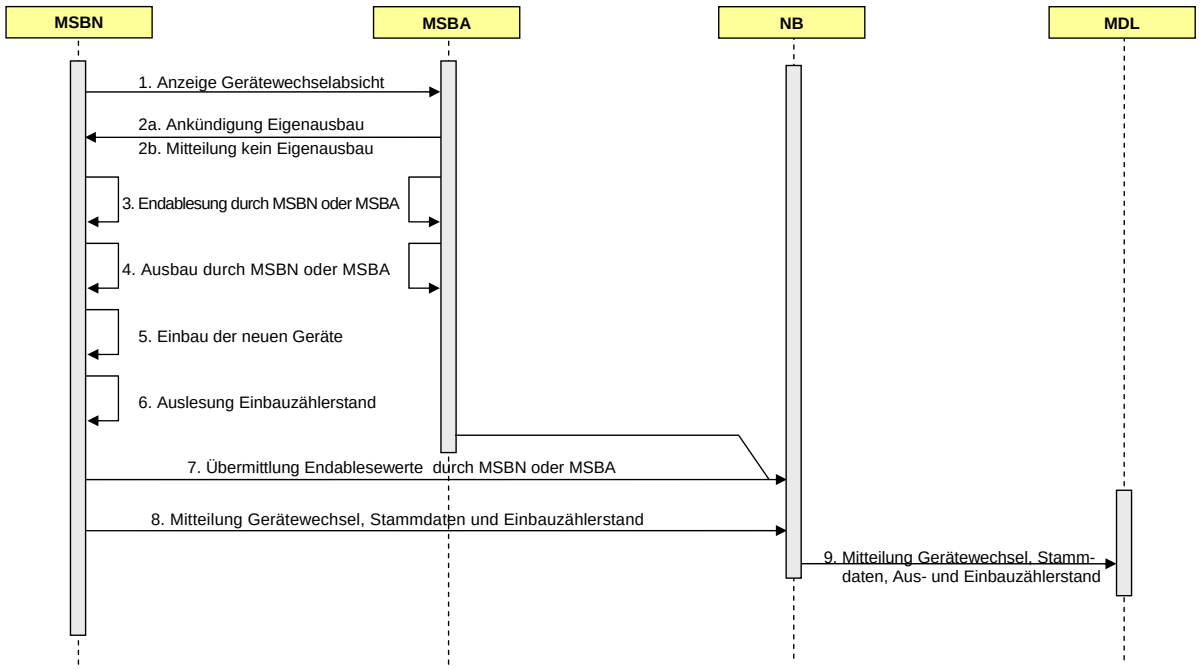

Figure B.104: Model 10-4 originally provided Sequence Diagram.

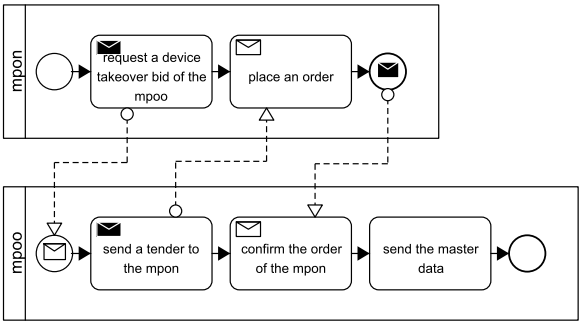

Figure B.105: Model 10-5 as generated by our system.

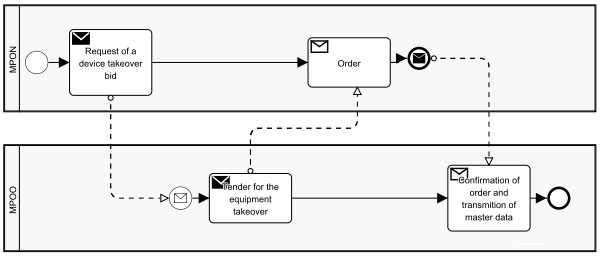

Figure B.106: Model 10-5 Sequence Diagram transformed to BPMN.

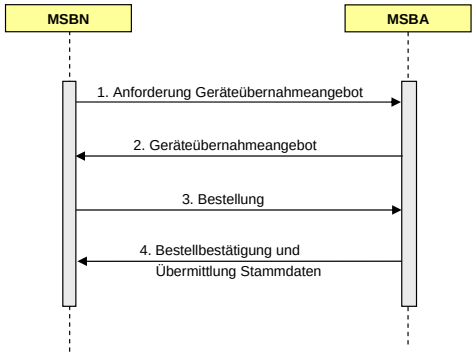

Figure B.107: Model 10-5 originally provided Sequence Diagram.

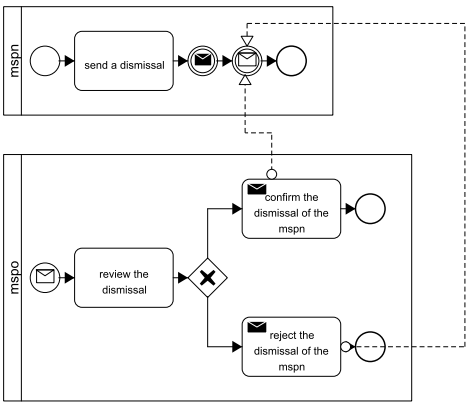

Figure B.108: Model 10-6 as generated by our system.

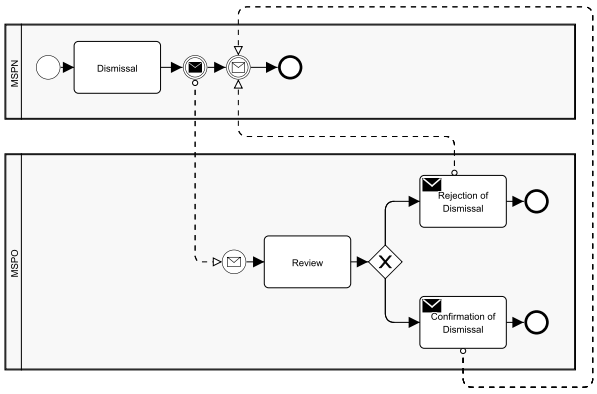

Figure B.109: Model 10-6 Sequence Diagram transformed to BPMN.

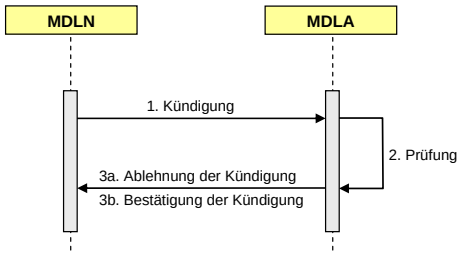

Figure B.110: Model 10-6 originally provided Sequence Diagram.

The MSPN registers the measurement at the GO. The GO examines the application of the MSPN. The GO rejects the application of the MSPN or the GO confirmes the application of the MSPN. The GO assigns the MSPN. The GO informs the MSPO about the assignment of MSPN. The GO informs the MPO about the assignment of the MSPN. The GO informs the SP about the assignment of MSPN.

Text 39: Process Description 10-7: Process B7.

The MSPO deregisters at the GO. The GO verifies the deregistration. The GO rejects the deregistration of the MSPO or the GO preliminarily confirms the deregistration of the MSPO. The GO assigns himself as MSP. The GO informs the MSPO about the end of the assignment and the beginning of the assignment of the GO. The GO informs the MPO about the assignment of the GO. The GO informs the SP about the assignment of the GO.

Text 40: Process Description 10-8: Process B8.

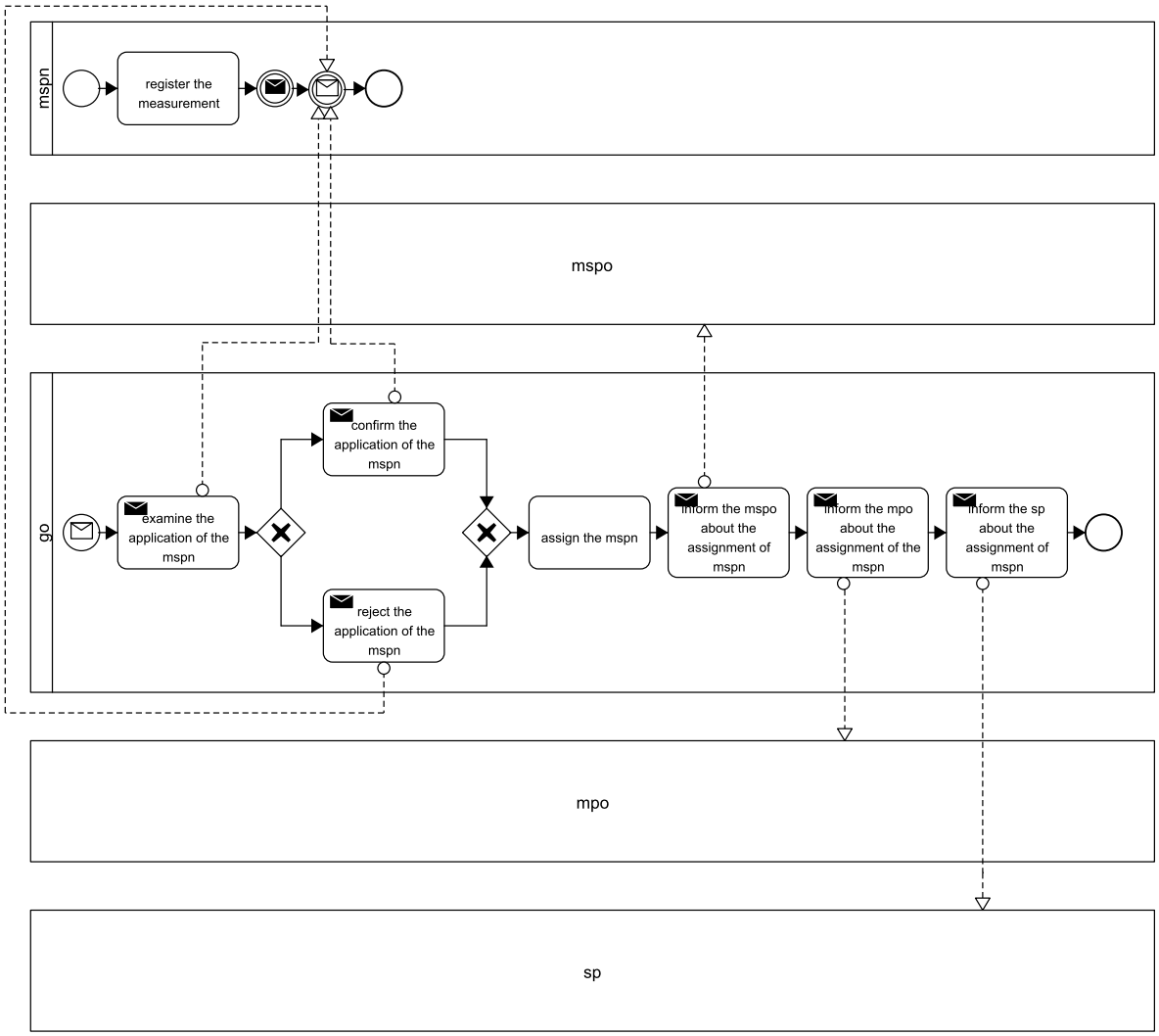

Figure B.111: Model 10-7 as generated by our system.

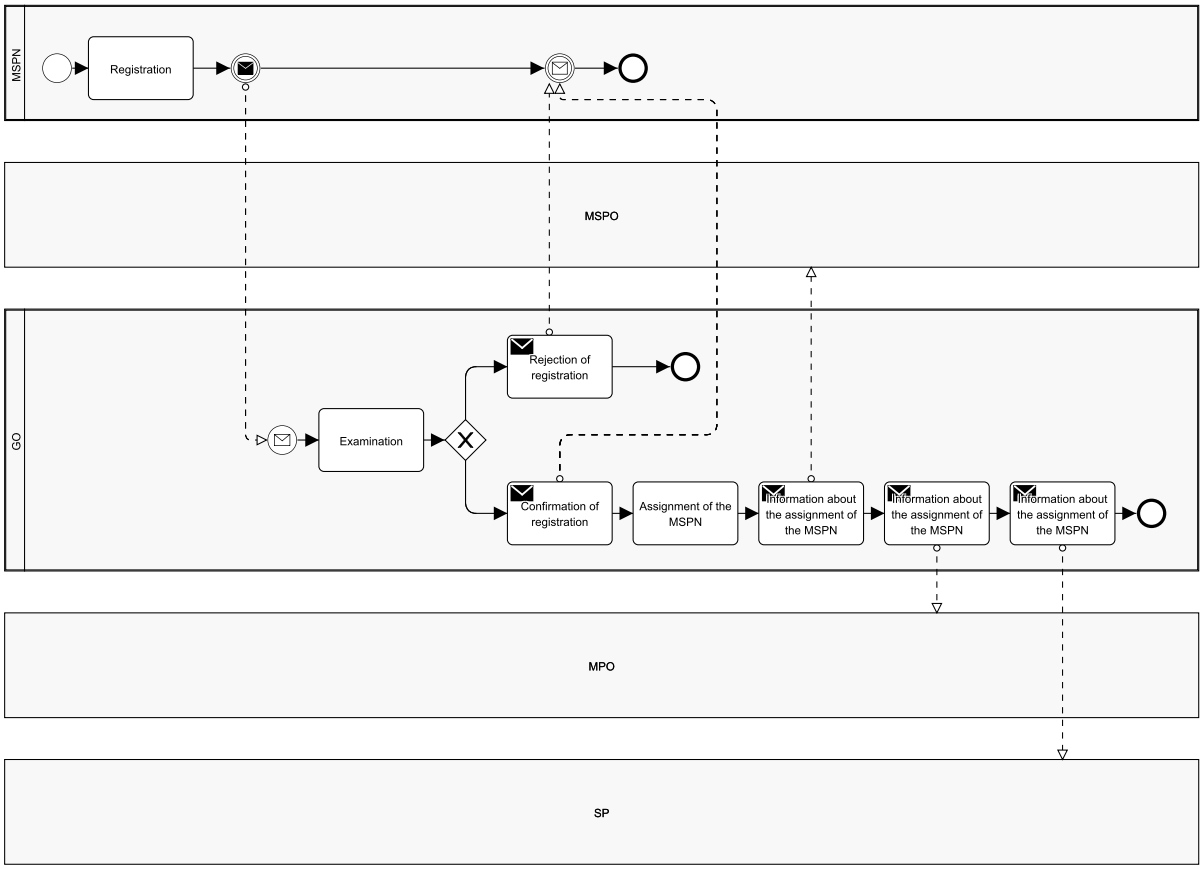

Figure B.112: Model 10-7 Sequence Diagram transformed to BPMN.

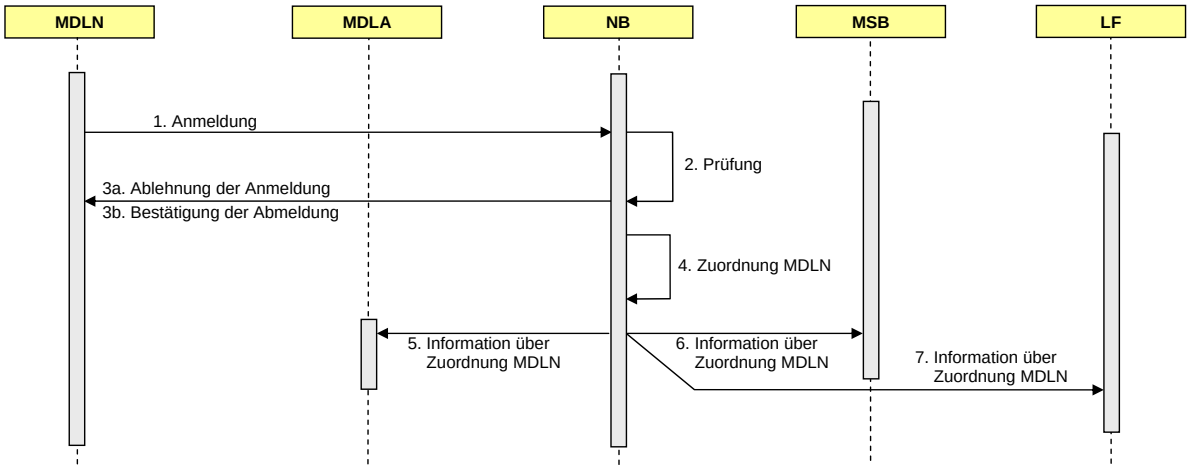

Figure B.113: Model 10-7 originally provided Sequence Diagram.

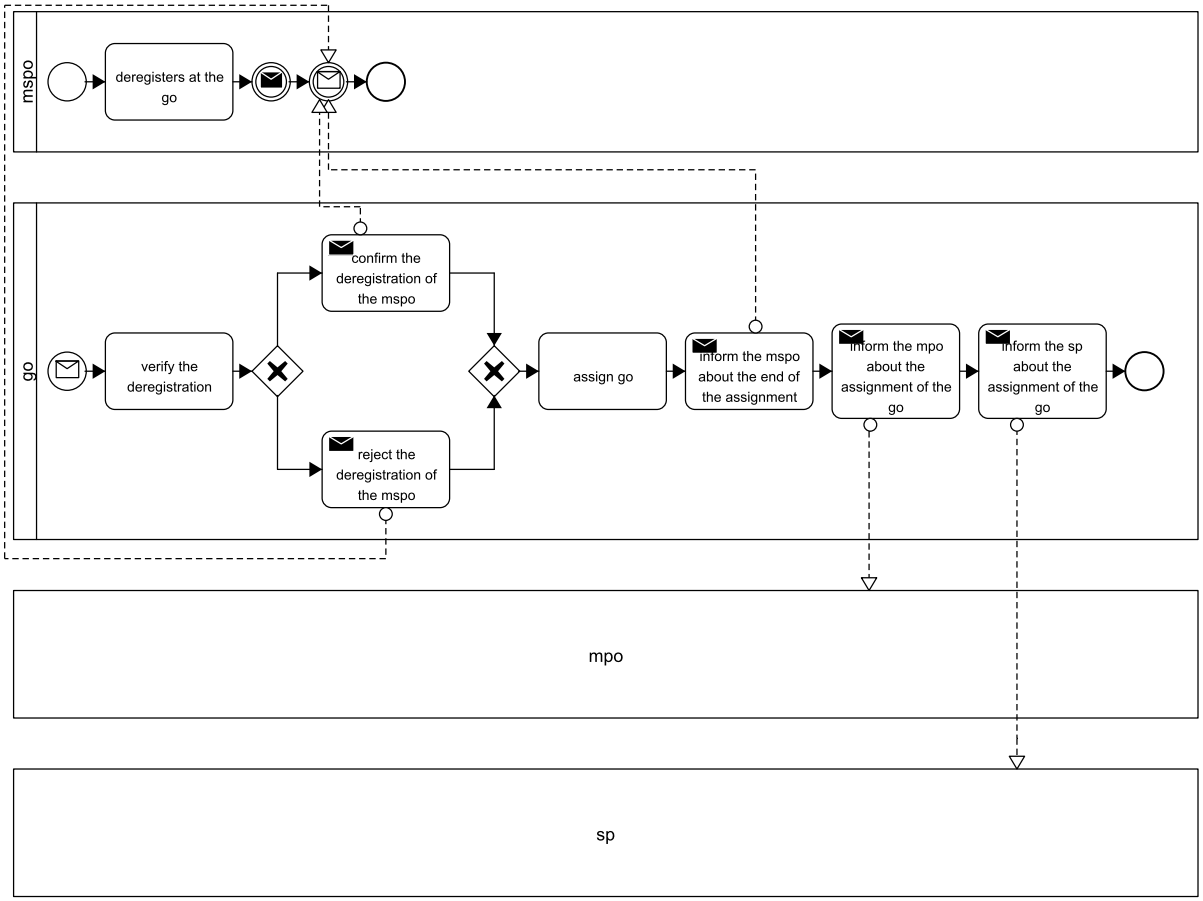

Figure B.114: Model 10-8 as generated by our system.

The SP/PU/GO request changes of the MPO or the MPO himself causes a change. The MPO reviews the change request. The MPO rejects the change of the measuring point by the SP/PU/GO or the MPO confirmes the request of the SP/PU/GO. The MPO performs the measuring point change. The MPO reports the implementation to the SP/PU/GO or notifies the SP/PU/GO about the failure of the changes.

Text 41: Process Description 10-9: Process C1.

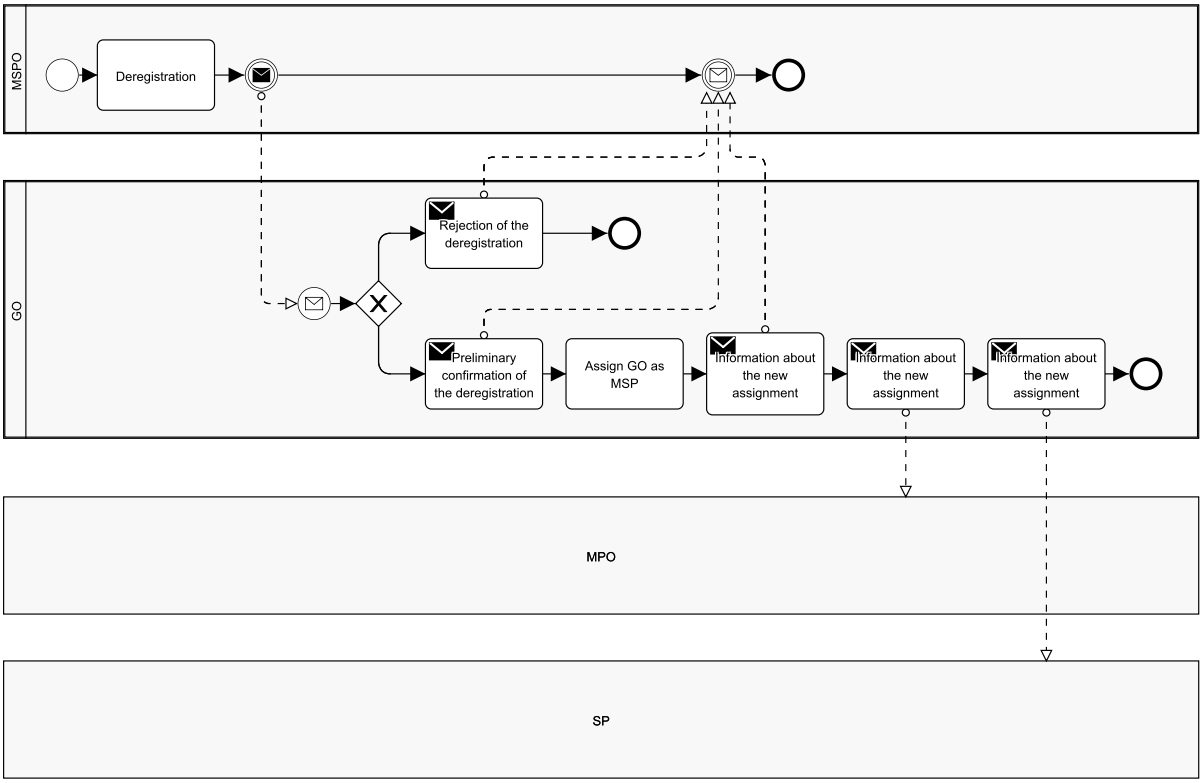

Figure B.115: Model 10-8 Sequence Diagram transformed to BPMN.

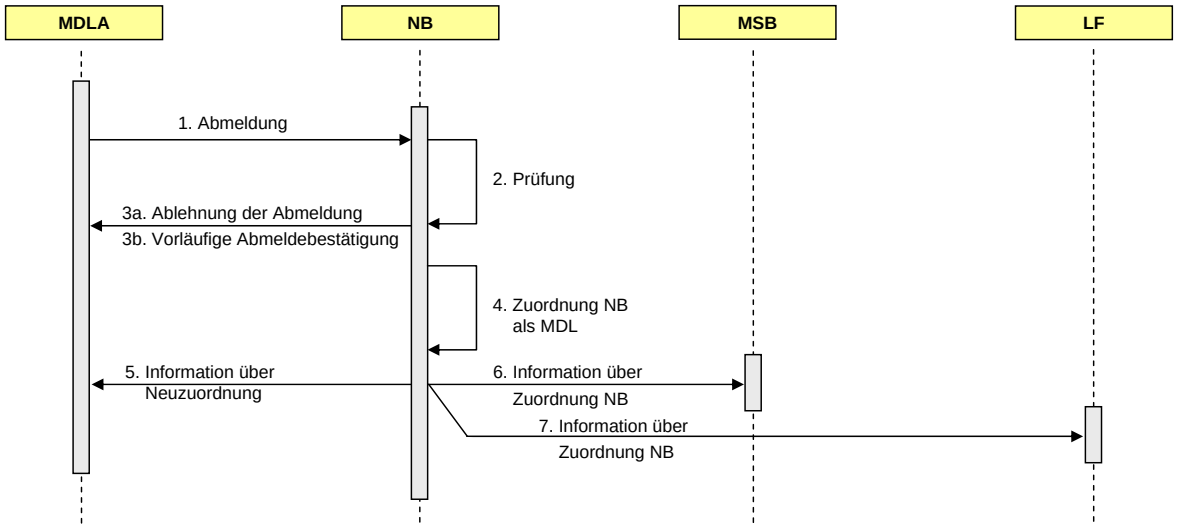

Figure B.116: Model 10-8 originally provided Sequence Diagram.

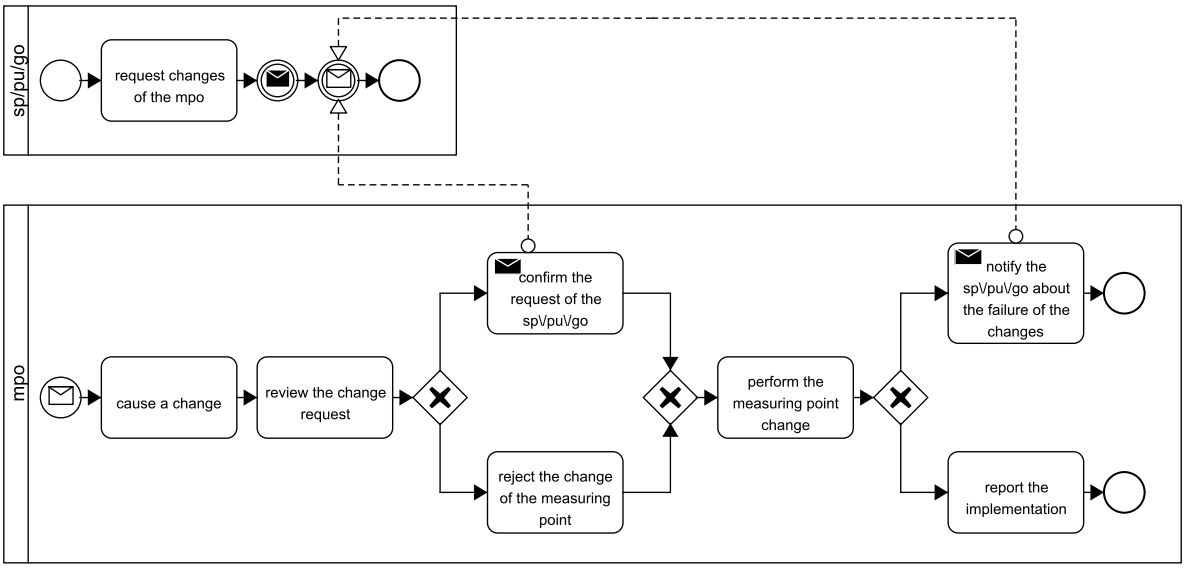

Figure B.117: Model 10-9 as generated by our system.

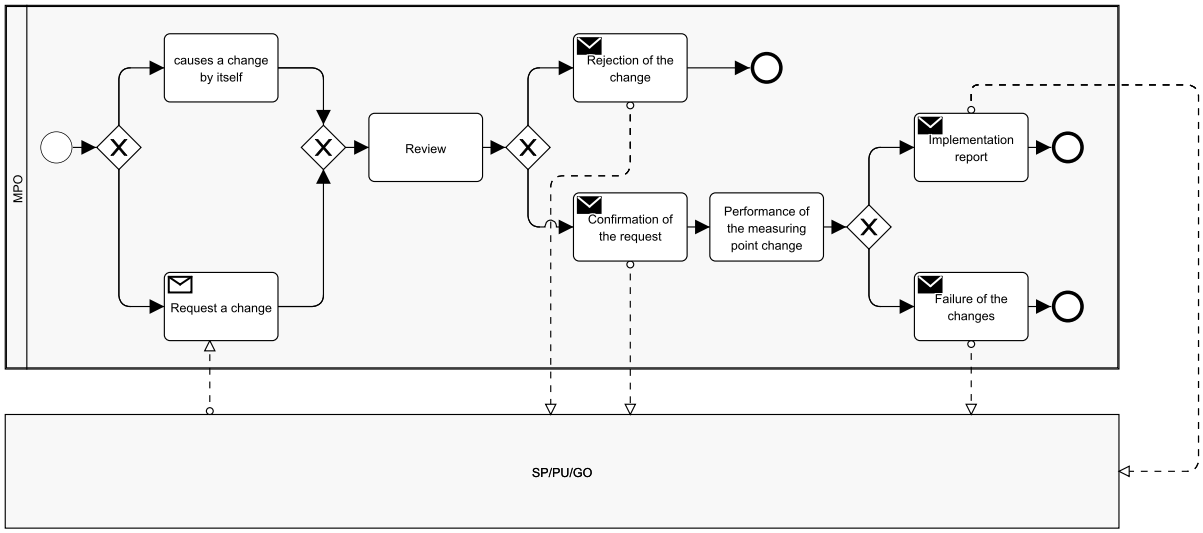

Figure B.118: Model 10-9 Sequence Diagram transformed to BPMN.

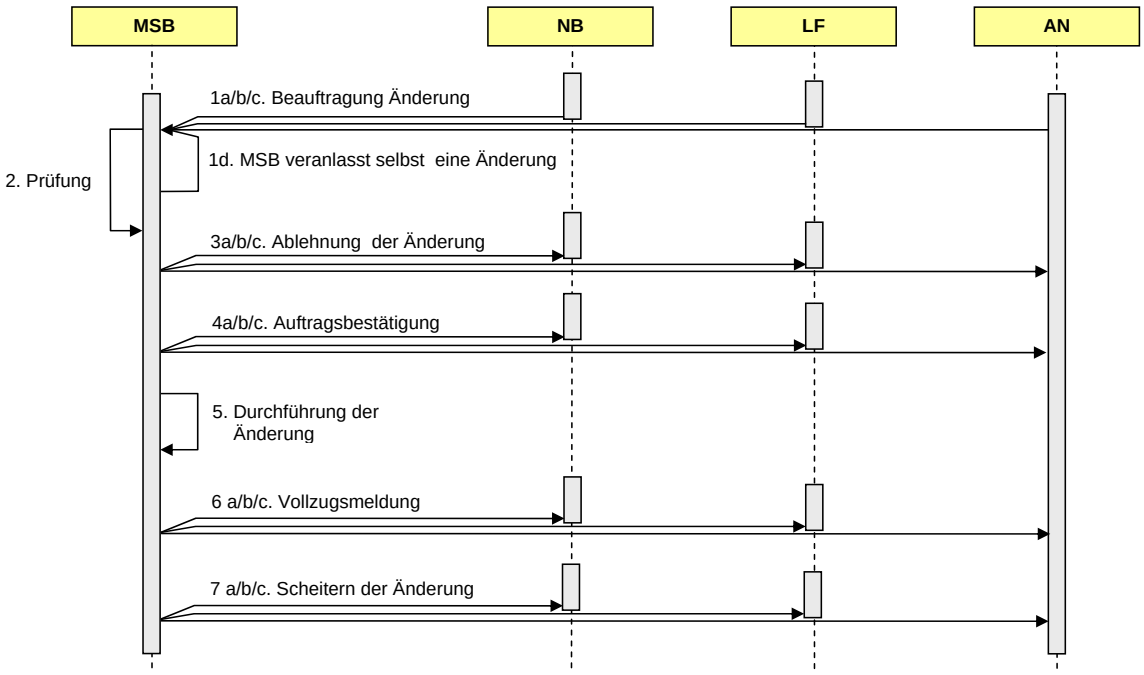

Figure B.119: Model 10-9 originally provided Sequence Diagram.

The fault detector reports a failure to the MPO or MPO has a suspicion of their own fault. The MPO shall examine the failure. The MPO rejects the failure of the fault detector or the MPO confirms the failure of the fault detector. If the MPO confirms the failure of the fault detector, he informes the GO and the MSP. The MPO fixes the fault at the measuring device. The MPO shares the results of the repairs carried out with the fault detector. The MPO will inform the GO about the resolution of the interference. The MPO will inform the MSP about the resolution of the interference.

Text 42: Process Description 10-10: Process C2.

The GO requests the measurements of the MSP. The MSP checks the received request. The MSP denies the request of the GO or the MSP performs the measurement. The MSP informs the GO about the failure of the reading or the MSP transmits the measured values to the GO. The GO processes the measured values. The GO sents the changed values to the MSP. The GO transmit the readings to the SP.

Text 43: Process Description 10-11: Process C3.

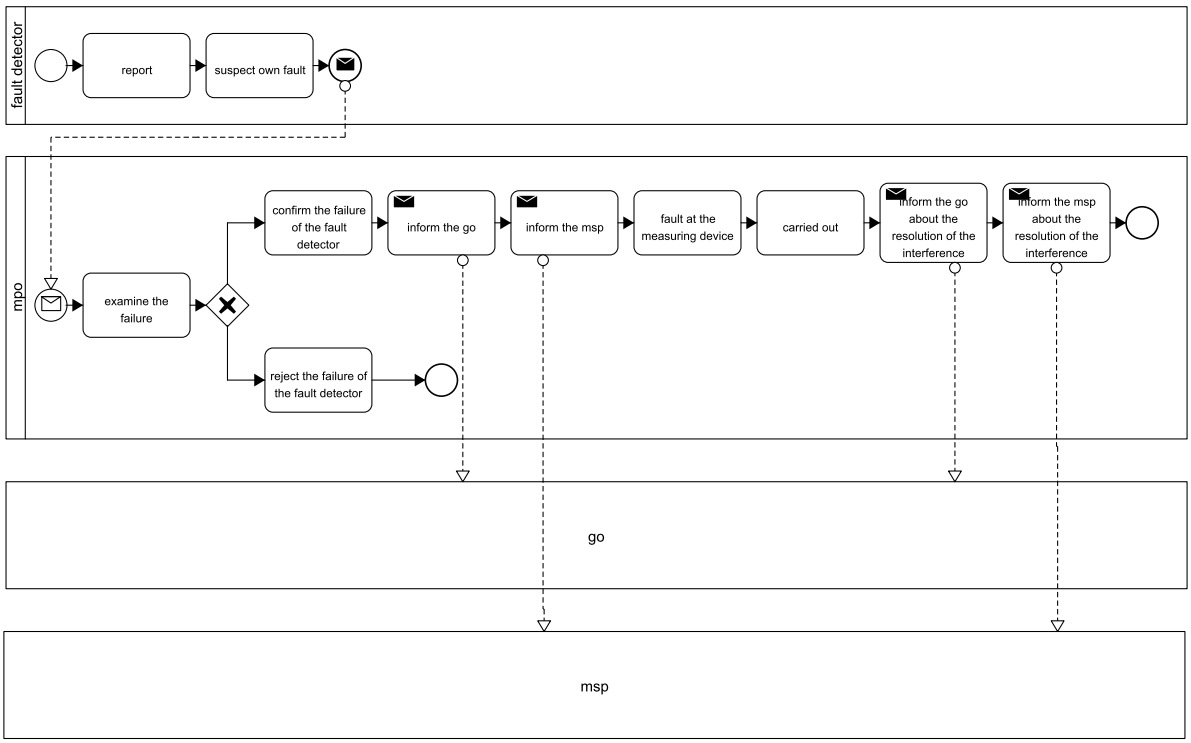

Figure B.120: Model 10-10 as generated by our system.

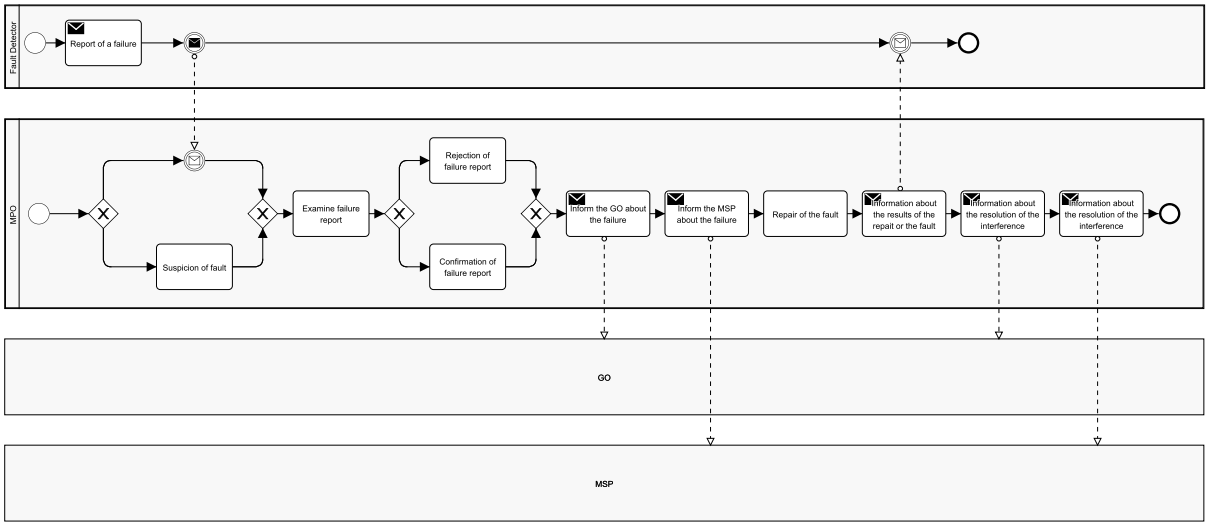

Figure B.121: Model 10-10 Sequence Diagram transformed to BPMN.

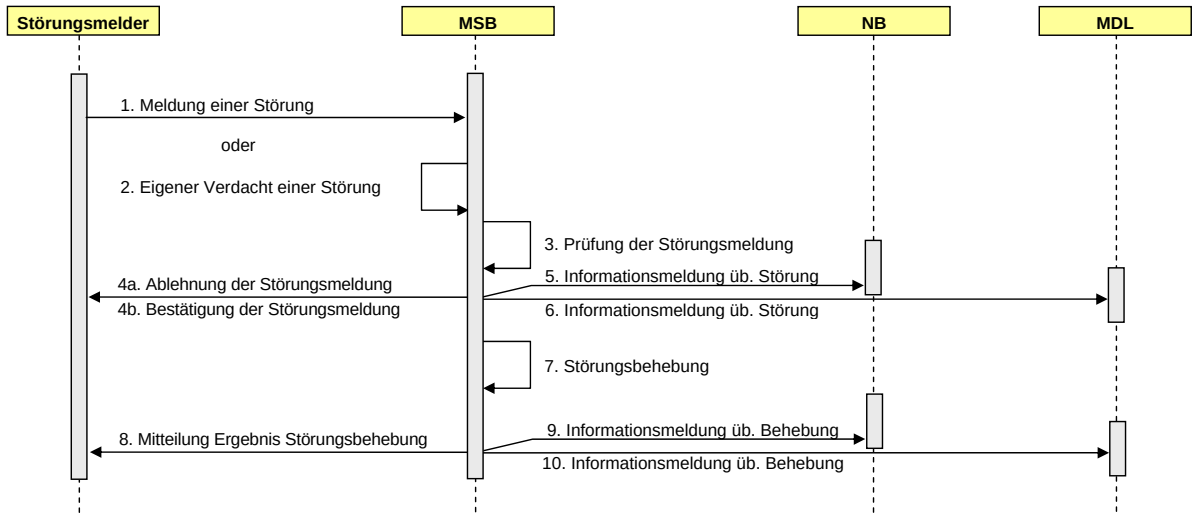

Figure B.122: Model 10-10 originally provided Sequence Diagram.

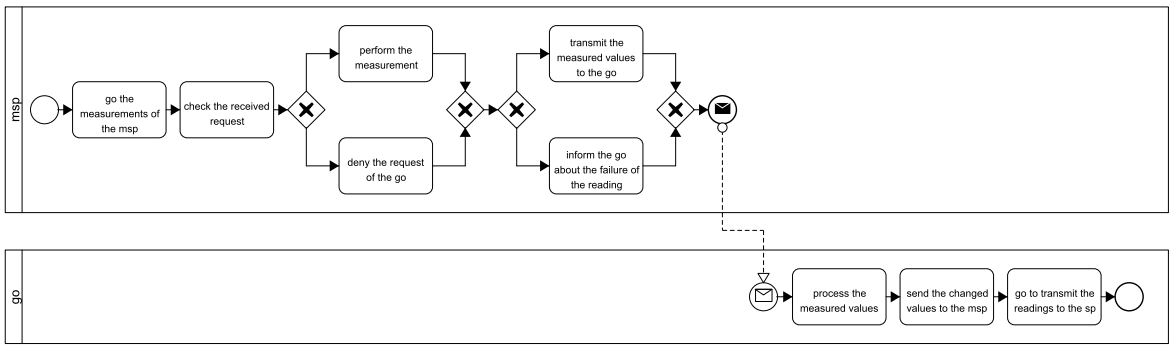

Figure B.123: Model 10-11 as generated by our system.

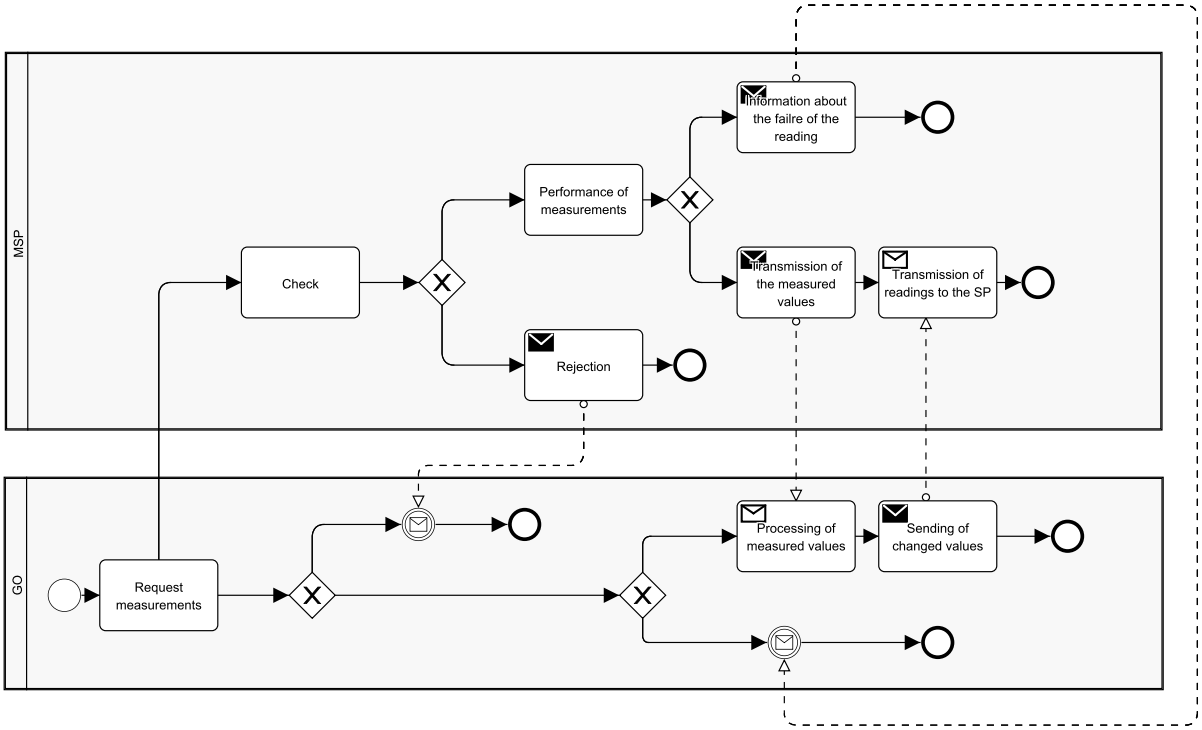

Figure B.124: Model 10-11 Sequence Diagram transformed to BPMN.

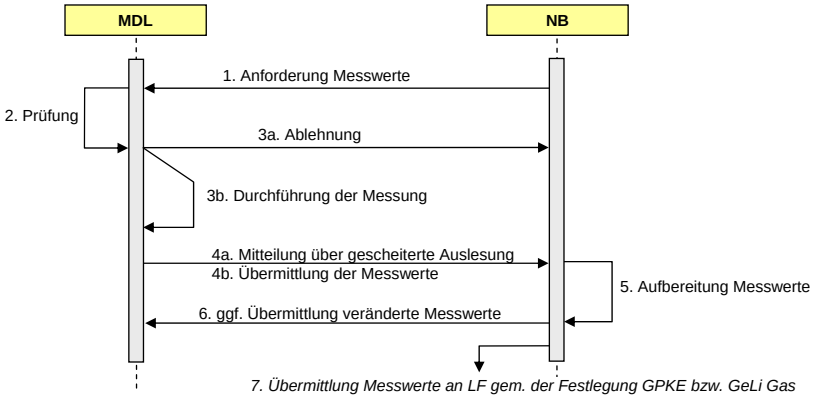

Figure B.125: Model 10-11 originally provided Sequence Diagram.

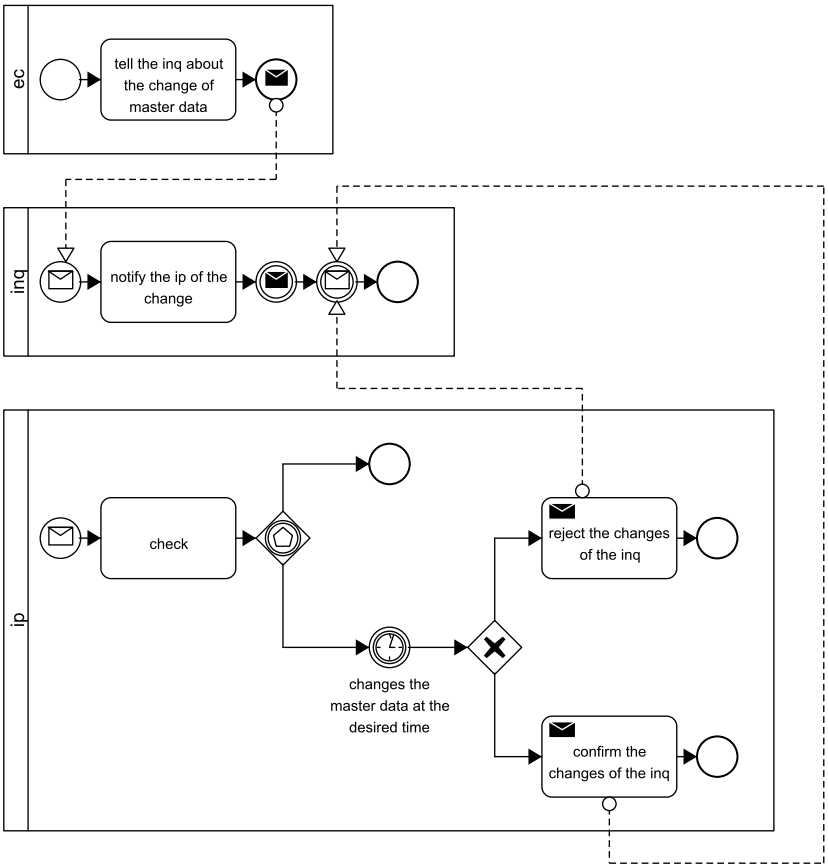

Figure B.126: Model 10-12 as generated by our system.

The EC tells the INQ about the change of his master data. The INQ notifies the IP of the change. The IP checks whether the master data can be changed at the desired time. The IP confirmes the changes of the INQ or the IP rejectes the changes of the INQ.

Text 44: Process Description 10-12: Process D1.

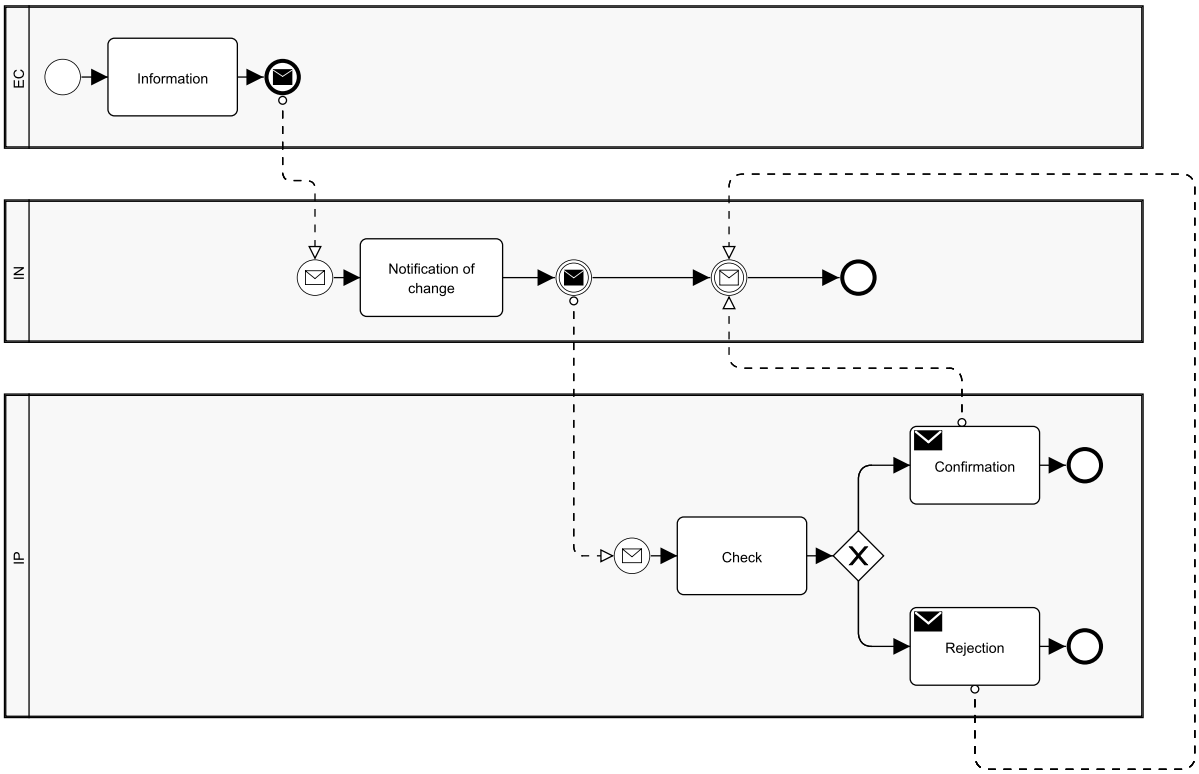

Figure B.127: Model 10-12 Sequence Diagram transformed to BPMN.

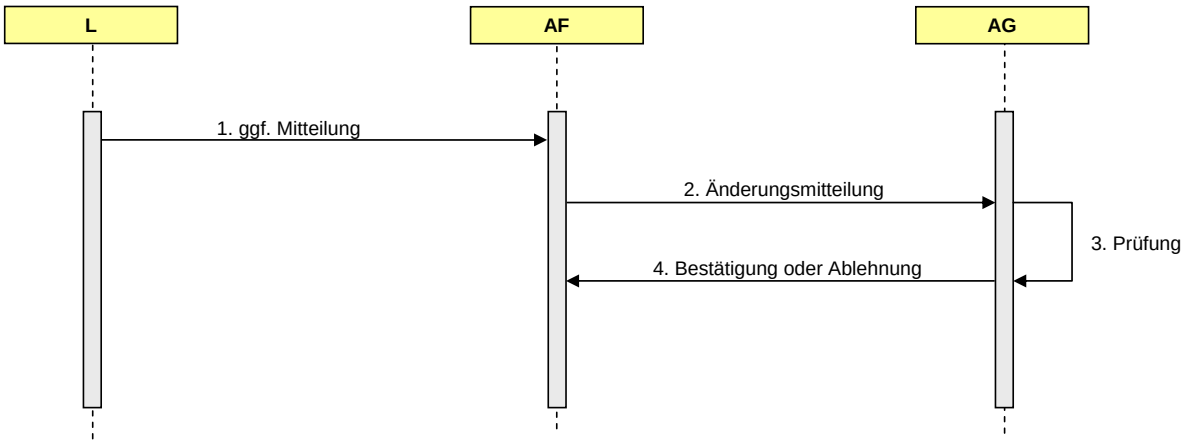

Figure B.128: Model 10-12 originally provided Sequence Diagram.

The INQ transmits the transaction data request to the IP. The IP checks the request of the INQ. The IP answers the question of the INQ depending on the outcome of the examination, i.e. Transmission of data or rejection.

Text 45: Process Description 10-13: Process D2.

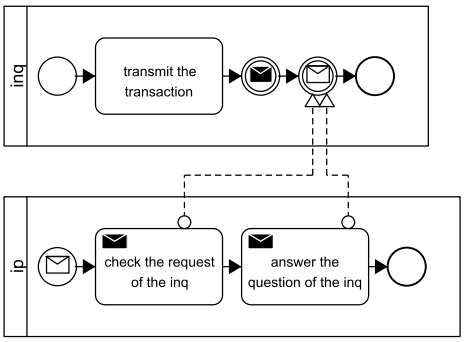

Figure B.129: Model 10-13 as generated by our system.

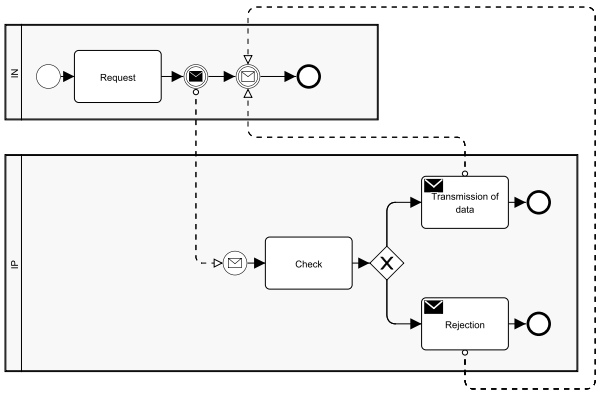

Figure B.130: Model 10-13 Sequence Diagram transformed to BPMN.

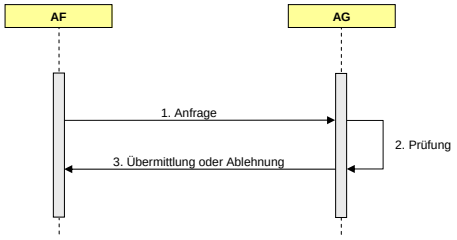

Figure B.131: Model 10-13 originally provided Sequence Diagram.

If the MPOO sends the bill for the temporary continuation of the metering point operations to the GO, the GO examines the bill. If the MSPO sends the bill for the temporary continuation of the measurement to the GO, the GO examines the bill. If the MSPO sends the bill for additional readings to the GO, the GO examines the bill. If the MPOO sends the bill for the equipment acquisition to the MPON or the GO, the MPON or the GO examines the bill. The GO or the MPON confirms the invoice with payment advice to the MPOO or the MSPO, or the GO or the MPON rejects the invoice of the MPOO or the MSPO.

Text 46: Process Description 10-14: Process D3.

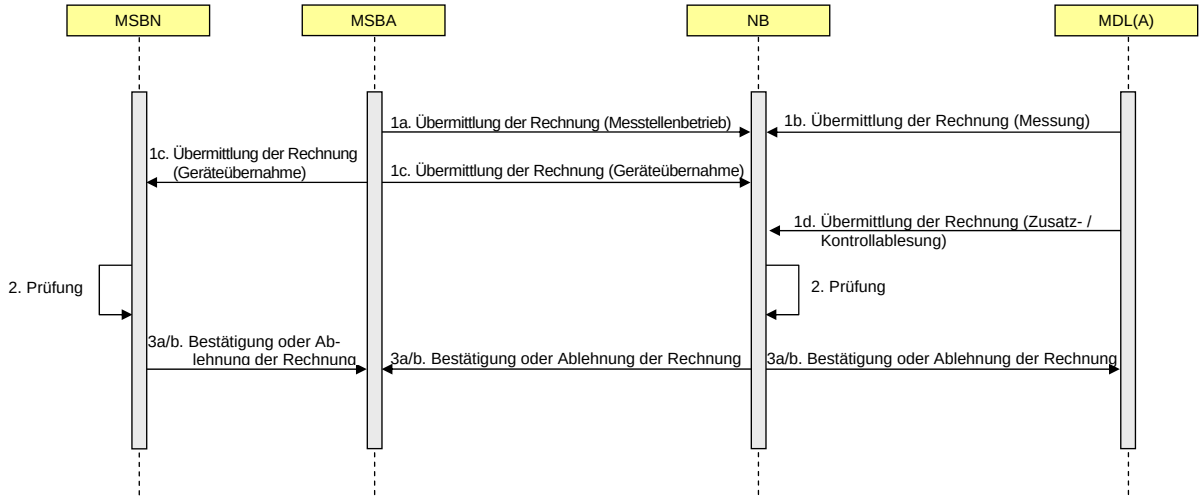

Figure B.132: Model 10-14 originally provided Sequence Diagram.

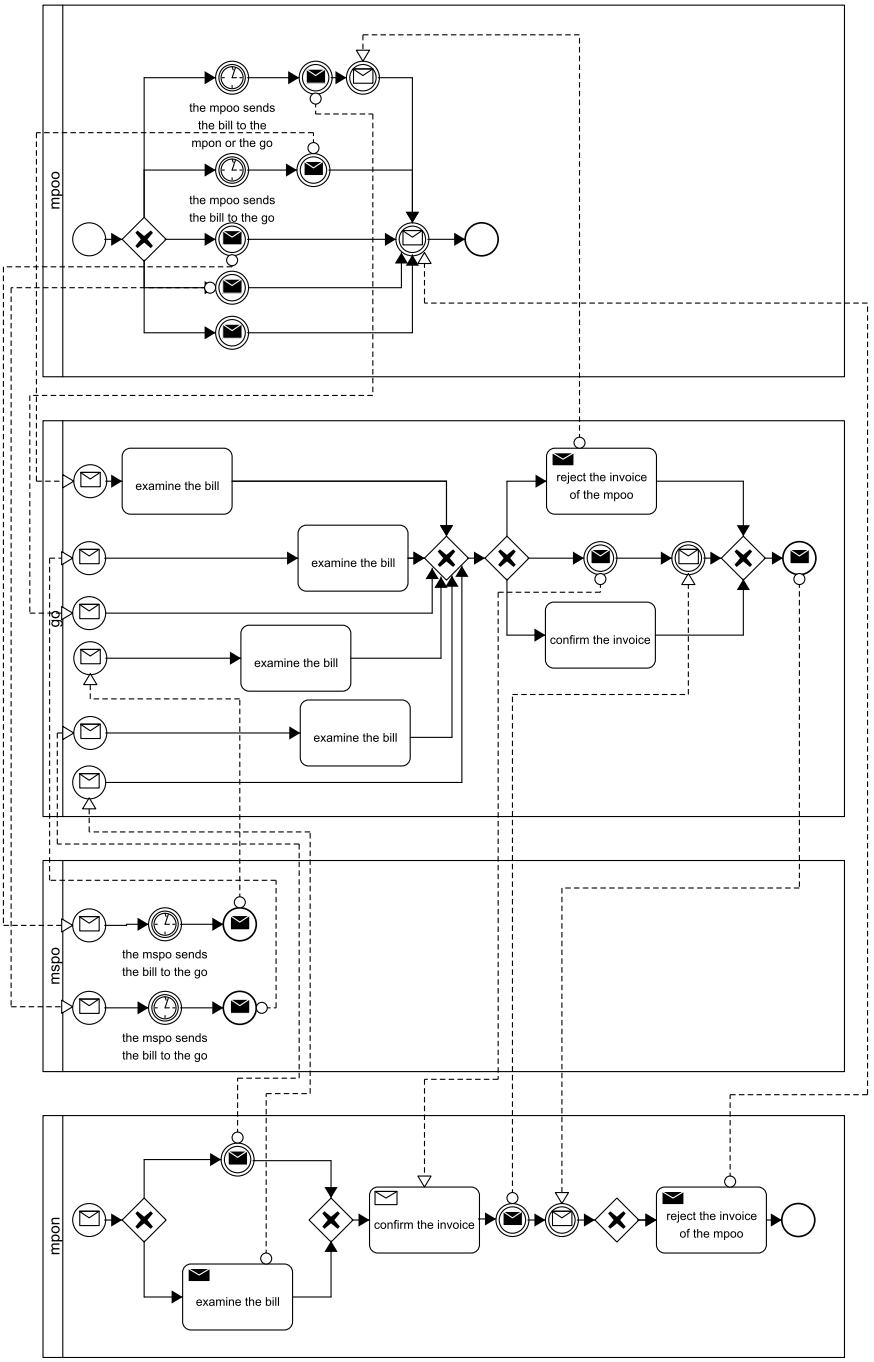

Figure B.133: Model 10-14 as generated by our system.

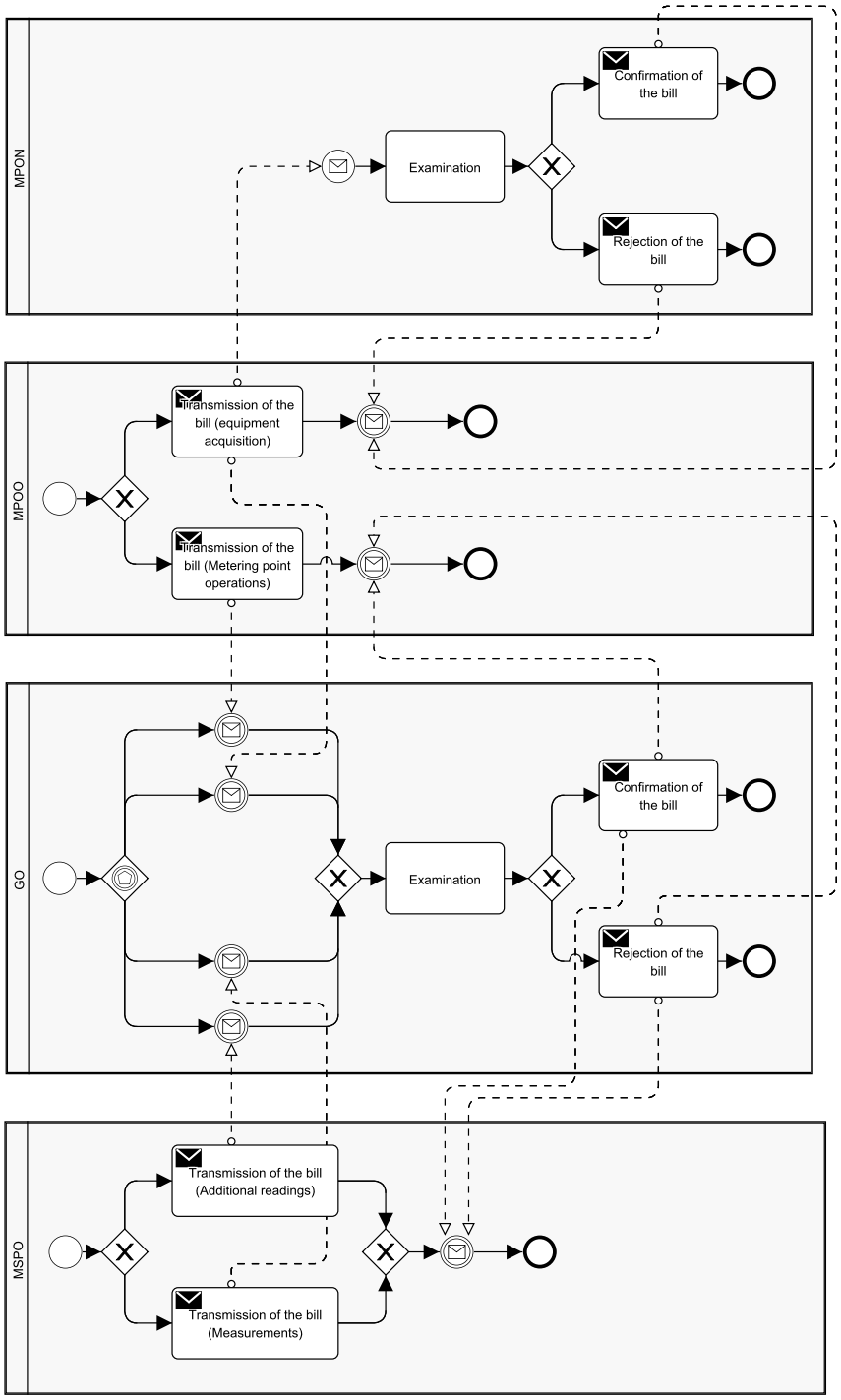

Figure B.134: Model 10-14 Sequence Diagram transformed to BPMN.

Appendix C. Employed Stop Word Lists

This section lists all stop word lists which were used during the transformation as described in section 3. The words were obtain driven by the exploratory analysis of the process models and by consulting the book "Practical English Usage" [123]. All lists can be accessed and modified by the user in our research prototype.

Person Corrector List:

1. resource provisioning
2. customer service
3. support
4. support office
5. support officer
6. client service back office
7. master
8. masters
9. assembler ag
10. acme ag
11. acme financial accounting
12. secretarial office
13. office
14. registry
15. head
16. storehouse
17. atm
18. crs
19. company
20. garage
21. kitchen
22. department
23. ec
24. sp
25. mpo
26. mpoo
27. mpon
28. msp
29. mspo
30. mspn
31. go
32. pu
33. ip
34. inq
35. sp/pu/go
36. fault detector

Conditional Indicator:

1. if
2. whether
3. in case of
4. in the case of
5. in case
6. for the case
7. whereas
8. otherwise
9. optionally

Parallel Indicator:

1. while
2. meanwhile
3. in parallel
4. concurrently
5. meantime
6. in the meantime

Sequence Indicator:

1. then
2. after
3. afterward
4. afterwards
5. subsequently
6. based on this
7. thus

Weak Verbs:

1. be
2. have
3. do
4. achieve
5. start
6. exist
7. base

Forward Link Indicator:

1. finally

Loop Indicator:

1. next
2. back
3. again

Frequency Words:

1. usually
2. normally
3. often
4. frequently
5. sometimes
6. occasionally
7. rarely
8. seldom

Data Object Determiner:

1. written material
2. record
3. message
4. design

Appendix D. Description of the implemented Prototype

As mentioned in section 3, our transformation approach was implemented in a research prototype. A screenshot of the graphical user interface is displayed in figure D.135. It consists of three parts, as marked in the illustration.

Part 1 shows the processed text and highlights the extracted elements within each sentence. Furthermore, the result of the anaphora resolution and textual link determination are shown as arcs connecting the different words. Using the buttons above the model those arcs can be hidden, a new text can be loaded, or the comparison procedure can be triggered. If the result of the anaphora resolution is not satisfying for the user, he is able to alter the target of those arcs and thus the resolution results. Furthermore, it is possible to save the annotated text to an XML format. By clicking on a sentence it can be inspected in more detail.

Part 2 in the upper right area shows the syntax tree of the currently selected sentence as determined by the Stanford Parser. Below, all typed dependency relations are listed to enable an easy analysis of the current sentence.

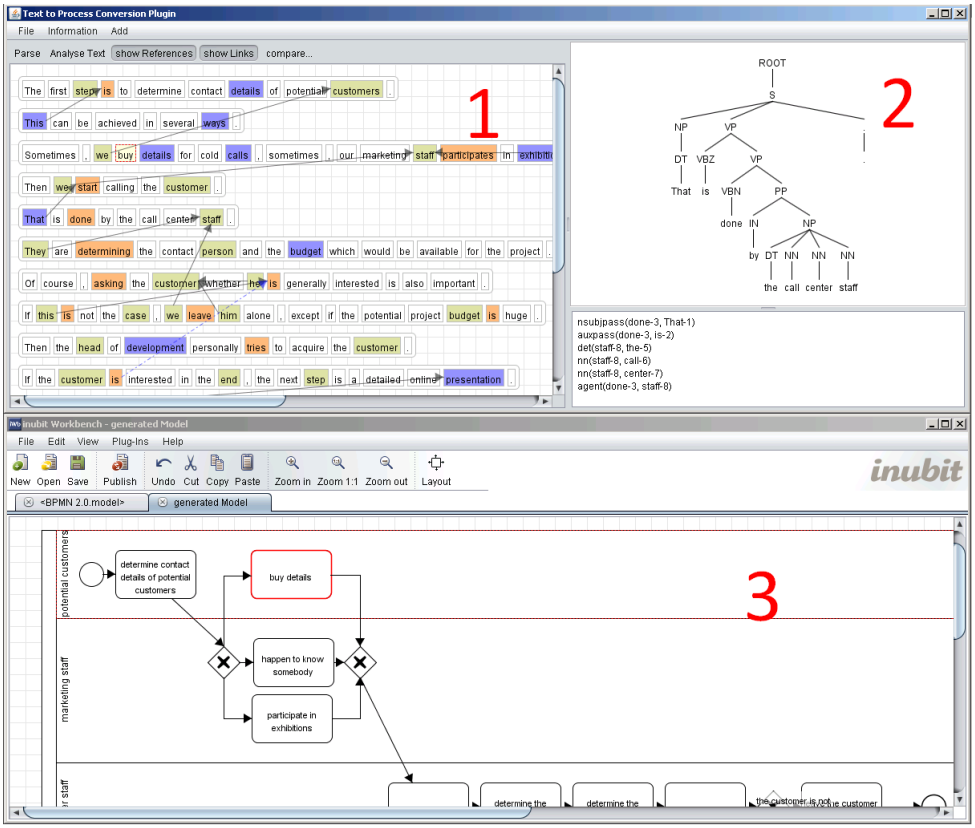

Figure D.135: Graphical user interface of the implemented research prototype.

Part 3 is the inubit workbench an industrial research prototype, where the generated model is displayed. Furthermore, the Lane split-off procedure is implemented here. Therefore, the user is able to quickly adapt the model to his needs, add or deleted nodes and edges, and export it in different formats to use it in other tools. All generated models depicted in this thesis were created using this research prototype.

Additionally, the tool is able to automatically compare two models and to calculate their graph edit distance and similarity. Traceability is also ensured. Whenever the user clicks on a Task node in the model, the corresponding element in the text will be highlighted and vice versa. Currently, the Task "buy details" is highlighted as the user has clicked on the word "buy" in the text.

Declaration of Authorship

I hereby confirm that I have authored this Master Thesis independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such. I am aware of the examination regulations. Until now, I neither submitted nor finally failed a Master Thesis within my course of studies.

Berlin, 29th November 2010

Fabian Friedrich