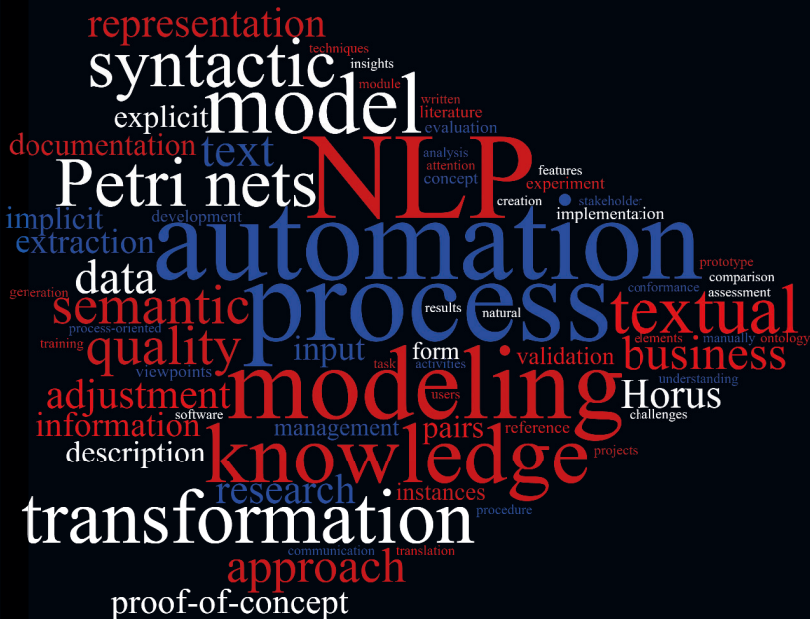


Text to Process Model: Automating Process Model Creation from Text

Felix Reinold Nolte



Text to Process Model: Automating Process Model Creation from Text

Inauguraldissertation zur Erlangung des akademischen Grades
eines
Doktors der Wirtschaftswissenschaften durch die
Wirtschaftswissenschaftliche Fakultät der
Westfälischen Wilhelms-Universität Münster

Vorgelegt von

Felix Reinold Nolte, MSc
aus Stadtlohn

Dezember 2020

Dekan	Prof. Dr. Gottfried Vossen
Erster Gutachter	Prof. Dr. Gottfried Vossen
Zweiter Gutachter	Prof. Dr. Dr. h.c. Dr. h.c. Jörg Becker
Dritter Gutachter	Prof. Dr. David Bendig
Mündliche Prüfung	08.02.2021

Felix Reinold Nolte

**Text to Process Model: Automating Process
Model Creation from Text**



Wissenschaftliche Schriften der WWU Münster

Reihe IV

Band 20

Felix Reinold Nolte

Text to Process Model: Automating Process Model Creation from Text

Wissenschaftliche Schriften der WWU Münster

herausgegeben von der Universitäts- und Landesbibliothek Münster
<http://www.ulb.uni-muenster.de>



Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <https://www.dnb.de> abrufbar.

Dieses Buch steht gleichzeitig in einer elektronischen Version über den Publikations- und Archivierungsserver der WWU Münster zur Verfügung.

<https://www.ulb.uni-muenster.de/wissenschaftliche-schriften>

Felix Reinold Nolte

„Text to Process Model: Automating Process Model Creation from Text“

Wissenschaftliche Schriften der WWU Münster, Reihe IV, Band 20

Verlag readbox unipress in der readbox publishing GmbH, Dortmund

www.readbox.net/unipress

Zugl.: Diss. Universität Münster, 2021

Dieses Werk ist unter der Creative-Commons-Lizenz vom Typ 'CC BY-SA 4.0 International'

lizenziert: <https://creativecommons.org/licenses/by-sa/4.0/deed.de>

Von dieser Lizenz ausgenommen sind Abbildungen, welche sich nicht im Besitz des Autors oder der ULB Münster befinden.



ISBN 978-3-8405-0256-9 (Druckausgabe)
URN urn:nbn:de:hbz:6-77079625138 (elektronische Version)

direkt zur Online-Version:

© 2021 Felix Reinold Nolte

Satz: Felix Reinold Nolte
Titelbild: Felix Reinold Nolte (WordCloud)
Umschlag: ULB Münster



Acknowledgements

First and foremost, I would like to thank my parents, Martin Nolte and Gabriele Worth-Nolte, who not only made sure that under their guidance I grew into the person that was able to go this way until the end, but also gave me all their support and love along this specific part of my life. This includes my sister, Carla Nolte, who was always giving me the feeling that I can prevail on the way I have chosen to pursue and at the same time reminded me of the fact that life consists of way more than work and research.

It is difficult to put my gratitude into words for Gülşah Özdil, the love of my life, who was at my side throughout my doctorate studies with love, patience, and support that goes beyond what one could ask for and imagine. She gave me purpose and made the physical distance between us feel as merely a formality. Words cannot capture how happy and grateful I am that she was at my side all this time.

I want to thank Julia Seither for being of big help with any issue and question of organizational nature, but most importantly also for taking care that I was able to keep myself together in difficult times, Dr. Denis Martins and Raquel Mello for all their effort and commitment to support me, not only as colleagues and in research but as friends, Nico Grohmann for all the times when I was visiting his office to either discuss our research or to just talk

about football as a welcoming distraction, Jan Everding, who with his experience and humor encouraged me to get into things instead of overthinking the planning, Dr. Leschek Homann, who wrote his thesis in parallel to me and of whom I was lucky to benefit from his experience and who reminded me from time to time to take things more seriously, Ralf Farke for his technical support and the interesting conversations, Dr. Jens Lechtenböcker for his feedback on my research and all his support in teaching, Dr. David Fekete, Dr. Nicolas Pflanz, and Dr. Fabian Schomm-von-Auenmüller for their support in my early days at the DBIS Group.

Furthermore, I want to express my gratitude towards Dr. Frank Schönthaler, Dr. Thomas Karle, Dr. Mana Taghdiri, and Johannes Micheler, who gave me the opportunity to work on an industry-relevant problem and provided me with valuable feedback along the way.

I would like to thank my doctoral supervisor Prof. Dr. Gottfried Vossen for his support throughout my studies and research, Prof. Dr. Jörg Becker for acting as a second reviewer of this thesis and Prof. Dr. David Bendig for acting as a third reviewer for the defense of this thesis.

Not directed to any specific person in name, I want to thank the Institute of Information Systems at the University of Münster, which I spend almost ten years at as a bachelor, master, and lastly doctorate student. I always felt welcomed, supported, and as a part of the institute.

Münster, April 2021

Felix Reinold Nolte

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Research Question and Goal	8
1.3	Thesis Structure	11
1	Foundations	13
2	Process Modeling and Knowledge Management	19
2.1	Basics of Business Process Modeling	19
2.1.1	Business Processes	21
2.1.2	Business Process Models	22
2.1.3	Business Process Modeling Languages	22
2.1.4	Imperative and Declarative Modeling	25
2.2	Business Modeling with Petri Nets	27
2.3	Quality of Business Process Models	31
2.3.1	Quality Frameworks and Guidelines	34
2.3.2	Quality Metrics	44
2.4	Knowledge Management via Models	47
2.4.1	Knowledge Management	48

2.4.2	Business Process Modeling in Knowledge Management	66
2.5	Model Generation and Transformation	73
3	Natural Language Processing	77
3.1	Fundamentals	77
3.1.1	Historical Development	80
3.1.2	Theoretical Foundations	82
3.1.3	Tasks and Challenges	86
3.2	Examples	96
3.2.1	Industry	97
3.2.2	Academia	99
3.3	Related Concepts	101
3.4	Related Implementations	104
II	Automated Model Creation from Text	109
4	Problem Specification	111
4.1	Related Work	111
4.1.1	Existing Approaches	112
4.1.2	Related Approaches	124
4.2	Challenges	130
4.2.1	Information Acquisition	130
4.2.2	Language Specifics	133
4.2.3	Petri Net and Horus Model Specifics	139
4.2.4	Modeling Conventions	143
4.3	Contribution	146

5	Context Description and Design	151
5.1	Horus Method	151
5.1.1	Horus Business Modeler	155
5.1.2	Horus Procedure Models	157
5.2	Concept	157
5.2.1	Objectives	161
5.2.2	Methods and Techniques	164
5.2.3	Reference Point to Contextual Knowledge Base	165
5.2.4	Transformation Approach	186
6	Implementation and Evaluation	207
6.1	Scope	208
6.2	Architecture	209
6.2.1	Pipeline	209
6.2.2	Input Requirements	210
6.2.3	Artifacts	211
6.3	Features	213
6.3.1	Natural Language Text Pre-Processing . . .	213
6.3.2	Linguistic Feature Extraction	218
6.3.3	Model Element Mapping	233
6.3.4	Process Model Generation	251
6.4	Evaluation	255
6.4.1	Experimental Setup	256
6.4.2	Summary of Results	260
6.4.3	Discussion	263

III Closing Remarks	275
7 Conclusions	277
7.1 Summary	277
7.2 Implications	279
7.3 Outlook	282
Bibliography	287
List of Abbreviations	317

List of Figures

1.1	Example Transformation: Text to Model	4
1.2	Thesis Structure	12
1.3	Thesis Foundations	17
2.1	Purposes of Business Process Models	23
2.2	Petri Net Patterns	31
2.3	Semiotic Ladder	37
2.4	Guidelines of Modeling	40
2.5	The SEQUAL Framework	42
2.6	From Data to Knowledge	53
2.7	The SECI Model	55
2.8	Pillars of Knowledge Management	62
2.9	The Four Steps of Knowledge Conversion	63
2.10	The Four Dimensions of Core Capability	64
2.11	Knowledge Management Assessment Tool	65
2.12	Process-oriented Knowledge Management	68
3.1	Taxonomy of Natural Language Processing	78
3.2	A sample three-layer Neural Network	84
3.3	Part-of-Speech (POS)-Tagging with StanfordCoreNLP	91
3.4	Named Entity Recognition	91

List of Figures

3.5 Constituency Parsing with StanfordCoreNLP 93

3.6 Dependency Parsing with StanfordCoreNLP 93

3.7 Semantic Role Labeling with StanfordCoreNLP 94

3.8 Coreference Annotation with StanfordCoreNLP 95

3.9 Machine Translation on the Example of Google Translate 99

4.1 Steps of the Sentence Level Analysis 114

4.2 Method for Process Elicitation 118

4.3 Subset of BPMN Elements 121

4.4 Natural Language Generation System 126

4.5 Process Architecture of UMGAR 127

4.6 Comparison to a Text Mining System 149

5.1 Phases of the Horus Method 153

5.2 Horus Perspectives 154

5.3 Model created with the Horus Business Modeler 155

5.4 Exclusive Choices in a traditional Petri Net 158

5.5 Exclusive Choices in a Horus Procedure Model 158

5.6 Running Example based on the Introduction 159

5.7 Process Steps of the Transformation Approach 160

5.8 Example of an Ontology 170

5.9 Key Features of Concept Maps 172

5.10 Example of a Knowledge Graph 174

5.11 Ontology-based Schema of the Knowledge Base 176

5.12 Example of a Knowledge Graph 178

5.13 One Instance of the Knowledge Graph 179

5.14 Second Instance of the Knowledge Graph 180

5.15 Syntax Tree of the Example Sentence 188

5.16 Inputs and Outputs based on a "creating" Verb 189

5.17	Inputs and Outputs based on a "processing" Verb	189
5.18	Example of a Sequence	194
5.19	Example of a Parallel Split	197
5.20	Example of an Exclusive Choice	197
5.21	Example of a Synchronization	198
5.22	Example of a Simple Merge	198
5.23	A Data Frame of a Horus Procedure Model	200
5.24	Conflict Resolution based on Object Occurrence	202
5.25	Conflict Resolution based on Activity Types	202
6.1	Implementation Pipeline	210
6.2	Object Types and Flow	212
6.3	Example Test Model for Evaluation	258
6.4	Example Generated Model for Evaluation	259
6.5	Petri Net Generation including a Loop	263
6.6	Incorrect Identification of an Exclusive Split	263
6.7	Textual Input and Generated Process Model	268

List of Tables

2.1	Model Quality Frameworks	35
2.2	List of Quality Metrics	46
2.3	Knowledge Management Frameworks	66
2.4	Knowledge Management in the Project Context	74
3.1	Information Extraction Tasks and Techniques	96
3.2	Information Extraction in existing Solutions	108
4.1	Overview: Related Work	113
4.2	Topics in Related Work	150
5.1	Model Elements from the first Sentence	195
5.2	Model Elements from the second Sentence	196
6.1	Output Data Frame Header Feature Extraction	219
6.2	Extraction of Subject, Verb, Object (SVO)-tuples	222
6.3	Extraction of Prepositional Phrase (PP)-tuples	224
6.4	Extraction from an Adverbial Clauses	227
6.5	Extraction from a Conditional Clauses	229
6.6	Extraction from a Relative Clause	231
6.7	Extraction from the Core Sentence	233

List of Tables

6.8 Output Data Frame Header Mapping 234
6.9 Mapping the Core Sentence 243
6.10 Mapping a Conditional Clause 245
6.11 Mapping a Adverbial Clause 247
6.12 Mapping a Relative Clause 249
6.13 Recall and Precision of the evaluated Models 261

List of Listings

6.1	Coreference Resolution	216
6.2	Sentence Splitting	217
6.3	Extraction of Subject, Verb and Objects	220
6.4	Extraction from a Prepositional Phrases	223
6.5	Extraction of Objectroles	224
6.6	Extraction of Adverbial and Conditional Clauses	226
6.7	Extraction from an Adverbial Clauses	226
6.8	Extraction from a Conditional Clauses	228
6.9	Extraction of a Relative Clause	229
6.10	Extraction from a Relative Clause	231
6.11	Extraction from the Core Sentence	232
6.12	Acquisition of the Antonym of a Word	240
6.13	Mapping the Core Sentence	242
6.14	Mapping a Conditional Clause	245
6.15	Mapping a Adverbial Clause	247
6.16	Mapping a Relative Clause	249
6.17	Combination of Mapping Results	252
6.18	Checking for Disconnected Parts	254
6.19	Connecting Model Parts	255

1 Introduction

The description and visualization of processes are among the core tasks in a variety of IT projects today. The translation of information between different forms of representation aims to integrate information in the different steps of a project efficiently. For instance, it is used to describe the as-is and to-be states in a commonly understandable manner. However, transforming a textual description into a visual representation of a business process model is still a costly and complex task that often comes with an inevitable loss of information.

To address shortcomings in modeling processes, different techniques of Natural Language Processing (NLP) are combined to analyze textual descriptions of a process to enable a transformation of texts into process models. The performed steps of analysis and transformation ensure conformance between process models and their textual description. With the help of an ontology and corresponding knowledge graphs generated from positive examples, a reference point is used to improve the transformation by providing additional information about the context the model is created in. In the upcoming sections, the motivation for this project, the research questions, and the structure of this thesis will be described. Lastly, an overview of the remainder of this thesis is given.

1.1 Motivation

Process modeling has grown into one of the main aspects of business process management and the business process life cycle. Embedded into different phases of an IT project, process modeling is used to support conceptualization and documentation, serving as a reference to the employees or enabling communication between stakeholders.

Even though the topic of process modeling is not new and has been extensively investigated since the 1990s, the modeling process itself remains costly and time-consuming for the staff involved. Next to the creation itself, ensuring that a model is of good quality and representing the circumstances and contents it is supposed to show is challenging. Misunderstandings between a process modeler and a domain expert or the lack of accessible information additionally hinder the creation of process models [Ber07]. The latter will typically provide a textual description of a process, which the former has to transform into a formal model.

However, both typically lack the other side's expertise, which is why the texts are often not written in a process-oriented pattern. Simultaneously, the models often miss implicit information hidden in the written text and fail to represent their content correctly. Even though there are already guidelines, frameworks, and best practices to support process modeling and ensure the mentioned quality and consistency of process models, the shortcomings did not vanish and still are the reason for additional costs. Consequently, supporting the translation from a textual description into a process model leaves significant room for improvement; to make a step in this direction is the goal of this thesis.

An automated way of creating process models from textual input to reduce modeling tasks for the user is presented to achieve this goal. Different related fields already investigate the problem and potentials at hand, such as text mining, process mining, process modeling recommender, social business process modeling, and insight from related work will be investigated and incorporated later on in Section 4.1.

One of the functional application areas of process modeling is Knowledge Management (KM), which has been gaining increased attention in the literature from different viewpoints since the early 2000s [KB06; JHS01; RL00]. With the rise of digital transformation and the increase of digital capabilities, new ways of capturing, storing, and sharing knowledge were discovered. They gave momentum to new approaches and extensions to Knowledge Management (KM).

One of the rather recent approaches aims to integrate KM activities into an organization's process landscape. In such process-oriented knowledge management, process models can store knowledge in a visualized and process-related manner. Knowledge residing in companies is often needed on the one hand for the realization of projects and on the other as part of product and service.

Motivating Example

As an example, consider a simplified process of checking a received lead. From asking one of the involved employees, the textual process description seen as input in Fig. 1.1 was obtained. Based on this textual description, a process model, more specifically, a specialized Petri net in our case, should be created automatically. The intended output is depicted in Fig. 1.1.

Although a simplified example is used throughout this paper, it is considered that its characteristics resemble the complex problems tackled by this approach. However, more extensive models and their textual descriptions can also be processed up to the limits described later.

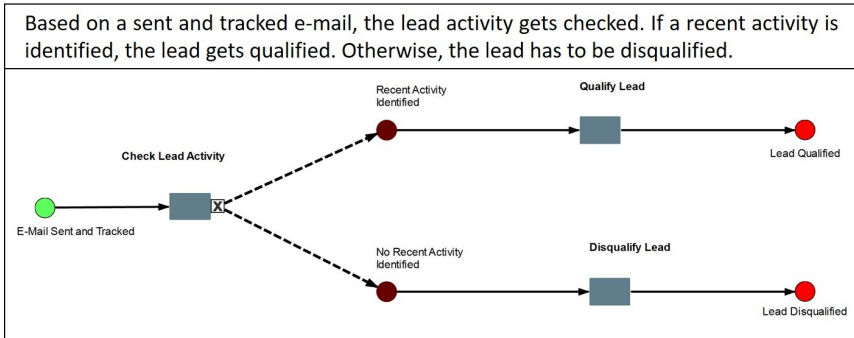


Figure 1.1: Example Transformation: Text to Model.

In the approach described in this thesis, the importance, but also the deficiencies of process modeling as means of representing knowledge are considered by presenting a prototype to translate knowledge residing in process descriptions into process models. The transformation further is intended to support employees in the creation and maintenance of knowledge bases using automated adjustments to the knowledge artifacts in the form of process models.

To this end, the focus is put on Horus procedure models as a specialization of Petri nets that are part of the Horus method. The Horus method incorporates different kinds of models to capture knowledge about an organization's business processes and related aspects, such as systems, staff, and resources. It focuses on practices

that support these processes over their whole life cycle. Maintaining the knowledge bases within a company supported by this approach should consistently lead to a more comprehensive body of knowledge that is also of higher quality. An ontology-based collection of positive examples of process models and fitting textual descriptions is used to enhance the transformation steps. With the help of the reference point, extracted information about process model elements can be validated or even extended by missing or implicit information that is not identified during our text analysis.

The resulting approach additionally supports the assessment of the quality of the mentioned Horus procedure models with the help of a comparison of manually created and automatically generated models. The degree of conformance between both models offers information about the quality of either text or process model. The task of translating a model into text and vice versa, and capturing the semantics of a model, is already acknowledged in the literature as a significant challenge nowadays and gained increased attention over the last years. With the automatic generation of models from text, three objectives are aimed at that capture the potential of enhancing the process of modeling:

1. **Automation:** Reduce human involvement in the process of modeling. The optimal goal is the full automation of creating a correct process model from a textual input. However, this is a difficult task to solve. This approach aims to reduce human interaction in the modeling process, meaning that some human interaction might still be necessary or advisable for a better result with the developed approach.

2. **Quality:** Increase the quality of process models, especially when it comes to consistency and correctness. Different dimensions of quality, foremost the semantic and syntactic dimensions, are explored, while the transformation addresses especially the semantic dimension due to the lack of recognition and attention it experienced up to now.
3. **Training:** With a transparent description of the transformation process and the included steps, inexperienced process modelers can be introduced to the topic of process modeling without the introduction of any specific tool or environment. The required textual input can be written by anyone, preferably with a process-oriented structure already in mind, and no need to get familiar with modeling conventions and specific modeling languages (even though just Petri nets are included).

Business Problem

Knowledge Management plays a significant role in consultancies, such as the business partners PROMATIS and Horus software GmbH involved in this project. PROMATIS' core business focuses on consulting companies on corporate strategies and business management combined with IT concepts and implementation-oriented project management. Horus, as a subsidiary company of PROMATIS, is a software company that has developed a process modeling software for businesses and private users in the form of the Horus Business Modeler (HBM) following the Horus method.

Knowledge in the case of both companies is needed on the one hand for the realization of projects and, on the other hand, as part of product and service. The scope and quality of the knowledge that is bundled in a Knowledge Base is of significant relevance for both use cases, and continuous quality control of the knowledge bases is desired and necessary.

As one method to manage knowledge integrated into the relevant processes, process-oriented knowledge management examines the necessity for the orientation of knowledge management to the business processes. Its goal is the recognition and further development of knowledge processing in the operative business processes. To this end, both the process flow and the knowledge management activities along the processes are examined in this approach.

Even though the knowledge bases can consist of various artifacts, process models created with the HBM represent the preferred used representation form for both business partners' knowledge. Focusing on process modeling with the HBM, which builds on Petri nets as an underlying modeling language, the concept considers using a typical development cycle of a model during a project. The automated transformation between text and model should contribute to a precise creation, maintenance, and use of the knowledge bases.

The approach described here aims at two scenarios: Process model as input as well as text as input. The latter scenario is at the core of this thesis. Nevertheless, the first scenario is relevant for the model adjustment step and provides different potentials to improve the different process modeling tasks during a process model's life cycle. Maintaining a company's knowledge bases should consistently lead to a more comprehensive body of knowledge and higher quality.

1.2 Research Question and Goal

For a thorough investigation and meaningful design and implementation of the approach described in this thesis, the problem statement and challenges coming with it have to be kept in mind. First, natural language texts that describe processes have to be analyzed to extract model elements. With the ability to identify model elements in textual descriptions, the translation of a text into process models can be addressed.

With the reference point's help, information about process model elements can be validated or even extended by missing or implicit information not identified during the text analysis. Results can support the assessment of a process model's semantic quality based on a comparison between the original process model and the from text generated process model. The degree of conformance between both models offers information about the quality of either the text or the model. The task of translating a model into text and vice versa, and capturing the semantic quality of a model, is already present in recent literature and research projects. Progress in this field is evident, but development is ongoing, and the translation between model and text is still a task with open challenges.

This thesis aims at answering the following research question:

How can Natural Language Processing (NLP) in combination with an ontology-based knowledge base be used to enable an automatic transformation of natural language texts into process models to support the process of modeling and assess as well as ensure the quality of a model?

The transformation approach between text and process models includes the step of *Model Adjustments*, which suggests adjustments to a Horus procedure model based on available contextual knowledge. As the input for this step is a process model, manually or automatically created, two possible executions of the approach are intended:

1. Following the main intention of this thesis, a textual input is provided that is transformed into a process model.
2. Focusing on the *Model Adjustments* an already existing process model can be used as an input, skipping the transformation steps. The process model can then be improved from this point on with the help of the reference point.

Insights gained from investigating the existing literature of this topic and related topics contribute to a deeper understanding of the semantic quality of process models and means to assess these quality aspects over the model generation and transformation from natural language texts. A description and explanation of how to evaluate a process model in the form of a specialized Petri net (Horus procedure models) towards its ability to represent the underlying content is one expected outcome.

With the help and the combination of techniques from the explored fields of *Process Modelling*, *Natural Language Processing* and *Model Transformation and Generation* and the use of an ontology-based reference point, the transformation of textual descriptions of processes into a Horus procedure model that represents this process is addressed. Using this transformation, both generated models and manually created models can be compared.

Lastly, the module contributes to an existing strategy to support users in creating correct and meaningful process models, including their attached elements, such as descriptions and related models. The implementation supports the combination of direct feedback about the syntactic and semantic qualities of models. It can contribute to different further activities, such as teaching users the modeling language or model-oriented thinking.

1.3 Thesis Structure

The remainder of this thesis is structured as follows and shown in Figure 1.2.

Part I provides an overview of the foundations of this thesis and focuses on the introduction into the related topics of Business Process Modeling (BPMoD), Knowledge Management (KM), and Natural Language Processing (NLP). An understanding of the three topics is established to, later on, follow with the connection of the three fields. Business Process Modeling (BPMoD) introduces a shared understanding of business processes, the corresponding models, and the idea of defining and measuring the quality of these. While the topic of KM represents the practical context for the business problem, NLP provides an understanding of the used technical approach that is followed.

Part II describes and explains the contribution of a model transformation approach that can generate business process models based on textual input. The problem at hand is defined in this part and includes challenges that have to be overcome by the solution. Following the concept and design of the transformation, the approach is described. From the concept, implemented features are presented in a subsequent section. Lastly, in this part, the implemented features are evaluated, and the results are discussed.

Part III concludes this thesis with a summary, a discussion of the drawn implications, and an outlook for future research.

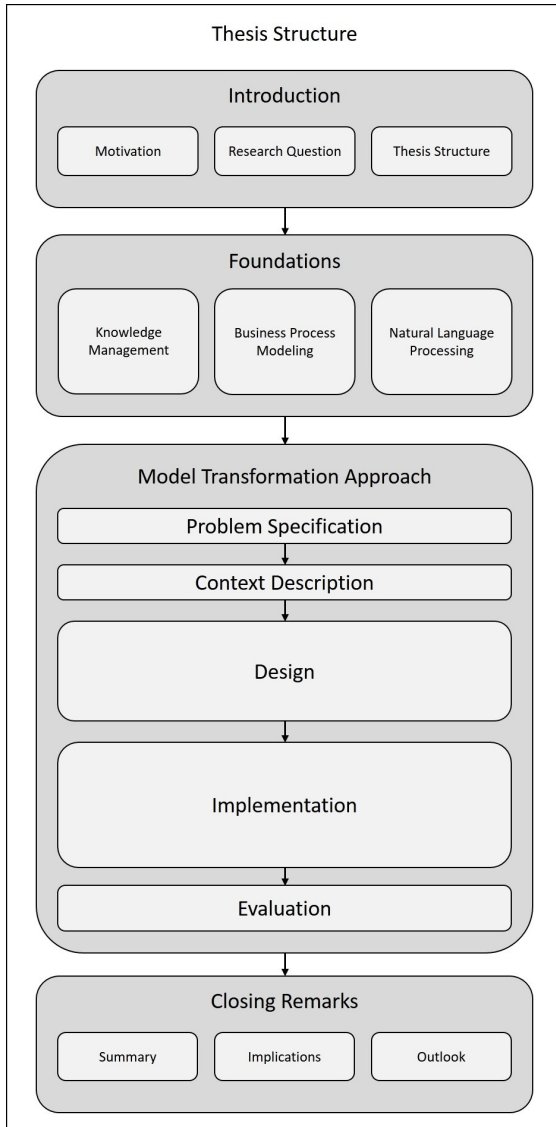


Figure 1.2: Thesis Structure.

Part I

Foundations

Before having a look at Natural Language Processing (NLP) and how it has already been utilized in the context of Business Process Modeling (BPMoD), especially Petri net (PN) modeling following the Horus Method, and the way this synthesis can be used to enable and support Knowledge Management (KM) processes, every topic will be investigated separately to establish a common understanding of the matter at hand.

Knowledge Management

This thesis is motivated from a practical viewpoint and gained insights from the KM of a business partner. Creating, storing, and sharing knowledge among the company and its workforce is a challenging task that is seen as necessary and beneficial for the companies success. The realization of mentioned knowledge management activities and the use of structures that facilitate these can be achieved through different approaches. In this regard, the business problem's origin lies in the field of Knowledge Management (KM) and focuses on ways to leverage visualization techniques, such as Business Process Modeling (BPMoD), to enhance the different steps of managing knowledge.

Business Process Modeling

Even though Business Process Modeling (BPMoD) is an already established method to visualize information and knowledge in different situations, a discussion has grown in the community of Business Process Management (BPM) about its future. While other approaches, such as Process Mining, gained more attention in the recent past, it cannot be denied that Business Process Modeling (BPMoD) still takes on a crucial role in managing information knowl-

edge. Particularly, business process models can be at the center of knowledge management related to the insights gained from practice. Thus, getting an idea of how these models are created and used and gaining an understanding of the quality of these models is essential for the success of the later presented transformation approach between texts and process models. Of importance is additionally the clarification of differences between individual process models and process modeling languages, as the structure and level of expression can vary and has a direct impact on the transformation.

Natural Language Processing

NLP is a rising field of interest that gained interest not only in academia but also is already used and applied in different commercial solutions. The different approaches and viewpoints on natural language, written or spoken, offer a foundation of existing information on the topic, accompanied by a broad spectrum of tools, libraries, and pre-trained language models that are in place to address different steps of analyzing textual inputs. NLP solutions may not deliver perfect results, but show a satisfying accuracy already and have been improved to such an extent that it can be used effectively in various areas. Examples of the application in practice and academia are provided in 3.2.

In the end, all three topics are coming together as the relationships between fields in each topic are discovered and investigated throughout this thesis. Business Process Models are used in the context of Knowledge Management to appropriately and efficiently deal with contents. Furthermore, enhancements in BPMoD are seen as going hand in hand with a potential enhancements in KM. To achieve such improvements and support the process of BPMoD,

methods and techniques offered in the field of NLP are utilized. In the end, these techniques and methods are used to improve the created business process models in quality, but also should lead to automated creation of these to reduce the time spend. The higher quality or less-time-consuming BProMod should facilitate the sharing and transformation of knowledge and improve knowledge management. The general relationships between the three topics are shown in Figure 1.3. Several approaches can be found in the literature that tried to utilize NLP already when working with different kinds of models. These previous works offer valuable insights into the topic and for the contribution of this thesis. In a compilation of different works [Leo+13; LMP14], Leopold et al. split these approaches based on whether NLP techniques were used on models themselves or any text related to modeling.

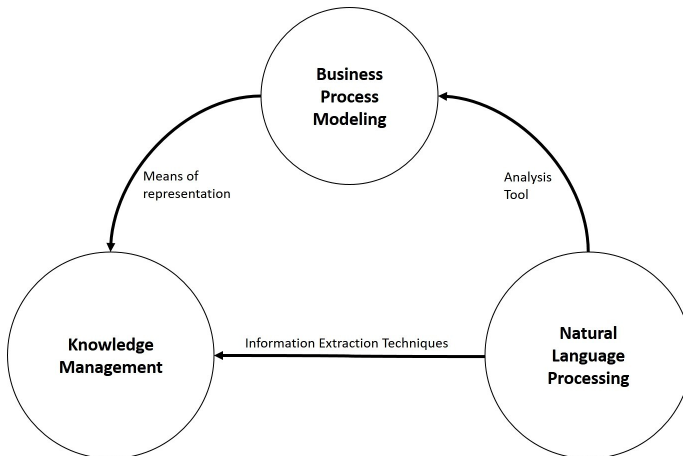


Figure 1.3: Thesis Foundations.

2 Process Modeling and Knowledge Management

In this chapter, one of the three main topics of this thesis's foundations, namely Business Process Modeling (BPMoD), is described. First, the business processes and business process models are explained. Following, BPMoD and commonly used modeling languages are introduced together with a look at modeling processes with Petri Nets (PNs). In the second part, the quality of Business Process Models and ways to capture the different quality dimensions are presented. Lastly, the topic of model transformation and generation is explained.

2.1 Basics of Business Process Modeling

Next to the key components of Business Processes and Business Process Models, central aspects of the process of modeling itself are explained here to facilitate the understanding of how BPMoD is commonly (manually) approached and how automation of this process can be achieved. As part of this section, different modeling languages are listed and described as ways to approach modeling business processes.

BPMod is the graphical representation and modeling of business processes or segments of these [BRVU00]. The focus lies on the representation of business processes with the addition of data and organizational structures [CKO92]. Business process models in this context are used to describe selected business processes to enable analysis and subsequent improvement of processes and thus company performance [Wes10].

BPMod includes individuals familiar with the processes and experts in modeling these, but can also involve non-experts and external stakeholders. Including all involved stakeholders in a project is necessary to obtain a clear impression and description of the processes and to improve process models regarding their representation of reality. Besides, different methods, modeling languages, are used and serve distinct purposes. Common modeling languages are the Business Process Modeling Notation (BPMN) [Whi04], Petri nets [Pet62] or Workflow nets [Aal98].

BPMod aims at clear goals embedded in the goals of the overall Business Process Management (BPM) life cycle. While Business Process Management (BPM) aims to manage and improve the company's business processes, and thus a company's performance, the goal of BPMod as part of the management is the structured and clear representation of current business processes [Hav05]. BPMod provides a clear idea and description of the processes to work with and the changes or improvements to apply and implement. As part of BPM, BPMod is responsible for the design of processes, including the conceptual development of improvements and changes. Nevertheless, BPMod and BPM face several challenges, such as the need for standardization, the inclusion of stakeholders from different fields of expertise, and the training of modelers [Ind+09].

2.1.1 Business Processes

Before introducing BPMoD and its modeling languages, a short clarification of the term business process is given to provide a working definition for this thesis. Weske includes definitions of terms regarding BPM in [Wes10], which are seen as suitable for the use in this thesis, as they match with definitions by other authors.

One definition given in [Wes10, p.5] for the term business process states that "*[a] business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations.*"

Another often used and considered established definition of a (business) process is provided by Becker, Rosemann, and Schütte as: "A process is the chronological and logical sequence of activities that are necessary to process a business-relevant object" [BRS95].

Both definitions emphasize the set of activities interacting sequentially and processing business objects to achieve or contribute to a defined business goal. With a focus on this thesis's core intention, to transform textual input into a process model, the mentioned commonalities between the two definitions represent the general elements that have to be extracted from the input. To generate a process model from text, three core elements of processes have to be identified and extracted, namely the set of activities, the processed objects (inputs/outputs of activities), and the control flow ("interacting in a sequential way").

2.1.2 Business Process Models

In this subsection, the next higher level of BProMod, Business Process Models, is provided to visualize the composition of connected business processes. The structure and underlying rules of creating business process models are relevant for the later-described analysis of textual inputs.

A definition given in [Wes10, p.7] for the term business process model states that "*[a] business process model consists of a set of activity models and execution constraints between them. A business process instance represents a concrete case in a company's operational business, consisting of activity instances. Each business process model acts as a blueprint for a set of business process instances, and each activity model acts as a blueprint for a set of activity instances*". Thus, business process models are representations of the business processes and instances of the included sequence activities. The sequence of and connection between elements follows defined rules provided, e.g., by the chosen modeling language.

Business Process Models are used in a broad spectrum of tasks. Becker, Kugeler, and Rosemann [BKR13] categorize these task into the fields of *organizational design* and *application system design*, as shown in Figure 2.1.

2.1.3 Business Process Modeling Languages

Business Process Modeling, as used for representing a business's processes to enable analysis and improvement of current processes, can be applied in different ways. Different ways include different approaches, methods, or languages. While all applications follow the

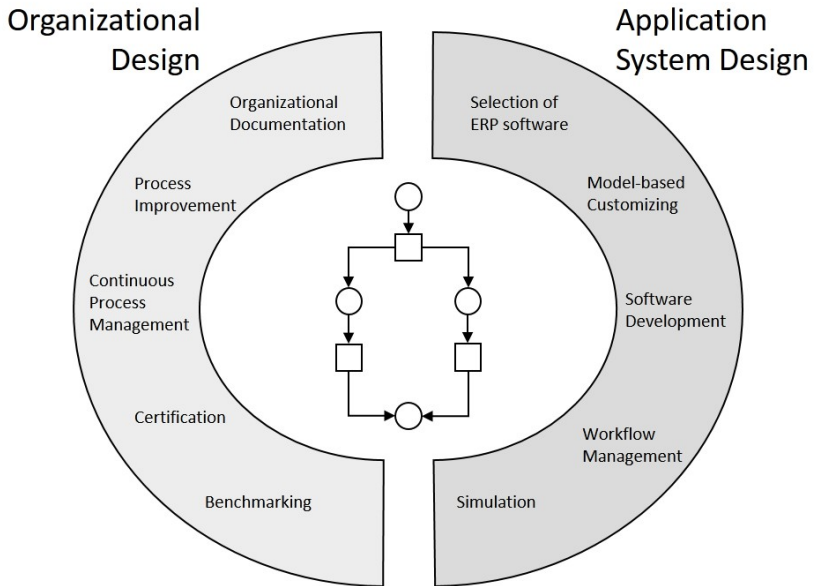


Figure 2.1: Purposes of Business Process Models. Source: based on [BKR13; BPV12].

primary purpose of representing processes, they can differ in their ability to support different tasks, such as analysis or monitoring. Different applications, using different languages, rules, and conventions, and following different goals are also supported through different process modeling software. New applications of BMod also raise new challenges with new possibilities [PV13].

In consequence, different modeling languages serving different purposes were developed over the years. Each modeling language follows rules that differ in aspects, e.g., in the used elements, con-

nections, and control flow. This diversity makes the interpretation for approaches such as the one described in this thesis highly dependent on the chosen modeling language.

An interpretation of a meta-modeling language covering a specific group of similar modeling languages could cover a broader spectrum of modeling languages. However, it would come along with an expected loss of information.

Examples of the most common and used in practice modeling languages are:

- **Petri nets:** A PN is a bipartite graph consisting of transitions, representing events that occur, and places, which represent conditions [Pet62]. A modeling approach with PNs is the Horus method (see [Sch+11]), which is supported by the Horus Business Modeler as a modeling tool.
- **BPMN 2.0:** BPMN 2.0 is the current version of the Business Process Modeling Notation, a graphical representation for business processes introduced in [Whi04] and is included in various modeling tools. It is based on flowcharts and shows similarities to other model types, such as UML diagrams.
- **Workflow nets:** Workflow nets are a subclass of PNs that are used to model the workflow of process activities. Workflow nets can be modeled with tools for modeling PNs, but also workflow focused software, for instance, the Camunda workflow-management system [Aal98].

2.1.4 Imperative and Declarative Modeling

Next to the mentioned languages, numerous other alternative languages and variations exist that serve different purposes, such as, for instance, imperative and declarative modeling. The distinction between declarative and imperative modeling languages is rooted in computer programming.

While imperative modeling languages explicitly define the execution of a process and express what action and how it is performed, declarative languages do not specify a procedure. They instead declare the constraints and requirements that define what has to happen for an action to be performed. The execution is left to be determined by the system itself. Declarative constraints are often used to reduce the space of possible executions of an imperatively described process [Pic+11].

Imperative models are often considered to follow an 'inside-to-outside' approach, which defines the execution, and possible alternatives or extensions have to be explicitly stated and added. In contrast, declarative models follow an 'outside-to-inside' approach, which includes constraints that specify the execution alternatives, and every execution within these constraints is considered valid. The number of alternative executions is regulated over adding and removing constraints [Pes08].

An imperative approach comes close to the idea of using process descriptions according to stated input requirements in the context of this thesis as a means to express the sequence of performed actions of the business process. Declarative languages in the form of, e.g., constraints or execution rules are nevertheless relevant when considering process models as part of a bigger construct,

such as the Horus method, which includes different kinds of models and their relationship to each other in a broader business management context. Additionally, as observed over a first iteration of experiments in [Pic+11], imperative process models promise better comprehensibility among experienced and inexperienced process modelers.

In this regard, in this thesis, the assumption is followed that imperative process models and the corresponding imperative process description are generally easier to assess. It describes the directly and visually represented contents. In contrast, the declarative expression of process model relevant elements adds complexity. Though the description of constraints and rules that add information to a process model, a declarative approach would help include the other models used in the Horus method and express the "background" of a process model.

2.2 Business Modeling with Petri Nets

In this section, the modeling language of Petri nets is explained. An understanding of basic PNs is required in later chapters that introduce a specialization of PNs, Horus procedure models used by the business partner.

Since Carl Adam Petri created his modeling language in 1939, aimed to illustrate chemical processes better [Rei13], it has undergone much research. It is nowadays utilized as a discrete state-based system modeling technique in different domains. The possibility to analyze these graphs mathematically while remaining accessible gave them a unique value and led to PNs being utilized more intensively. As a result, various techniques have been created, allowing users to check these models, extract information, or work with those. For the usage in specific domains, additions to the initially small set of notation elements and definitions were designed and applied like Stochastic PNs [HTT00] or Coloured PNs [BRR06].

The application areas of PNs are diverse. The modeling language is widely used in technical areas like creating and analyzing communication protocols [ZZ94], in functional areas such as the alignment of manufacturing systems, or software development, especially when dealing with particular cases like parallel execution as a means to analyze and verify the code [ZC06]. As a result, PNs are practiced in domains where discrete event systems of every shape prevail. There are efforts to extend the standard PN notation to include new elements utilizable in more specific domains. However, there exists a common ground to describe the systems through only places, transitions, and arcs.

PNs offer the potential of expressing complex structures with a limited amount of model elements. With PNs, one can ensure to represent almost all the process-related knowledge that may come up. In contrast, the generation of PNs from texts can focus on textual structures mapped to a small number of model elements. This enables us to provide an early concept and prototype of mapping that includes all relevant model elements of a PN. An early prototype might be flawed and not recognize all structures perfectly, but the extension of this working prototype that can generate in this first iteration already PNs from texts, even though not completely correct, would cover the set of elements used for PNs. An extension and further development of such a prototype would then have to focus solely on structures and patterns found in PNs and not on additional elements that have to be integrated.

PNs are used as a modeling tool that applies to various systems. A single PN model is a bipartite graph. It consists in its base form of three different elements:

- **Places**, which contain zero or more tokens, and describe the formal state of a system.
- **Transitions**, which enable tokens to move between places.
- **Arcs**, connecting a place with a transition or vice versa.

Based on this simplicity, PNs can be handily depicted both graphically and mathematically. The mathematical representation also allows for the analysis of said models by creating equation systems and checking for correctness properties through algorithms [Mur89].

Furthermore, asynchronous and concurrent activities can be modeled while satisfying constraints on sequential properties [Pet77].

To have a shared understanding of terms and to use consistent notation across implementation and analysis, the definition by Murata [Mur89] will serve as a starting point: This standard PN notation is as such isolated from any specific domain. However, in practice, the modeling of a PN may generally take different techniques. The mentioned original intention of Reisig to use those models to make the structure of chemical reactions accessible [Rei12] has a different semantic meaning for places and transitions when comparing those to PNs used for analyzing software patterns despite sharing the same elements. The procedure to create this model would differ and require a distinct domain language when talking about the graphical illustration.

As one goal of this thesis is to analyze to what extent and how reliably a machine can automatically generate PNs from natural language texts. The focus will be fixed on a single domain: The translation and creation of PNs depicting real-world processes.

PNs are typically used when creating a state-based version of business processes for analysis and deploying them into other systems. Specialized PNs, such as the Horus models used by the business partner, which are introduced in Section 5.1.1, are often called Workflow Nets (WNs) [ELS10]. Workflow Nets (WNs) can be characterized by their transition based structure. In addition to the semantic differences between PNs and WNs, WNs do have supplementary definitions and properties. The most common characteristic is called soundness, which guarantees a deadlock-free model through its sub-characteristics liveness and boundedness [VDA+11]. With regards to the distinct differences that can be identified between PNs and

WNs, Van der Aalst and Hofstede claim that a well-formed business process described through a PN can be considered compliant with the idea of a WN.

The difference between both is thus not of significant relevance for the transformation process in this thesis. However, it should be kept in mind as WNs are often closer to the application level of a business process and important when it comes to the execution. The focus will lie on the semantic notion of the specialized version of PNs in the form of Horus models as a tool to model real-world processes with the notion of WNs as a formal concept with the focus on the practically related workflow of a process.

The main challenge of transforming text sources into a modeling notation such as PN or based on PNs will be the extraction of atomic process steps and connecting them in a meaningful way, not violating either the modeling language's underlying rules or being contrary to the source text.

A promising starting point for the efficient and precise extraction of Petri net elements is provided by regular patterns that can be identified additionally. Such patterns describe additional structures common in PNs and can be used to identify "sub-models" of the respective PN that fulfill a certain purpose, such as choices, parallelism, or sequence [LVD09]. Examples of patterns found in PNs, such as parallel splits or exclusive choices, are shown in Figure 2.2.

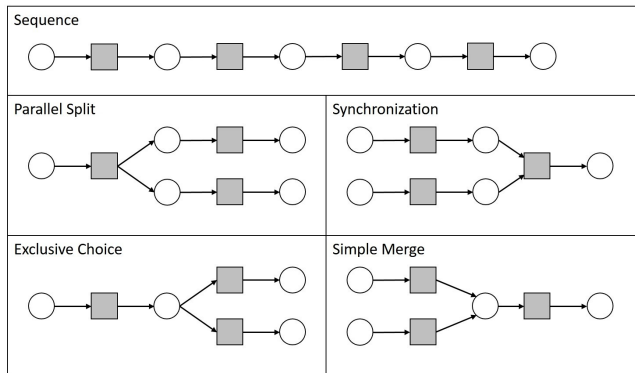


Figure 2.2: Petri Net Patterns. Source: based on [LVD09, p. 4].

2.3 Quality of Business Process Models

The quality of business process models is often perceived differently based on the subjective interpretation of the particular model user [HS06; SG+12]. Therefore, proposing a universal definition covering all aspects and possible utilization scenarios is hardly possible.

As a consequence, Overhage, Birkmeier, and Schlauderer define the quality of business process models in a more generalized way as *"the totality of its characteristics that bear on its ability to satisfy stated requirements"* [OBS12, p.232]. Ensuring a high degree of model quality can imply a higher quality level for any information system that is implemented based on the former [Moo05], as models generally are not created solely for their representational aspects but serve a purpose in different kind of projects.

Two different categories of quality assurance approaches can be distinguished in practice:

1. *Analytical* approaches aiming at quantifying the quality level of a particular model,
2. and *constructive* approaches with the intention to provide guidance for the modeler during the actual modeling process and to ensure that the resulting model satisfies defined quality criteria [Bal08; OBS12].

Analytical quality assurance approaches build upon *quality metrics* and corresponding *measurement procedures* [Moo05; OBS12]. Such metrics are, e.g., the size, diameter, and density of a business process model [Men08]. Additionally, comprehensive *quality frameworks* exist to assess general model quality on a higher level of abstraction. The mentioned frameworks exhibit a hierarchical structure, with the model quality residing at the top and the corresponding quality metrics at the lowest level. Between these two levels, there can be one or more additional levels that are structured around *quality (sub-)characteristics* to aggregate metrics of a similar nature [Moo05].

Frameworks commonly discussed in the BPMod literature include, for instance, the 3QM-Framework described in [OBS12]. Overhage, Birkmeier, and Schlauderer define three characteristics: syntactic quality (rule adherence), semantic quality (validity and completeness), and pragmatic quality (comprehension). Another exemplary framework is the SEQUAL framework [KSJ06], which is compared to the 3QM-Framework more extensive, as it also considers empirical quality (readability), social quality (feasible agreement), and

organizational quality (goal fulfilment). The literature on quality assurance found in the field of BPM focuses on approaches ranging from the design and evaluation of entire modeling languages down to the level of individual models [OBS12]. For the latter, prominent approaches include the *Guidelines of Modeling (GoM)* [BRS95] and the *Seven Process Modeling Guidelines (7PMG)* [MRA10].

Another approach is presented in [LR+11] and focuses on the *secondary notation* of process models. This notation focuses on the representation and visualization for the user. La Rosa et al. [LR+11] describe different patterns that aim to change this type of representation to improve a model's understandability. Patterns, such as shown in Figure 2.2, include the provision of guidance for finding a good spatial arrangement of model elements, using enclosures, color, or other visual elements for emphasis, and adding additional annotations to a model.

In practice, measuring the quality of business process models is difficult, especially as there is no single number, quantified measurement, or value that can delimit good from bad quality or something in between. While the mentioned approaches generally allow creating higher-quality models, they usually do not define any concrete metrics or measurements and do not enable an actual quantification [OBS12]. Additionally, approaches such as the Guidelines of Modeling (GoM) require particular expertise in modeling to be applied and are less suitable for novice modelers [MRA10].

Independently of the approach, most frameworks focus on quality categories but leave out an actual definition of metrics and measurement procedures. Consequently, Metrics are rendered subjective and too abstract to be directly used in practice [MRA10; Moo05; OBS12]. Lastly, one component that many analytical frameworks

are not paying attention to is guidelines that allow the improvement of models based on any shortcomings identified during quality measurement [Moo05].

This section establishes an understanding of business process model quality and means to assess such. A subset of the introduced frameworks and guidelines is further provided with a brief description of each approach in Section 2.3.1.

2.3.1 Quality Frameworks and Guidelines

In the following, different frameworks and guidelines used to ensure and assess the quality of business process models or related artifacts are introduced. The most commonly addressed and mentioned ones in the literature are presented. A more extensive list of existing frameworks can be found in [Moo05].

Quality frameworks are of relevance on the one side to assess the quality of process models, while on the other side, they also define what quality is expected to consider a good quality process model. Additionally, quality frameworks can support the definition of requirements for inputs in the form of textual process descriptions used in this work.

Stampers Semiotic Framework [Sta91]

When talking about information and assessing their nature and their management, semiotics is an often considered field. Semiotics *"is the study of sign process (semiosis), which is any form of activity, conduct, or any process that involves signs, including the production of meaning"* [Cha07].

Table 2.1: Overview: Model Quality Frameworks.

Title	Authors	Date
Guidelines of Modeling	Becker, Jörg and Rosemann, Michael and Von Uthmann, Christoph	2000
SEQUAL Framework	Krogstie, John and Sindre, Guttorm and Jørgensen, Håvard	2006
SIQ	Reijers, Hajo A and Mendling, Jan and Recker, Jan	2010
3QM	Overhage, Sven and Birkmeier, Dominik Q and Schlauderer, Sebastian	2012
CMQF	Nelson, H James and Poels, Geert and Genero, Marcela and Piattini, Mario	2012

One of the most used frameworks, which Business Process Model Quality frameworks build on directly or indirectly, is the semiotic framework or semiotic ladder by Stamper. The semiotic ladder is intended as a framework of information systems research and partly applicable to process modeling as part of information systems development. Stamper used semiotics and his semiotic ladder as a framework for information and knowledge in the context of information systems research.

In [Sta91], the different steps of the framework are described as shown in Figure 2.3. The steps include:

1. Physical World - Represents the physical properties of signs, the physical tokens.
2. Empirics - Explores the physical communication towards, e.g., channel capacity and efficiency.
3. Syntactics - Refers to the structure and used vocabulary.
4. Semantics - Includes the meaning of as well as the relationship between a sign and what it refers to.
5. Pragmatics - Is about the usage, the purposeful use of signs.
6. Social World - Refers to the formal and informal norms and social effects.

In this thesis, the framework by Stamper can serve as a bridge between process model quality and knowledge quality as part of information systems. As process models serve as knowledge carriers in a knowledge-oriented context, an assessment based on semiotics

can be applied. However, not all parts of the semiotic ladder can be mapped to this project. The syntax and semantics steps are especially relevant here, with minor references to pragmatics and the social world.

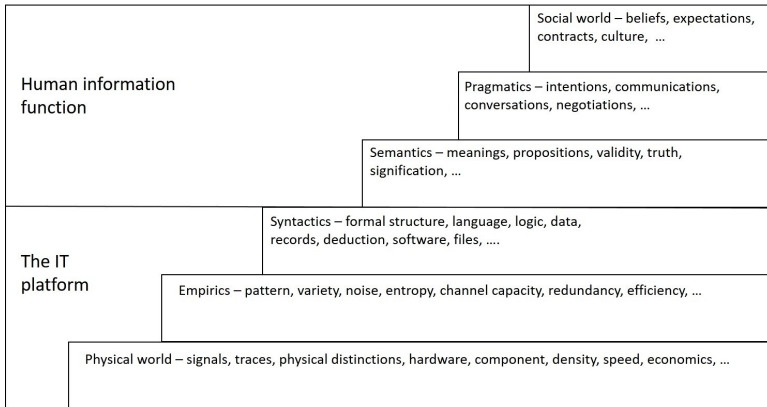


Figure 2.3: Semiological Steps between the Physical and the Social Worlds. Source: based on [Sta91].

Guidelines of Modeling [BRVU00; BPV12]

Modeling business processes can evolve into a difficult task with complex and huge process structures. The guidelines for modeling all the variety of conceptual models were developed over time, such as in [BRVU00; BRS95], and are applicable to process models. A process modeling-focused extension of these guidelines is described by Rosemann in [Ros96].

The GoM framework by Becker, Rosemann, and Von Uthmann [BRVU00] includes six general guidelines in total, which describe properties of well designed models and their languages. The three basic guidelines include the model's requirement to be syntactically and semantically correct and hold relevant information. Also, the economic efficiency guideline aims at providing reference models or tools for modeling. The most relevant optional guideline requires models to possess a high degree of clarity, enabling the user to understand these models more quickly. Further guidelines are related to systems and models of other views, which are not directly relevant to this thesis's target implementation.

The six guidelines from [BRVU00; BRS95] include the following statements that are shown in Figure 2.4:

1. **Correctness:** Refers to the syntactic and semantic correctness of a model. Syntactical correctness is given when a model conforms with the meta-model of the used modeling language. Simultaneously, a model is semantically correct when it is consistent with the real processes it represents.
2. **Economic Efficiency:** Involves all other guidelines and can be seen as a cost-benefit constraint of the model. The creation of the model and consistency with the guidelines have to be feasible and benefit the involved people.
3. **Relevance:** The guideline of relevance encourages the creator of a model to focus on the relevant elements to represent in a model. A model should be condensed to the relevant and needed elements to represent. An object in a model can be

- considered irrelevant if the model loses no information or meaning after deleting the object.
4. **Comparability:** Refers to the consistent use of modeling conventions and rules during a project. Modelers are encouraged to model consistently with the, in the modeling project used, modeling language, connected rules, and conventions to ensure that models created during the project can be assessed and compared.
 5. **Systematic Design:** The systematic design emphasizes the relationship between different models. For example, every data within a process model needs a corresponding data model. A common approach is the ARIS-approach, which includes the systematic layers *functions, organization and process* [SN00]. A meta-model of the used language includes all the views and is necessary to provide a consistent, systematic design.
 6. **Clarity:** The guideline of clarity is somewhat subjective but supports the modeler in creating a readable and understandable model. A model should be accessible for other involved people and thus not include knowledge outside the model or used modeling language. Layout conventions of modeling languages or set by the involved stakeholders help ensure a degree of clarity of created models.

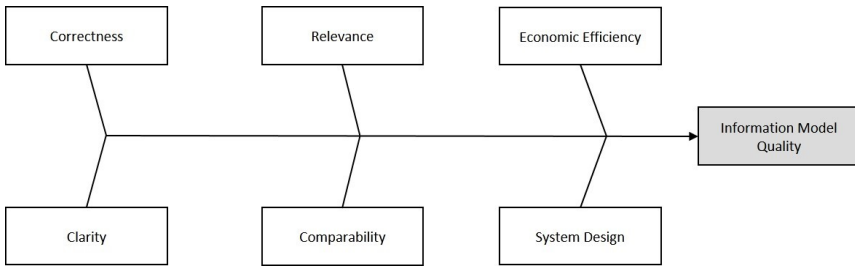


Figure 2.4: Guidelines of Modeling. Source: based on [BRS95; BRVU00].

SEQUAL Framework [KSJ06]

SEQUAL is a semiotic framework used to evaluate conceptual models, which was first introduced by [LSS94] and further refined over the years in, e.g., [KLS95; KSJ06]. The Semiotic Quality Model (SEQUAL) was created to evaluate all types of conceptual models, while modifications emphasize different kinds of models, such as business process models [Kro16].

The framework makes use of the similarity between process modeling and the used natural language to describe the resulting models to link concepts from linguistics and semiotics to evaluate the model. Conceptual models are considered a representation of a set of expressions in a language and in consequence can be evaluated with approaches from linguistics.

The quality aspects described and included in the framework are the actors that work with the models, their knowledge, their model interpretations, the modeling language, the domain expressed in the model, and the modeling goal.

Based on [Kro16], different types of quality are described:

- Physical quality: Actor Access
- Empirical quality: Model Externalization
- Syntactic quality: Language Expression
- Semantic quality: Domain of Modeling
- Pragmatic quality: Technical and Actor Interpretation
- Social quality: Social Actors Interpretation
- Deontic Quality: Goal of Modeling

A refinement of the mentioned types can be seen in Figure 2.5. It has to be acknowledged that no concrete quality metrics are defined, but rather a conceptual framework described.

SEQUAL was analyzed and considered a widely, accessible, and variously applied framework in practice by [Moo+02]. This can be seen in the number of other frameworks that build on the idea of the SEQUAL framework, such as the 3QM-Framework.

3QM-Framework [OBS12])

The 3QM-Framework is another approach to assess the quality of a business process model inspired by the SEQUAL-Framework. It is based on a hierarchical structure with syntactic, semantic, and pragmatic quality at the next refined level below the overall model quality. In the more detailed levels of the quality hierarchy, nine

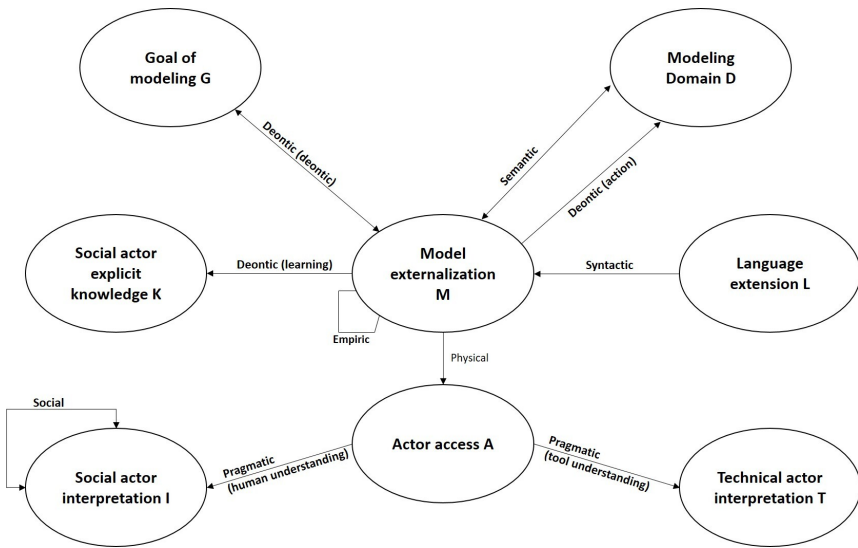


Figure 2.5: The SEQUAL Framework. Source: based on [Kro16].

characteristics are introduced as sub-characteristics of the three mentioned ones on the above level.

The sub-characteristics include, e.g., correctness, relevance, and completeness. The nine characteristics are measured by 35 quality metrics that are defined with the respective measurement. Besides, the authors propose a weighting to the framework’s individual metrics to adjust the calculation of a single quality metric for a process model.

7 PMG [MRA10]

Another set of guidelines compiled by Mendling, Reijers, and Aalst distinguishes four categories that revolve around the correctness of the model, layout, used elements, and the language used to create the model with regards to a fixed dictionary or defined labeling conventions. The 7 Process Modeling Guidelines (7PMG) by Mendling, Reijers, and Aalst are aligned with the EPC notation but are mostly applicable to any process modeling toolset. The guidelines mostly focus on model readability and error reduction. Aside from using as few elements as possible and minimizing the routing paths, the 7PMG advice avoids using the inclusive OR connector and suggests strict labeling.

The Seven Process Modeling Guidelines (7PMG) have a similar purpose than the GoM by Becker, Rosemann, and Von Uthmann, but introduce a less detailed set of rules that are easier to follow also for non-expert modelers. The defined rules are supposed to influence the modeling style to yield a result with high understandability and a reduced number of syntactic errors in the model [MRA10].

Consequently, they are covering a narrower scope but are more applicable through their concrete formulation. Such formulations are, for example, *"use as few elements as possible"*, *"use verb-object activity labels"*, and *"use one start and one end event"* [MRA10, p.130].

Further Guidelines and Frameworks

Several other frameworks that are merely combinations of the mentioned ones, not applied by a big audience or focus on highly spe-

cialized scenarios, were not mentioned here due to the mentioned characteristics.

Nevertheless, these frameworks might provide helpful insights in a different scenario and should not be neglected. Examples include the Conceptual Model Quality Framework (CMQF) by Nelson et al., the MAQ model [Sad15] or the SIQ framework [RMR10].

2.3.2 Quality Metrics

Considering the introduced frameworks and guidelines and extending these by existing works, such as [Pfl18], different quality metrics can be identified. Even though most of the frameworks and guidelines are kept at a rather conceptual level, precise definitions of metrics can either be taken from the remaining approaches that define them or be extracted based on the conceptual description provided. Considering process model quality on different levels, similar to Stamper's semiotic framework and looking at the levels of pragmatism, semantics, and syntax, it becomes clear that it is especially for the syntactic quality aspects metrics are either in place or can be defined rather straightforward. The next possible quantification of quality should be possible over pragmatics. In this case, the model itself is not necessarily evaluated, but rather metadata, such as the number of views and references.

Lastly, the initial starting point of the research conducted for this thesis is the semantic quality of process models. Metrics for these are scarce and hard to define, as often the content at hand and the intended or wanted results are hard to generalize and describe. To come up with objective metrics in this context inherits a significant complexity, which can be addressed by reducing the interpretation

space for the content by adding, e.g., contextual information or personal feedback. Some metrics might indicate some overlap with the mentioned levels of quality. Syntactical metrics might have an indirect impact on related semantic metrics and vice versa. The interrelation between metrics of different categories has to be kept in mind for an approach that considers syntactic and semantic quality for the assessment of the overall model quality.

This thesis aims to address semantic and syntactic quality dimensions of process models via automated model generation and the inclusion of contextual knowledge. The syntactic quality dimension is assumed to be covered by an automated generation of a process model based on a given input, e.g., a data frame that contains the model description. The placement and arrangement of elements are made automatically in such an approach, and syntactic metrics such as edge-crossings, diameter, or overlap are automatically addressed as a side effect of the used generation algorithm.

Nevertheless, just a subset of the syntactic metrics, e.g., as listed in Table 2.2 is relevant for and influenced in the model generation. An example of syntactic metrics for business process models is provided in table 2.2. These metrics are the results of a former thesis, which focused on using syntactic metrics to measure the quality of process models to calculate progress and rewards in a gamification setting.

Under these circumstances, this thesis focuses on the semantic quality dimension and a process model's ability to express the related content wholly and correctly. At the same time, syntactic metrics are paid attention to as a side effect of the proposed approach.

Table 2.2: List of Quality Metrics in the Horus Gamification Concept. Quality Metrics are categorized based on different Characteristics and the Optimization Goals. Source: based on [Pfl18].

Characteristic	Quality Metric	Optimization Goal
Readability	Edge Crossings	▼ Remove all edge crossings
Readability	Edge Bends	▼ Use bend points sparingly
Readability	Node Occlusion	▼ Remove node overlaps
Readability	Angular Resolution	▲ Maximize angles between leaving arcs
Readability	Consistent Flow 1	▲ Arrange elements from top to bottom
Readability	Consistent Flow 2	▲ Arrange elements from left to right
Readability	Orthogonality	▲ Lay model elements out on a grid
Complexity	Size	▼ Keep the size small, split large models up
Complexity	Diameter	▼ Minimize the value of this metric
Complexity	Density	▼ Minimize the value of this metric
Complexity	Connectivity Coefficient	▼ Minimize the value of this metric
Complexity	Avg. Connector Degree	▼ Minimize the value of this metric
Complexity	Max. Connector Degree	▼ Minimize the value of this metric
Complexity	Connector Mismatch	▼ For each split, model a corresponding join
Complexity	Control Flow Complexity	▼ Limit the number of execution paths
Complexity	Cyclicity	▼ Try to avoid cycles
Complexity	Token Split	▼ Try to avoid AND and OR splits
Complexity	Sources and Sinks	▼ Use exactly one start and end element
Completeness	Names	▲ Provide names for all elements
Completeness	Short Names	▲ Provide short names for all elements
Completeness	Descriptions	▲ Provide descriptions for all elements
Completeness	Notes	▲ Provide notes for all elements
Completeness	Business Rules	▲ Provide business rules for all activities
Completeness	Documents	▲ Provide documents for all elements
Completeness	KPIs	▲ Provide KPIs for all activities
Completeness	Object Types	▲ Provide object types for all object stores
Completeness	Refinements	▲ Refine all activities with further details
Completeness	Resources	▲ Provide resources for all activities
Completeness	Risks	▲ Provide risks for all activities
Completeness	Roles	▲ Provide roles for all activities
Completeness	Services	▲ Provide services for all activities
Completeness	System Components	▲ Provide system comp. for all activities

2.4 Knowledge Management via Models

Knowledge Management plays a significant role in consultancies, such as the business partners PROMATIS and Horus software GmbH. Knowledge is needed on the one hand for the realization of projects and on the other hand, as part of products and services. The scope and quality of the expertise bundled in a Knowledge Base are of significant relevance for both use cases, and continuous quality control of the knowledge bases is desired and necessary. The ambition to outperform competitors has driven companies towards finding opportunities to increase their performance. Cost reduction and the gain of overall efficiency are necessities to increase performance.

Business Process Management (BPM), and included BPM_{Mod}, is used as a set of instruments to analyze and improve business processes across companies [Wes10]. The understanding of processes in an enterprise and how to enhance them is essential to this management process. One instrument to visualize and store knowledge, BPM_{Mod}, has become a crucial instrument of managing such for many companies [Ind+09]. Because of this, BPM_{Mod} from a knowledge management perspective and the understanding of how to identify and create models of acceptable quality is of high relevance for the internal organization of companies [KB06]. Many tasks revolving around business processes are represented and documented in mathematics-based formal language or natural language, such as requirements engineering. There is a tendency to use other, more expressive techniques, such as process models.

One of the difficulties of BPM_{Mod} that companies face lies in the variety of involved people [Ind+09]. Stakeholders of a project in

the context of BPMoD are often rooted in different departments, fields, and levels of experience. Efficiently involving all relevant stakeholders presents itself as a challenging task.

In this chapter, an introduction to Knowledge Management is given, and the connection between different Knowledge Management (KM) characteristics and BPMoD as a tool, especially of relevance in a process-oriented KM approach, is explained. The covered topics are focused with regards to the context of this thesis, including the practical use case of the related business partners.

2.4.1 Knowledge Management

Managing knowledge in a company is a complex process. It requires using appropriate methods and the users' motivation to apply and manage knowledge. In the age of modern technology, IT plays a decisive role alongside methods and motivation. Even if all participants' knowledge is documented, the question arises on how this knowledge can be made tangible and available to everyone. The solution to this problem is information and communication technologies. The trend towards information technologies motivates scientists and software manufacturers to develop knowledge management systems to manage knowledge more efficiently [FM04].

In the following subsections, a basic understanding of knowledge management is provided. First, the differentiation of knowledge types is explained together with related definitions. Second, the goals of knowledge management are presented. Third, existing frameworks and models are introduced, and key characteristics are emphasized. Finally, application areas of knowledge management in an academic and practical context are presented.

Basics

In the following the basics of KM are described together with different definitions of KM to explain relevant terms. First, a brief overview of the origin, the history as well as intended goals and practical relevance of KM are given. Second, a set of used terms is defined to clarify their use in this thesis.

The recent interest in organizational knowledge both within and between companies leads to an increasing need to manage knowledge for the company's benefit. Recognizing that knowledge can be a source of sustainable competitive advantage, managers turn to tools and approaches to identify and leverage collective knowledge and experience within organizations [VK98]. Knowledge has become a valuable asset in different forms across a broad spectrum of business areas [DDLB98]. KM is recognized as one of the critical drivers of organizational performance, competitiveness, and profitability. Three key components, people, process, and technology, are considered essential for a successful knowledge leveraging [Omo15]. In this context, technology is an essential instrument of structural knowledge management needed to create new knowledge. Linking information and communication systems in organizations can help integrate previously fragmented information and knowledge flows.

These links can, for example, also eliminate barriers to communication that can occur between different parts or departments of the company [GMS01]. In literature, different core processes of knowledge management can be found that include creating, storing or organizing, transferring, and applying knowledge [DP11; AL01] or defining, identifying, acquiring, distributing, using, preserving and

evaluating knowledge [PRR97]. These processes are often divided into smaller processes. These imply, e.g., the generation of internal knowledge, the acquisition of external knowledge, the storage of knowledge in documents and the storage of routines, and the updating of knowledge and the transfer of knowledge internally and externally [AL01]. The different knowledge management activities are supported by specialized systems, knowledge management systems, that are specially designed for the different activities and use cases [DP11]. Beyond the organizational and, in consequence, economic value, knowledge management takes on a leading role in the context of innovation. The efficient management of knowledge is recognized as essential to establish, sustain, and boost an innovative environment [DP07].

Definitions

A set of terms was established in the field of KM of which each term refers to a specific part. As some of these terms are used in daily language and can be understood in different ways, the most important terms are briefly introduced and defined.

Knowledge

In literature, different classifications of knowledge can be found, which in their abstract nature, avoid the view on implicit-explicit dimensions [AL01]. However, for a detailed view on knowledge, the implicit-explicit knowledge classification of Polanyi [Pol15], as it is a widely cited and acknowledged classification, is considered for this thesis. Polanyi classification of implicit-explicit knowledge categorizes human knowledge into two dimensions - the implicit

and explicit dimensions. Explicit knowledge is often described as codified knowledge and refers to knowledge that can be transferred in formal, systematic language. Implicit knowledge, on the other side, has a personal quality, and it is not easy to formalize and communicate. Compared to explicit knowledge, implicit knowledge is deeply rooted in action, engagement, and commitment in a specific context. In this regard, communication between individuals can be understood as an analogous process aimed at sharing implicit knowledge to build mutual understanding [Non94]. Besides, implicit knowledge includes both cognitive and technical elements. The cognitive element refers to an individual's mental models, such as mental maps, beliefs, paradigms, and viewpoints. The technical components consist of real know-how, craftsmanship, and skills that apply to a specific context. An example of implicit knowledge is knowing the best means of convincing a customer to buy a product [AL01].

In contrast, explicit knowledge is either discrete or digital. Knowledge is recorded in, e.g., historical records in libraries, archives, and databases and is evaluated sequentially [Non94]. An example of explicit knowledge is a manual accompanying purchase of electronic products. The manual contains knowledge about the proper operation of the product [AL01]. Whether implicit or explicit knowledge is more valuable is, however, still controversial. According to Polanyi, these two are not binary states of knowledge but interdependent and reinforcing qualities of knowledge. The implicit knowledge forms the background, which is necessary to develop and interpret the structure of explicit knowledge [IL99].

Furthermore and in the context of processing available *data* towards *information* and then *knowledge*, a distinction of and the

relationship between these three terms has to be established. One visualization of such distinction is often used to explain the relationships from data over knowledge up to wisdom, as shown in Figure 2.6.

As it is crucial to understand the difference between mentioned terms three terms of *data*, *information* and *knowledge* are briefly described in the following. A commonly used distinction can be found in the DIWK-Pyramid, stated in [Row07]. Based on the approach of Rowley, the three terms can be described as follows.

- **Data:** Consists of facts and figures that capture something specific but are not organized and do not provide insights about any further aspects, such as patterns, context, or relationships between the data. The important term for the delimitation of data from information and knowledge is "*unstructured*", as used in the definition of data by in [Thi99].
- **Information:** To gain information based on data, the aspects missing in the data have to be present. When data is put into a context, categorized and condensed, it becomes information [DP98]. Ackoff described the information as essential for answering questions that begin with "who, what, where, when or how many" and thus provides insights about a bigger picture and is, compared to data, enriched with relevance and a certain intention [Ack89].
- **Knowledge:** Contrary to information, knowledge implies an individual perception as it is considered the result of the evaluation of newly gained information based on a person's previously gained experiences [DP98]. Such evaluation and

the ability to establish relationships between new inputs lead to a person's knowledge based on the information that a person has acquired. A definition by Gamble and Blackwell refers to knowledge as *"a fluid mix of framed experience, values, contextual information, expert insight, and grounded intuition that provides an environment and framework for evaluating and incorporating new experiences and information. It originates and is applied in the mind of the knowers"*[GB01]. Knowledge can also be considered to be acquired from a network of information [JHS01].

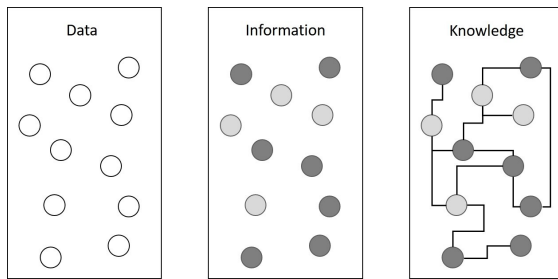


Figure 2.6: From Data to Knowledge. Source: based on [Hen74].

Knowledge Base

A knowledge base is by the definition found in the Cambridge Dictionary "[...] a collection of information about a particular subject"¹. This collection of information is often structured and organized towards a specific purpose. The methods applied together with

¹ See <https://dictionary.cambridge.org/dictionary/english/knowledge-base>. Last accessed: 08.12.2020.

a knowledge base, most often already integrated into the chosen technical realization, enable the transformation between data, information, and knowledge. The set of constructs and methods used in this work are described together with other design choices in Section 5.2.3.

An example of a knowledge base can be found when taking a look at Wikipedia or in the format of wikis that store information and knowledge. However, knowledge bases can take various formats depending on purpose and technology used, such as databases, wikis, or ontologies.

Knowledge Management System

Knowledge Management System (KMS) are a category of information systems to manage knowledge and the related organizational processes following frameworks such as the ones that will be introduced in Section 2.4.1. They are used to support the KM related process in different ways. Alavi and Leidner refers to three main and common applications of Knowledge Management System (KMS) that can be projected to three of the four steps of KM introduced by Nonaka and Takeuchi [NT95]. While the mentioned support does not address all process steps involved in a knowledge managing organization with the same impact, it still facilitates diverse important tasks, specially included in the externalization and combination steps from 2.7. Additionally but not as present, KMS can also be used to, e.g., enhance the establishment of knowledge networks and cultivating knowledge communities as part of the socialization step [Rug98]. In this regard, a traditional knowledge management system enables the storage, distribution, and retrieval of knowledge in various formats. It is often also defined as a knowledge repository.

Simultaneously, the recent developments of knowledge management systems focus not only on the mentioned tasks but also on the social component and the connection between knowledge carriers and knowledge seekers.

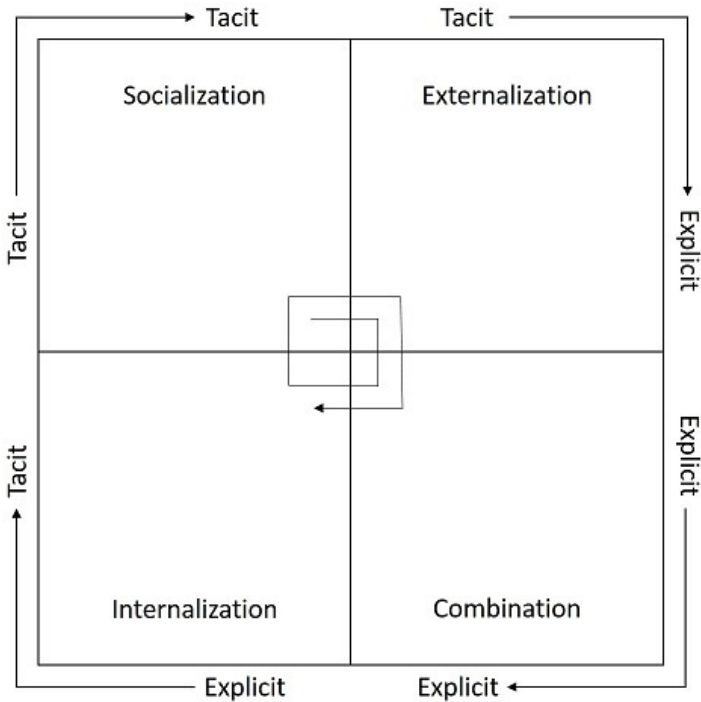


Figure 2.7: The SECI Model. Source: based on [NT95].

Goals

The here described goals are oriented on the later explained knowledge management frameworks. The selected frameworks are considered to capture the common approaches in academia and practice to assess knowledge management and structure related processes. In the following three goals of *Knowledge Generation*, *Knowledge Storage*, and *Knowledge Transfer* regarding the four main knowledge activities (creating, storing, transferring, applying) based on [AL01] and the four phases of the SECI Model by Nonaka [Non94] are described as the general KM goals. It has to be acknowledged that the approach by Nonaka mainly focuses on *Knowledge Generation*. Even though the model by Nonaka can be seen as the most popular and most applied one, other approaches describe how to organize and structure knowledge management, such as those mentioned in Section 2.4.1.

Knowledge Creation

Under the assumption that ideas are formed in individuals' minds, the interaction between individuals takes on an essential role in the development and generation of knowledge [Non94]. Besides communication, organizational knowledge creation involves developing new content or the refinement of existing content within the body of implicit and explicit knowledge of organizations. Nonaka presents four processes of knowledge generation to represent the flow of knowledge between individuals: Socialization, externalization, internalization and combination [Non94]. Socialization refers to transforming implicit knowledge into new implicit knowledge through social interactions and shared experiences among

the organization's members. Externalization and internalization involve interactions and transformations between implicit and explicit knowledge. Externalization is considered to transform implicit knowledge into new explicit knowledge and the internalization of new implicit knowledge from explicit knowledge. Finally, combination implies creating explicit knowledge by merging, categorizing, reclassifying, and synthesizing existing explicit knowledge [AL01].

Knowledge Storage

Empirical studies have shown that companies generate and acquire knowledge but can also forget it [ABE90]. Against this background, knowledge in companies is often recorded in written documents and then stored and retrieved in the company's knowledge repository in the form of structured information, codified human knowledge, (documented) organizational procedures, processes, and the implicit knowledge of individuals and networks. These are usually stored in electronic databases [Tan+98]. Similar to knowledge generation, storing knowledge is differentiated between individual and collective memory. Individual memory is developed based on a person's observations, experiences, and actions [AL01], while collective memory is how knowledge from the past is linked to current business activities. These development factors extend beyond individual memory to other components, such as organizational cultures, changes (production processes and workflows), structures (formal organizational roles), ecology (physical working environment), and information archives (both internal and external) [SZ95]. The storage of knowledge helps to reapply practical solutions in the form of standards and procedures. Such artifacts can help to avoid surplus organizational resources when replicating previous

work. Advanced IT storage technologies and sophisticated retrieval techniques, such as query languages, multimedia databases, and database management systems, can be effective tools for improving the organization's memory. These tools increase the speed and efficiency at which knowledge can be stored and accessed [AL01].

Knowledge Transfer

A fundamental process of knowledge management in companies is transferring knowledge to people and places where it is needed and used. However, this is not a trivial process, as companies often do not know what they know and how to find and access knowledge within the company [Hub91]. Help is provided by IT-supported networks, electronic billboards, and discussion groups. These tools create a forum that enables contact between the person and the knowledge and allows those who have access to the knowledge to enter. For example, this can be achieved by asking a question in the forum and allowing other people to answer the question [AL01]. Another possibility is to provide classifications or organizational knowledge maps in which the respective knowledge can be found.

In comparison, without IT-based support, it is possible to access the knowledge faster. Often this metadata, the knowledge about where to find specific knowledge, is as vital as the actual knowledge itself [Off97].

In addition, groupware software enables companies to generate internal knowledge in structured and unstructured formats and distribute this knowledge in time and space. For example, McKinsey publishes central project documentation on online platforms to promote the memory of individuals and create the possibility of learning company-wide [SZ95].

Knowledge Application

The application of available and processed knowledge is at the core of gaining a competitive advantage through knowledge. Mechanisms that are identified to be essential to integrating knowledge in an organization's value-creating processes are, e.g., the introduction of directives, routines, or a specific organization of collaboration in teams. Such mechanisms aim at different aspects of applying knowledge, such as the transfer of tacit to explicit knowledge to facilitate the communication between experts and non-experts [AL01]. Knowledge application can benefit from the integration into the IT-landscape and underlying processes, such as in a process-oriented knowledge management approach.

Knowledge Management Frameworks

To assess and locate the contribution of this thesis in the field of Knowledge Management, different Knowledge Management frameworks and approaches are investigated. With the help of these, the approach described in this thesis is later related to the different activities of these frameworks to emphasize where and how the results of this thesis can provide benefits. Several reviews of knowledge management frameworks, models, approaches, and industrial cases were already conducted in the past. One review of theoretical knowledge management frameworks was conducted by Lai and Chu in [LC00], while another comprehensive overview of Knowledge Management frameworks can be found in [HJ99].

Another work by Dalkir explores the KM cycle together with (1) major approaches to knowledge management, e.g., the Zack KM Cycle [MZ96], the Bukowitz and Williams KM Cycle [BW00], the

McElroy KM Cycle [McE99], the Wiig KM Cycle [Wii94]), and (2) major knowledge management models, e.g., the von Krogh and Roos Model of Organizational Epistemology [VKR95], the Nonaka and Takeuchi Knowledge Spiral Model [NT95], the Choo Sense-making KM Model [Cho96b], the Wiig Model for Building and Using Knowledge [Wii94] and the Boisot I-Space KM Model [Boi98].

With the mentioned approaches and models, the authors explain the steps of knowledge capturing and codification regarding tacit and implicit knowledge as well as sharing and applying knowledge with the focus on objectives, existing approaches, and a discussion of each step. Additionally, the roles of organizational culture and knowledge management tools are investigated towards their contribution to the knowledge management steps.

Framework of Knowledge Management Pillars [Wii94]:

The framework of the three knowledge management pillars by Wiig focuses on the major activities required to successfully manage knowledge [Wii94]. A visualization of the framework is shown in Figure 2.8. As shown, the three pillars are based on an understanding of knowledge creation, manifestation, use, and transfer. Each of the pillars is a representation of the major activities in knowledge management:

1. Knowledge Exploration: The first pillar represents tasks dealing with exploring and assessing knowledge. As regular tasks, the survey and categorization of knowledge, the analysis of knowledge and knowledge-related activities, the elicitation, codification, and organization of knowledge are mentioned.

2. Knowledge Evaluation: The second pillar involves the review and evaluation of knowledge and knowledge-related activities. Essential for this pillar is the assessment of available knowledge towards its value for the organization.
3. Knowledge Governance: The third pillar focuses on the governance of knowledge management activities and includes three main functions: The synthesis of knowledge-related activities, the handling, use, and control of knowledge, and the leverage, distribution, and automation of knowledge.

Framework of Knowledge Conversion [Non94]:

The Framework of Knowledge Conversion was introduced first in the year 1994 by Nonaka and focuses on the creation of organizational knowledge. Under the assumption that organizational knowledge is created through dialogue and the transformation between tacit and explicit knowledge, the authors propose four patterns of converting knowledge between these two knowledge types as part of a framework that provides an analytical perspective on knowledge creation.

The four steps of knowledge conversion from [Non94] are shown in Figure 2.9 and include the steps of *Socialisation*, the transfer of tacit knowledge, *Externalisation*, the articulation and capturing of tacit knowledge and transformation into explicit knowledge, *Combination*, the combination of existing explicit knowledge, *Internalisation*, the conversion of explicit knowledge into tacit knowledge.

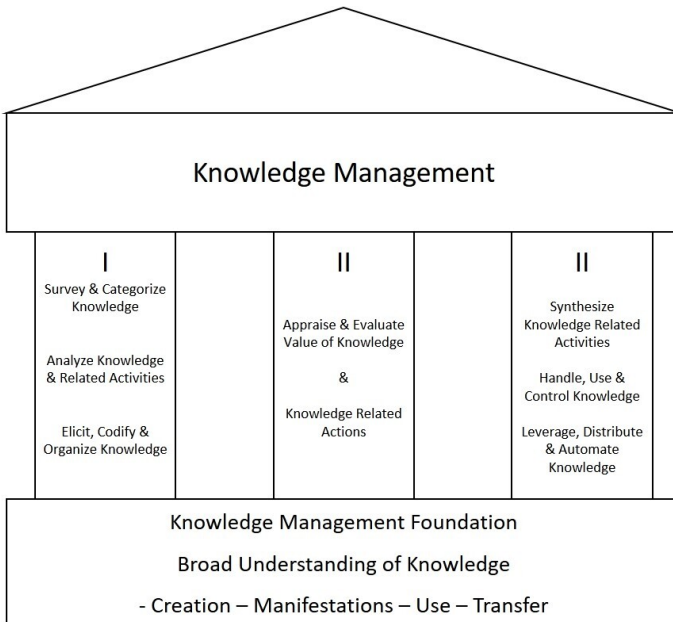


Figure 2.8: Pillars of Knowledge Management. Source: based on [Wii94].

Framework of Core Capabilities and Knowledge Building [Bar95]:

Barton introduces actions for creating, developing, and growing the experience and knowledge of an organization into reusable assets and competitive advantage. The framework is built around the dimensions of the core capabilities shown in Figure 2.10, which organizations should adjust their actions according to. These include the skills and knowledge base, technical systems, organizational

	Tacit	Explicit
Tacit	Socialisation	Externalisation
Explicit	Internalisation	Combination

Figure 2.9: The Four Steps of Knowledge Conversion. Source: based on [Non94].

systems, as well as values and norms of behavior. However, neglecting and not regularly assessing these capabilities can negatively affect the organization, such as a loss of flexibility.

The framework described in [Bar95] revolves around the concept of "*core technological capability*" and includes the following steps:

1. Problem-Solving in the context of the present.
2. Implementing and Integrating in the internal context.
3. Experimenting in the context of the future.
4. Importing Knowledge from an external context.

Model for Organizational Knowledge [And96]:

The work of Andersen focuses on assessing knowledge management and providing a benchmark for knowledge management activities and outcomes.

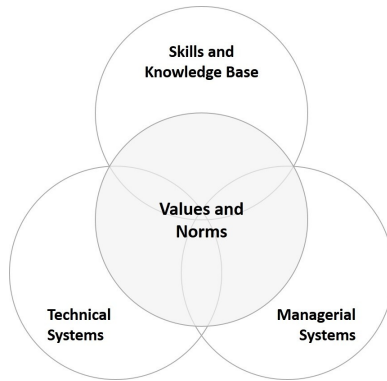


Figure 2.10: The Four Dimensions of Core Capability. Source: based on [LB92].

The developed tool (KMAT - Knowledge Management Assessment Tool) aims to support organizations to make an initial assessment of how well they manage knowledge. Execution of the KMAT is supposed to identify shortcomings in an organization's knowledge management as well as identify aspects that already drive the different processes of managing knowledge. The KMAT proposes three so-called enablers ("leadership", "culture", "technology, and measurement") that can support the development of organizational knowledge through the analyzed knowledge management process. All relevant knowledge management activities and enablers are seen as part of a dynamic system. The process of executing the KMAT leads the organization to identify the information needs and address how this information is collected, transformed, and transferred. An overview of the underlying principle of the KMAT is shown in Figure 2.11.

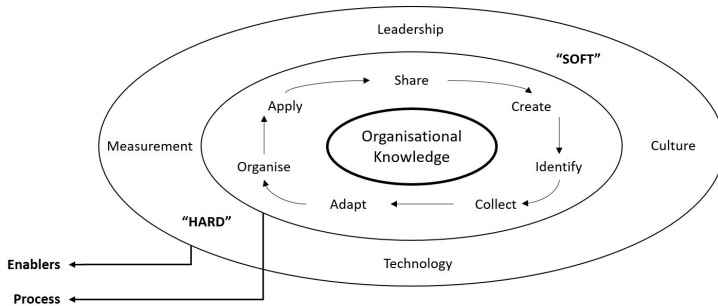


Figure 2.11: Knowledge Management Assessment Tool. Source: based on [Cho96b].

Framework of the Knowing Organization [Cho96a]:

Choo focuses his proposed framework around the organization’s ability to adapt to external changes and promote internal development based on the strategic use of information. His model emphasizes the role of people and groups within the organization in creating and sharing knowledge for decision support, developing new knowledge and capabilities, and making more purposeful and promising decisions. Different strategies of efficiently managing intellectual capital and knowledge are evaluated and related to practical use cases in [CB02] with the help of this framework. Beyond the knowledge management processes that aim to create, share, and preserve knowledge, the framework includes the subsequent actions resulting from better decision-making.

Table 2.3: Overview: Knowledge Management Frameworks.

Title	Authors	Date
Frame of Knowledge Management Pillars	Wiig, Karl Martin	1993
Framework of Knowledge Conversion	Nonaka, Ikujiro	1994
Framework of Core Capabilities and Knowledge Building	Barton, Dorothy L.	1995
Model for Organizational Knowledge	Andersen, Arthur	1996
Framework of the Knowing Organization	Choo, Chun Wei	1996

2.4.2 Business Process Models in Knowledge Management

Knowledge Management incorporates different steps that benefit from the means of visualizing knowledge. The accessibility, the understanding, and technical storage can be improved by choosing visual representation over other representation forms such as texts. One defined approach to combine business processes and knowledge managing activities is process-oriented knowledge management, which strongly depends on the included processes and used artifacts.

Process models and object models, can serve as a backbone of visualizing knowledge and as the foundation of different other model types, such as organizational models or technical models.

Process-Oriented Knowledge Management

In this section, process-oriented knowledge management is emphasized as one application scenario for this thesis's results.

Process-oriented knowledge management intends to bridge the gap between two streams of knowledge management: human-oriented and technology-oriented. This gap is addressed by integrating resource-based and market-based views and establishing the organization's view as a foundation to a process-oriented in [MR03]. An overview of established concepts and best practices for KM from a business process-oriented approach used to connect to the process-oriented knowledge management is provided in [MHV03].

Process-oriented KM shows advantages over traditional KM, such as orientation on the value chain of an organization, the support of KMSs through the provision of context and structure, as well as the design, implementation and integration of KMSs. Process-oriented KM is supposed to maintain the knowledge within and between business processes in relation to the knowledge life cycle [MR03]. A starting point to initiate process-oriented knowledge management is shown in Figure 2.12. Maier and Remus define in this context key roles of an exemplary lightweight approach to process-oriented KMs for different levels:

1. Strategy: Guidance in the design of business and knowledge processes to address the "core rigidity" problem mentioned and explained in [LB92]. The "core rigidity" problem refers to the internal core capabilities as an inhibitor of innovation, which can be addressed through the inclusion of new and external sources of knowledge.
2. Content: Extension of the knowledge base by process knowledge that is mainly residing in process models and knowledge used within processes.

3. Instruments and Systems: Used instruments and systems include, for example, content management systems, process communities, knowledge maps, and best practices, but also process management tools, such as process modeling. Instruments for process-oriented KM are related to process modeling, simulation, monitoring, and controlling.
4. KM organization and processes: Focus on so-called knowledge-intensive processes that are often core processes along the value chain and often depend on knowledge to create, e.g., a product or provide a service.

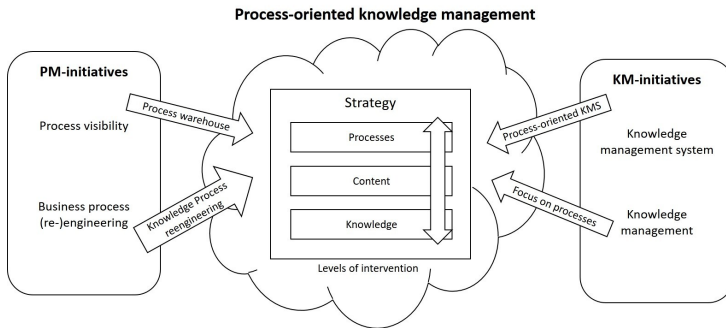


Figure 2.12: Process-oriented Knowledge Management. Source: based on [MR03].

KMSs are used to provide the user with task-relevant knowledge at the related process steps. For the foundation of design and usage of KMSs, process-oriented enterprise modeling or BPMoD can be used, as processes serve in this scenario as context-providers and navigational components [RL00].

One of the tools used in process-oriented knowledge management is BMod . It is often used as a tool for analysis and documentation of business processes. In contrast, the knowledge management activities are used to manage the available workforce's know-how, experience, and skills. Both show drawbacks that can be mitigated through the combination in a process-oriented approach. This way, associations between business processes and relevant knowledge can be captured efficiently. Crucial elements in the process-orientation are the so-called knowledge carriers, artifacts containing information, such as documents, graphics, and models. The connection of knowledge carriers, the carried information, the domain, and the related process is supposed to lead to better integration of both mentioned tools and are at the core of process-oriented knowledge management. [JHS01].

In specific application scenarios, such as, e.g., for small and medium-sized enterprises (SMEs), process-oriented knowledge management offers potentials, and the integration of knowledge management core activities in the business process landscape over the related knowledge domain is seen as an enabler of production and value creation [KOS15; KOS14]. However, it is to acknowledge that KM is realized under different conditions depending on the companies size, business domain, and organization; guidelines, such as the in the paper presented ProWis, are a recent topic investigated to support companies in adjusting their business landscape towards a knowledge-driven and process-integrated organization [KOS15].

One realization of process-oriented knowledge management is introduced by Woitsch and Karagiannis. They describe a service-based approach to knowledge management. In this approach, the knowledge is embedded into the business processes, and these

processes are modeled, while the KMS is used as a structuring tool on a "meta-level". Based on the dependency of KM on the used tools, knowledge-related services are introduced as a provider of KMSs or KM tools on a conceptual level, independent from underlying technology [WK05].

A study by Greiner, Böhmman, and Krčmar about the influence of organizational environment on the selection of knowledge management strategies investigates the relationship between business and knowledge management strategy and argues that the selection of knowledge management strategies based on the organization's goals, such as for organizations that focus on process efficiency, should focus on codification strategies. In contrast, innovation-focused businesses should focus on personalizing knowledge [GBK07].

For a successful implementation of process-oriented KM, used instruments have to be assigned to the KM-related activities and the corresponding processes. Knowledge is made available in a precise way and provides meaningful support for the user. An essential aspect of this integration is the transparency of the included processes. Together, these structures in knowledge integration can provide additional value in context to interpret and apply process-relevant knowledge.

Remus and Schub describes a blueprint for such integration, which refers to actions that have to be performed on the procedure model level and on the conceptual model level, such as preparing the business processes and integrating them in the process-oriented knowledge management or on a conceptual level being aware of the process landscape, representing this in a process model or structured diagram, and using action charts for the integration. Follow-

ing the blueprints procedure, process-oriented KM can build on existing structures, such as quality management or BPM, to handle the complexity of knowledge integration [RS03].

Application in the Project Context

The context considered in this project and the Horus method of the business partner are closely related to process-oriented knowledge management approaches. As defined in Section 2.4.1 that introduced the basics of Knowledge Management (KM), knowledge is often codified to be accessible and efficiently shared and used. Furthermore, established procedures for managing knowledge exist. The representation as a means of communication and transfer of knowledge is considered significant and a feasible solution for this exchange. However, in this context, a distinction between data, knowledge, and information must be kept in mind in the design and implementation. The representation of information and knowledge in different forms (codification) is relevant in different knowledge management-related activities. These activities are included and described, e.g., in the SECI model [Non94].

As one approach to bridge the gap between knowledge management strategies that so far are often either human- or technology-oriented, process-oriented knowledge management was introduced. Process-oriented knowledge management activities are supposed to be integrated into an organization's process landscape to foster the access, availability, and generation of knowledge directly in the relevant process steps. Process modeling, as one tool, is therefore essential for the respective processes, but also as means of representing the related knowledge, especially considering exten-

sions like the Horus methods, which brings together different kinds of models. Based on the process-oriented approach to knowledge management, the different aspects of knowledge management introduced are relevant for this thesis, even though the relationships are of different kinds. An overview of the main aspects of the introduced frameworks and their project relevance is given in Table 2.4.

With this in mind, the presented approach addresses process models as a tool in process-oriented knowledge management and means to automate their generation from another type of knowledge artifact (text). This addresses different aspects of practical use cases, which were provided by the business partner, with the HBM as a modeling tool that incorporates the Horus Method as means to provide a broad overview of the business over processes, data, and organization.

Next to the automation, a standard can potentially be established through the generation rules. In consequence, the quality of process models can be controlled and possibly even increased. Another potential side effect presents the training in process modeling. The direct and automated translation between text and process model provides the user with indirect feedback about his description or a manually created process model.

For the latter, aspects such as automation, social environments, and codification of knowledge take on an important role. Processes, their corresponding process models and the automated generation of these from texts, and thus the externalization and combination as in [Non94], are present in this context in the form of the process-oriented knowledge management and leave room for improvement strategies.

This approach addresses the bridge between process- and knowledge-orientation, as also intended by process-oriented knowledge management, on the one side by focusing on a tool that supports process-orientated knowledge management by integrating a model landscape consisting of process models, different data and organizational models, on the other side by providing a way to support especially the knowledge creation and transformation using process models and emphasizing the process-knowledge relationship.

2.5 Model Generation and Transformation

In this section, the fundamentals of model generation and transformation are introduced. In the later steps, the here described approach of model generation and transformation is of relevance.

In Process Modeling, the search for an automated way of creating or altering models is present in the recent literature [MVG06]. Intended to save time and effort as well as to reduce possible errors during the modeling process, the automatic transformation of models is used in different areas, e.g., for comparing two models and assessing their equivalence [NH15]. The complexity of the used modeling language and included syntax as well as in the used natural language for labeling and describing process steps are a challenge in transforming models. In this thesis, the transformation between modeling languages is not of relevance. However, the generation of models based on a textual artifact together with the transformation of models of the same language, which promises a

Table 2.4: Overview: Knowledge Management in the Project Context.

Aspect(s)	Reference	Project Relevance
Knowledge Externalization and Combination	Nonaka and Takeuchi	Visualization and Structuring of Knowledge
Assessment of Knowledge Management	Andersen	Quality and Expectations towards Knowledge Management
Knowledge-intensive environment	Hislop, Bosua, and Helms; Kohl, Orth, and Steinhöfel	Requirements and Purpose of Knowledge Management and Artefacts
Role of Knowledge Management Systems	Woitsch and Karagiannis	Expectations towards a Knowledge Management System
Tools of Knowledge Management	Jablonski, Horn, and Schlundt	Process Modeling as a Tool in KM
Knowledge Carriers	Jablonski, Horn, and Schlundt	Knowledge-related artefacts in KM (Process Models)
Business Processes in the Knowledge Life Cycle	Maier and Remus	Integration of Knowledge Management Activities into the business processes

less complex scenario, are emphasized. Nevertheless, the translation and transformation might be relevant for future research and projects, as they can aim at text-to-model transformation based on different modeling languages. For instance, in the business partner's case, the knowledge base can include models of different kinds.

Model Generation can be based on different kinds of input of which this thesis focuses on the mentioned textual input. Model Generation itself is not a new topic and has been investigated for some time. Especially in fields such as Process Mining or Data Mining, models serve as a frequent form of representation of information. In these cases, automated generation of models such as, e.g., UML diagrams [JD12] are already paid attention to in different projects.

However, Model Generation and Transformation are not only relevant terms in the context of this thesis but also in other concepts, such as transformation between models, generation from other inputs as texts, e.g., mentioned UML-diagrams that are generated from code, or transformation in the form of attributes, e.g., the placing of elements.

The existing approaches provide helpful insights into known challenges in model transformation and generation, further investigated in Section 4.2. Next to the natural language-specific and modeling language-specific challenges, these challenges are to address in this thesis, and shortcomings of unaddressed challenges are kept in mind when evaluating the approach.

Model transformation is, as can be seen in the number of intents identified by Amrani et al., used in several different practical scenarios, aiming at different purposes and dealing with different challenges and tasks. Amrani et al. introduce a catalog of intents

that describe the use of model transformation in Model-Driven Engineering and the related properties of such intents. Intents include, e.g., refinement, abstraction, (meta-)model generation, and reverse engineering. In this thesis's context, the intents of model generation can be found in refinement and model generation and, to some degree, in reverse engineering. In [Lúc+16] the work of [Amr+12a; Amr+12b] gets extended by additional entries to the catalog of the intention of the model transformation and the consideration to use these insights in the validation of the transformed model.

One challenge in the transformation and generation of models is the validation of the new or altered model to ensure that it is consistent with the target modeling language rules and can be considered a correct model. Amrani et al. proposes a model validation approach in the context of model transformation based on a three-dimensional assessment [Amr+12b]. The dimensions of "*the transformation involved, the properties of interest addressed, and the formal verification techniques used to establish the properties*" are used to make statements about the validity of a model.

Other approaches already address the automation of model transformation and generation. As the model generation is at the core of this thesis, the related works in this field are further described in Section 4.1. It is to mention that the later introduced approaches mainly focus on inputs of imperative nature. In contrast, just a small number of research projects focus on the automated generation of models based on declarative specifications, e.g., [MMB15].

3 Natural Language Processing

In this chapter, the third main topic of the foundations, namely Natural Language Processing, is described. A basic understanding of Natural Language Processing is given together with a description of used methods and techniques originating from Natural Language Processing used in the implementation part of this thesis. The term NLP can nowadays be found in a wide variety of application areas. As the name implies, the research field combines different techniques in attempts to automatically process, understand, and generate natural languages trying to match the human spoken and written language. The field of NLP itself considers a variety of sub-fields, shown in Figure 3.1, which are summarized under the three topics *Parsing*, *Semantics Interpretation* and *Pragmatics*.

3.1 Fundamentals

This section provides the fundamentals of Natural Language Processing. The section begins with a brief overview of the historical development and continues with the theoretical foundations found in Natural Language Processing. Lastly, an impression of tasks

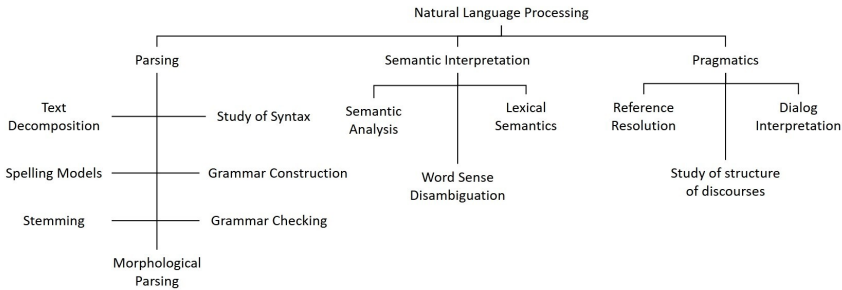


Figure 3.1: Taxonomy of Natural Language Processing. Source: based on [Len02].

and challenges that practitioners and researchers face nowadays is given.

The automatic generation of models is often applied together with techniques of Natural Language Processing (NLP). NLP is incorporating a variety of methods used in different application areas. NLP's underlying idea builds on the syntactic and semantic analysis of text or speech and the use of gained information to accomplish an almost human-like language processing [Lid01]. NLP as a whole combines linguistics with computer science and includes different research areas, and is already widely applied in practice. Application areas include speech recognition, language generation, and the parsing and interpretation of written text for purposes such as personal assistants in smartphones, text translation, and chatbots. Many tools already support natural language processing in different forms, from packages for single tasks faced in NLP over whole NLP-toolkits for different programming languages up

to openly accessible analysis tools, such as the toolset provided by the Stanford NLP Group¹.

While the common understanding and definition of what is considered part of NLP is still openly discussed [Lid01; ID10; Col+11; BI18], innovation and improvements since the first projects in the early 1930s are present in the different research fields.

Often, research related to NLP does not use the term itself but instead uses their distinct notations and definitions in their respective fields. However, since these research advancements mainly deal with computerized understanding or generation of natural languages, they can be included under the umbrella term of NLP.

The analysis, interpretation, and emulation of human languages is a complex problem, which has gained increased attention over the last decades, driven by increased computational power and the ability to employ and use the required algorithms in a feasible manner. Methods and techniques applied in this area generally need as much processing power and resources as there are available. It is only a natural development that the interest in this field increased when both industrial and private computers became affordable and more powerful. The exponential increase in memory and computing power allows for the calculation of more complex NLP models and, consequently, better results when it comes to the precision of interpretation and emulation.

¹ See <https://nlp.stanford.edu/>. Last accessed: 08.12.2020.

3.1.1 Historical Development

To understand the field of NLP, a brief overview of the historical development is given. The past provides an idea about the origins of NLP and where challenges lie, and which concepts it relates to. The research fields NLP connects to are vast, and each of its own often has a far-reaching history that precedes the use of personal computers. Thus, NLP roots are manifold, which can be observed in the methods and techniques used that stem nowadays from or are inspired by these different fields.

Already at the beginning of the last century, people have started to use machines to solve complex tasks that involved human spoken language. With early experiments in the 1930s, the first significant results were achieved in the 1950s, when the automatic translation between two languages was enabled using a computer. A remarkable milestone in this research field is the collaboration between the Georgetown University and IBM that resulted in a running prototype in 1954 as part of the so-called Georgetown Experiment [Hut04]. The prototype was already able to translate Russian sentences into English in almost real-time, even though it was far from what one is used to in the field of translators today, as most of the programmed functions were specifically tailored around the used sentences and words. The processing relied heavily on dictionaries explicitly crafted for the task at hand. Thus, even though marking a big step in the automated processing of human language, the experiment had its shortcomings.

The prototype translator performed best in the translation of sentences, which included words where rules were implemented for and present in the dictionary based on its origin in the domain

of chemistry. The focus on the chemistry domain, the use of basic sentences, and precisely crafted dictionaries had to be considered in evaluating the experiment.

Next to this, another problem was the word-by-word translation approach. As the translator only included the direct surroundings in the translation of a word and was not considering the complete sentence and its structure, it had no means to distinguish, e.g., similar words that differ in meaning [NOMC11].

Over the years, new tools and methods were introduced into the field and improved the implemented approaches. Aiming at mitigating the most significant problem taken from the Georgetown Experiment, the use of grammars was introduced to incorporate the sentence structure in the analysis and enable the mentioned differentiation of similar words based on their position in the sentence structure. Improvements, based on Chomsky's work and the Backus-Naur Form (BNF) from 1963, were achieved in the analysis of ambivalent words and the structure of whole sentences. Lexical-analyzers and parsers became capable of transforming natural language inputs into a machine-readable equivalent [NOMC11].

While overcoming a significant obstacle in the automated translation between languages, another problem was identified along with the progressing methods and techniques. Grammars used for the analysis were manually created, and it was a time-consuming task to create a representation of a natural language that fit the purpose of the translator. It was, furthermore, never able to cover all linguistic constructs [Win72].

While the description of common sentence structures, phrases, and sentences that were not part of these individual solutions often lead to significant misinterpretation considering the resulting

output, the increase of computational power driven by the technological developments provided rather incidentally mitigation of this weakness.

Nowadays, techniques are capable of calculating models that rely on a set of defined and fixed rules of the language and enable a dynamic analysis of languages that considers the mannerism of the individual language. These techniques often stem from the field of statistics. While the variety of approaches is manifold, they always use comprehensive models that incorporate information to assess new input data.

3.1.2 Theoretical Foundations

Natural Language Processing is an umbrella term that is used nowadays for a broad spectrum of diverse applications. Aiming at “accomplishing human-like language processing” [p.3, [Lid01]], it is often placed in and seen as a combination of linguistics, computer science, and cognitive psychology. It is used in areas that focus on languages as the subject of interest and their manipulation through computers. Application areas include topics such as text summarizing [Cho03], speech recognition [Web03], and natural language generation [RD00].

In the approach described in Chapter 5, different tools and techniques stemming from the NLP domain will be utilized to analyze written language and extract the required information to transform an input text into a process model.

NLP is driven by developments achieved in different areas and uses methods from the fields of statistics, neural networks, artificial intelligence, or rule-based systems. While the early years build

on static rule-based approaches that required manual and thus time-consuming configuration, later approaches of NLP showed a transition to the use of methods of statistical analysis and the exploration of rules based on insight gained from this analysis. Recent developments base their approach on the rapid progress achieved over the last years, especially in machine learning. Nowadays, approaches aim to learn rules to analyze the textual inputs based on the supervised analysis of available data. One of the most used techniques is the so-called Word2Vec approach [GL14].

Word2Vec [GL14] is an approach to identify and learn word associations from a large corpus of text using a neural network model. A trained model can detect different linguistic features, such as synonyms, or can provide additional information, such as suggestions of words for a partial sentence. Words are represented as a numerical vector and enable their analysis with methods from linear algebra. Different traits or features are identified and calculated for the words, and each of these traits represents a dimension in the vector space. The vectors of words can then be analyzed and compared in the vector space. This analysis can then support identifying similar words, the prediction of words, or topic extraction.

Word2Vec uses a shallow, two-layer neural network with three layers of which one is hidden. The neural network is trained to reconstruct linguistic contexts of words. An instance of a neural network used in Word2Vec is shown in Figure 3.2. In the example, the inputs of "King", "Man", and "Woman" are analyzed towards four traits, and the same color in a column indicates a high similarity in this trait.

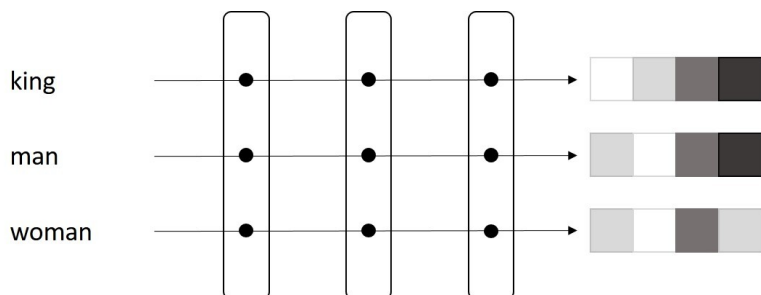


Figure 3.2: A sample three-layer Neural Network. Source: based on <http://jalammar.github.io/illustrated-word2vec/>.

Typical practical scenarios that one encounters in the field of NLP can be found in a condensed way with regards to implementing these in [BKL09] and include:

- **Access of Text Corpora and Lexical Resources:** A text corpus represents a large body of text. Corpora are often designed towards a specific domain. Each corpus can be described through different statistics before further processing, such as the average word length, the average sentence length, or the number of times each word or sign appears (on average). Such a description of text corpora provides preliminary information that can support the decision about further processing steps [BKL09]. This information is the foundation for further processing of available texts.
- **Processing Raw Text:** Most of the accessible sources, nowadays available over the Web in high quantities, offer texts and can be used to extract corpora. Sources that are often used

for practice and research are, e.g., Twitter, different types of blogs, or simple websites that are crawled. NLP techniques used here are, for example, tokenization and stemming. The processed raw text is transformed into an accessible and uniform format as preparation for the subsequent analysis and information extraction steps.

- **Categorizing and Tagging:** The previously processed text parts are categorized and annotated with tags according to different models to further process and enrich the textual input with additional information. Classifying and tagging texts and included words are often known as POS tagging. POS tagging provides additional information about word classes and lexical categories that can be used in the next steps to identify and retrieve useful information. NLP techniques that are used here are, for example sequence labeling (part-of-speech tagging) or n-gram models [BKL09].
- **Extracting Information from Text:** The complexity of natural language makes the analysis of textual inputs and the extraction of information from these a difficult task. Next to the preparation of texts for further processing, the used information extraction methods are of significance. As the current state of NLP does not include a general-purpose solution, questions and information needs must be selected and clearly defined beforehand. The text analysis is thus focused on a specific set of information to extract, such as entities, linguistic patterns, or connections between words [BKL09].

- **Analyzing Sentence Structure:** To deal with the ambiguity in natural language and the almost unlimited number of possible grammatical constructs of sentences, the analysis of sentences with the help of the used grammatical rules is used to improve interpretation [BKL09].

3.1.3 Tasks and Challenges

Research in NLP is conducted towards different tasks that either directly impact real-world applications, serve as a contribution to solving parts of a larger problem, or are of exploratory and innovative nature. A general overview of the most commonly addressed tasks in the current state of research and real-world applications is provided in the following. While some tasks are considered as *Main Tasks* and focus on an independent problem and a larger field, a second category is seen as *Techniques* that focus on approaches that are supposed to tackle specific (sub-)tasks. The upcoming tasks are chosen and described with the English language in mind. However, it is not to neglect that most such tasks and challenges are highly dependent on the language to process. The difference in the word-corpus, grammar, and pronunciation requires a different approach to NLP for each individual language or at least language family. Additionally to the relevant tasks, a comprehensive overview of challenges faced in NLP can be found in [BW06], which is addressed in Section 4.2 as part of the problem specification of this thesis.

Main Tasks

There are different categorizations when it comes to the field of NLP. In this thesis, a general categorization of the tasks is chosen that includes three main challenges:

- **Speech Recognition:** Speech Recognition aims to recognize spoken language and the translation into mostly written text with the help of computers [Jur00]. However, the translation of speech into also other forms of representations that can be interpreted by a machine gained attention over the last years. It refers to an interdisciplinary field of computer science and linguistics.
- **Natural Language Understanding:** As part of NLP natural-language understanding or interpretation is concerned with the machine reading comprehension of natural language text or speech. The understanding of natural language is still considered an AI-hard problem. Moreover, it takes on a significant role in the field of human-computer interaction [All95].
- **Natural Language Generation:** The computer-supported process of transforming (structured) data into natural language is called *Natural Language Generation*. Natural-language generation software is used to automatically produce different types of natural language documents, e.g., reports, summaries, or content for websites. The interpretation and information flow can be considered to be directed from *content to form*, contrary to the understanding of natural language, where the processing aims at *form to content* [McD10].

Next to the three mentioned categories, the tasks can be further distinguished based on a categorization by Feldman [Fel99]. In the following, related fields relevant for this thesis are briefly explained, and exemplary techniques from NLP described. The categories are:

- **Morphology:** The analysis of morphological features that represent the nature of words. A morphological analysis focuses on the internal structure of words and explores possible interpretations. The analysis consists of identifying and splitting words and their parts, such as word roots, prefixes, and suffixes, which often leads to the comparison and assumed similarity to syntax. The relevance or presence of morphological analysis is dependent on the examined language [HS13].
- **Lexical:** After pre-processing a textual input, the text is analyzed by a lexical analyzer. The lexical analyzer splits documents and sentences into tokens based on their syntax. Unnecessary tokens are often removed in this process, such as white spaces. Tokens carry information about the associated word or sentence part, which aims at further processing. The additional information enables the transformation of a raw text into a machine-readable input that can be, e.g., used by a parser or compiler in a subsequent step [Far95].
- **Syntax:** The set of rules or principles that describe the structure of documents, sentences and phrases is considered the syntax. It refers to the grammatical sense of a sentence and the corresponding arrangement of the included words. In NLP, the syntactic analysis can provide insights about the

alignment of sentences and words with the underlying grammar. Based on the grammatical rules, computer-supported algorithms can derive meaning from texts.

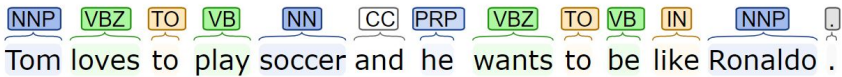
Typical techniques applied in the field of syntax are lemmatization, morphological segmentation, word segmentation, Part-of-speech tagging, parsing, sentence breaking, and stemming. A subset of here relevant techniques is further explained later on.

- **Semantic:** Semantics can be distinguished into relational semantics [RHF02] and lexical semantics [Cru86]. It refers to the meaning that is carried by a text. The analysis of the semantic of a textual input is one of the most challenging aspects in NLP research at the moment. Different computer-supported algorithms are used to improve the clarification of the conveyed meaning of a sentence and the interpretation of word and sentence structures [CW14]. Typical tasks in the field of semantics are Named Entity Recognition (NER), sentiment analysis, terminology extraction, semantic parsing, and semantic role labeling.
- **Discourse:** The analysis of the structure of the different types of textual inputs is used to gain additional insights into the meaning of a natural language text. This process is called *Discourse* in the NLP context and involves activities such as identification of topics, co-references, text summarizing, sentiment analysis and machine translation [Jot+19; Fel99].

Techniques

Different solutions were proposed over the years for the mentioned NLP tasks. The most relevant techniques in the context of this thesis are briefly introduced in the following. The description and explanations are based on the existing implementations later used in the transformation approach, mainly originating from [BKL09; SFCP17], <https://spacy.io/> and <https://nlp.stanford.edu/>.

- **Tokenization:** Tokens are smaller units a text can be separated into. A token can be either a word, a character or a distinct part of words, e.g., in word compositions. The process of identifying and splitting a sentence into tokens is called tokenization. Types of tokenization focus on the three mentioned elements words, characters, and word-parts [BKL09].
- **Morphological analysis or Part-of-speech tagging:** With the annotation of part-of-speech tags, words are classified based on the role they serve in a sentence. Annotated part-of-speech tags are often also referred to as word classes or lexical categories. The set of used tags is dependent on the intended analysis task and is called a tagset [BKL09]. An example of a with part-of-speech tags annotated sentence is shown in Figure 3.3. The annotated labels show the classification of words into categories, such as proper nouns (NNP), third person present tense verbs (VBZ), singular nouns (NN), and pronouns (PRP).
- **Named Entity Recognition (NER):** NER is a technique to identify and segment entities for additional information extraction. The extracted entities are further categorized into

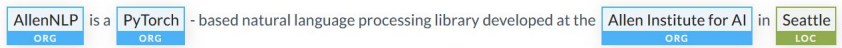


Tom loves to play soccer and he wants to be like Ronaldo .

Figure 3.3: POS-Tagging with StanfordCoreNLP. Source: based on <https://corenlp.run/>.

different defined classes, e.g., persons, systems, or companies. Entities found in sequences of words are labeled with the respective class².

An example of NER is shown in Figure 3.4, wherein the used sentence, different entities are identified as either organization or location.



AllenNLP is a PyTorch - based natural language processing library developed at the Allen Institute for AI in Seattle .

Figure 3.4: Named Entity Recognition with AllenNLP. Source: based on <https://demo.allennlp.org/named-entity-recognition/>.

- Word Sense Disambiguation:** An open and still investigated problem is about identifying the sense of a word in a sentence [SFCP17]. Word Sense Disambiguation (WSD) is viewed as a classification problem. Based on defined classes of distinct senses, a word in a sentence and its possible senses are analyzed based on the available dictionary, the defined

² See <https://nlp.stanford.edu/software/CRF-NER.html>. Last accessed: 08.12.2020.

lexical databases, such as WordNet [Mil95], its grammar, and the word assigned to one of these sense classes³.

The resulting classification contributes to the performance of other different NLP techniques, such as coreference resolution, e.g., as shown in Figure 3.8.

- **Dependency and Constituency Parsing:** Parsing aims at defining the grammatical structure of a sentence. Two types of parsing used frequently in the context of NLP are dependency parsing and constituency parsing. In dependency parsing, the grammatical structure is defined over the description of each word as a node. Establishing the links between other nodes depends on using so-called dependency grammar. An example of dependency grammar and the used tags for labeling used for one of the most prominent solutions found on <https://stanfordnlp.github.io/CoreNLP/> can be found in [DMM08]. On the contrary, Constituency parsing does not rely on a dependency grammar. Instead, it generates a tree of the syntactic structure of the corresponding sentence using a context-free grammar [Jur00].

Examples of constituency parsing and dependency parsing are shown in Figure 3.5 and Figure 3.6.

- **Semantic Role Labeling:** The labeling of spans in sentences with pre-defined roles is understood as *Semantic Role Labeling*. Identifying semantic roles different parts of a sentence take

³ See http://www.scholarpedia.org/article/Word_sense_disambiguation. Last accessed: 08.12.2020.

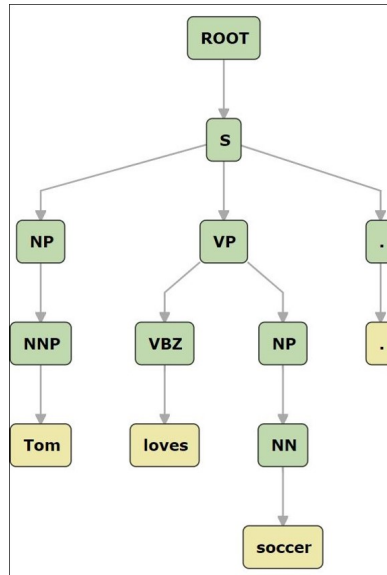


Figure 3.5: Constituency Parsing with StanfordCoreNLP. Source: based on <https://stanfordnlp.github.io/CoreNLP/> and <https://corenlp.run/>.

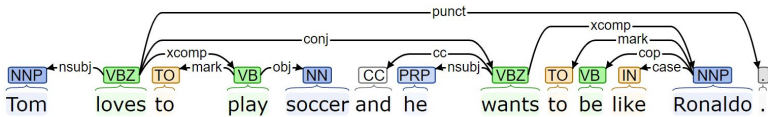


Figure 3.6: Dependency Parsing with StanfordCoreNLP. Source: based on <https://stanfordnlp.github.io/CoreNLP/> and <https://corenlp.run/>.

in the context of the sentence is a means to answer basic questions about the meaning of a sentence, such as "who did what to whom"⁴.

The primary concern of semantic role labeling is identifying the semantic relationships between a predicate and the associated participants. The identification is based on a list of pre-defined semantic roles, such as locations or names [Mår+08].

An example of the identification of two arguments and a secondary predication as defined roles over semantic role labeling is shown in Figure 3.7.

For the product , the carpenter creates the table from metal or plastic .

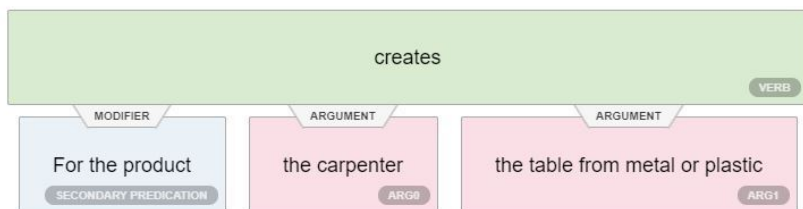


Figure 3.7: Semantic Role Labeling with StanfordCoreNLP.

Source: based on <https://corenlp.run/>.

⁴ See <https://demo.allennlp.org/named-entity-recognition/>. Last accessed: 08.12.2020.

- Coreference resolution:** Expressions that refer to the same entity are considered coreferences. The identification of such coreferences is used to add information to the textual input and especially identify the relationship of words, entities that are mentioned in different sentences. Coreference resolution is of significant relevance in different NLP tasks, such as question answering or information extraction⁵.

An example of a coreference resolution is shown in Figure 3.8, where the connection between a mentioned person and a pronoun in the following sentence is identified.

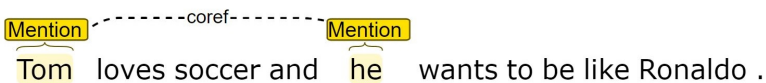


Figure 3.8: Coreference Annotation with StanfordCoreNLP.
 Source: based on <https://corenlp.run/>.

Table 3.1 provides an overview of common techniques used in Natural Language Processing (NLP) and the corresponding field of tasks they are categorized into. The relevant techniques for this thesis are stated together with a set of other widely applied techniques to establish an overview of the different sub-fields of tasks NLP encounters.

⁵ See <https://nlp.stanford.edu/>. Last accessed: 08.12.2020.

Table 3.1: Information Extraction Tasks and Techniques.

Task	Technique
Text and Speech Processing	Tokenization Text Segmentation Text-to-speech
Morphology	Lemmatization Part-of-speech tagging Stemming
Semantic	Name Entity Recognition Semantic Role Labeling Word Sense Disambiguation
Syntax	Dependency Parsing Constituency Parsing Sentence Splitting
Discourse	Coreference Resolution Topic Segmentation and Recognition Sentiment Analysis Text Classification

3.2 Examples

To provide a better understanding of the relevance and impact of NLP, the following section includes different examples of the practical application of NLP. The section includes six examples, three from projects originating in academia and three taken from practical applications.

3.2.1 Industry

NLP has been embedded into a broad spectrum of (commercial) solutions up to this day. In the following, three examples of NLP common industrial application areas are briefly described to emphasize the practical relevance of NLP and the different related techniques.

Chatbots

One of the industrial realizations of NLP can be found in so-called chatbots. A chatbot is a, often commercial, software application used to conduct a written or spoken conversation with a human counterpart. It builds on the exploration of available data from knowledge bases and activity logs with the help of NLP techniques. Companies use these chatbots to provide a direct contact channel for their customers, patients, and employees without the urge to invest in human operators [BF17; SA07]. In this regard, chatbots became most popular in customer service to provide the mentioned additional or less investment heavy communication channel over, e.g., the company's website. They are used to provide the customer with directions, answer pre-defined questions and automate tasks such as collecting relevant data, conducting surveys, or providing initial guidance to the questions or problem a customer might have. Consequently, a chatbot is supposed to reduce the time a real person has to spend on communication with the customer.

Intelligent Virtual/Personal Assistants

While chatbots have proven their use in a commercial and business context already, intelligent virtual assistants (IVA) or intelligent personal assistants (IPA) already found their way into almost each person's private environment. The IVA is a software agent that, based on the individual commands it receives spoken or written, performs tasks, and provides services to the user [CL18]. Nowadays, IVA can be found integrated into the most common brands of mobile phones or other devices, such as tablets, laptops, or computers. Most prominent solutions can be found in the form of assistants such as Siri from Apple, Alexa from Amazon, or Google Assistant from Google. Personal assistants in this context enable more comfortable use of the device by supporting voice commands via speech recognition.

Machine Translation

With regards to the historical origin of NLP and the Georgetown Experiment [Hut04], Machine Translation is still an intensively investigated field. As a sub-field of computational linguistics, it comprises the use of software to translate initially just written, but nowadays also spoken natural language into another language [HS92]. Known examples include the Google Translator, which translates texts between languages and can translate spoken language in real-time. An example of the Google Translator is shown in Figure 3.9, where a translation between the English and German language is shown. In software designed for an international market, translations are often based on such automated translations

of the original language, even though it often has to be refined to match the language specifics on the level of a native speaker.

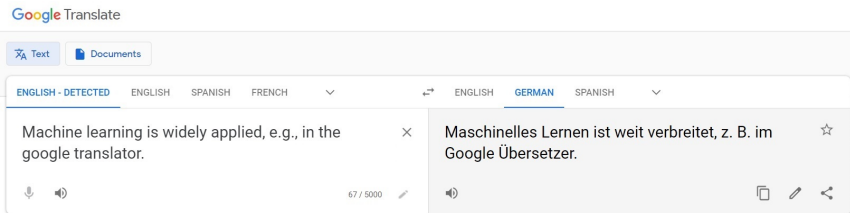


Figure 3.9: Machine Translation in Google Translate. Source: based on <https://translate.google.com/>.

3.2.2 Academia

On the other side, NLP still receives attention from research. In the following, three examples of academic driven research projects are described to point out recent topics and show either promising advancements or present significant problems to overcome and solve.

Propaganda Detection - Project PropStop

PropStop is a collaborative research project funded by the German Federal Ministry for Education and Research (BMBF). The project aimed at studying hidden propaganda dissemination in online media to identify and prove accordant attempts⁶. The interdisciplinary project team included researchers from the fields of

⁶ See <http://www.propstop.de/>. Last accessed: 08.12.2020.

statistics, communication science, IT-security, as well as journalists and IT-security companies.

Next to the large-scale analysis of propaganda and its characteristics in an online environment, public opinion statements from different digital and public platforms were analyzed towards semantic and technical patterns. Insights gained from the analysis of a vast collection of data were used to improve the detection of hidden propaganda and to develop mechanisms and technologies for the identification and verification of (online) propaganda attacks [Gri+17].

Stanford NLP Group

The Natural Language Processing Group at Stanford University is a team of scientists that focus on the development of algorithms that enable computers to process, understand, and generate human languages. Their work covers fundamental research in fields such as computational linguistics over the design of algorithms intended to solve specific language-related tasks up to the practical application of human language technologies. Tasks and challenges they address include topics such as sentence understanding, automatic question answering, machine translation, syntactic parsing and tagging, sentiment analysis, and models of text and visual scenes. Furthermore, they investigate the application of NLP to digital humanities, and computational social sciences⁷.

⁷ See <https://nlp.stanford.edu/>. Last accessed: 08.12.2020.

PAMBot

The PAMBot⁸ is based on a project conducted by the Information Systems Department of the Westfälische Wilhelms-Universität (WWU). The project's goal was to replace the current way of communicating information about exams and study regulations in a more accessible way, including a chatbot that can be used by the students. The PAMBot is used as an access point for students to inquire information about study relevant topics. The project incorporated different techniques from the field of NLP to understand textual input given by the user, to analyze the input by identifying the topic, e.g., the lecture or exam information are requested about, and to provide an adequate answer. Additionally, the integration of relevant and available data provided by the university in the form of the examination dates and descriptions, lecture register, and student data was crucial to offer a meaningful data foundation for the chatbot's different components.

3.3 Related Concepts

Different other research fields show similarities with the field of NLP, either by addressing similar problems or using similar approaches to their solution. As they may share common concepts, tasks, and challenges, such as the data processed or the algorithms used, a selection of these fields is briefly introduced. Related concepts to NLP are:

⁸ See <https://pambot.uni-muenster.de/>. Last accessed: 08.12.2020.

1. **Information Retrieval and Extraction:** Information Retrieval explores different sources of information to retrieve information to provide an answer to a specific question. Questions aim at specific information that the user intends to acquire, where the user defines "*what to search for*". A database query or a search engine are typical examples. Information is retrieved in a human-understandable form [MSR08]. Information Extraction is focused on the automatic extraction of structured information from machine-readable documents [App99; MT00] that are of a more generalized nature, e.g. "*Get all mentioned cities from this document*". Contrary to retrieving information, the extraction does not necessarily provide an understandable response for the human user but rather focuses on formats for further computer-supported processing. NLP techniques can be viewed as a means of information extraction, as they aim at extracting information from a text. Techniques for the acquisition of these texts, e.g., web crawling, belong to the field of information retrieval.
2. **Text Mining:** The computer-supported discovery of information and the automatic extraction of this information from written text is called *Text Mining*. Next to the identification and extraction of information, the connection of the extracted information to form new facts and knowledge is a significant contribution [Hea03]. Text Mining uses traditional Data Mining and analysis methods to process textual inputs for information extraction [BDF05]. The different methods originating from Data Mining and Data Analysis have proven useful in NLP research.

3. **Language Technologies:** The topic of Natural Language Processing (NLP) as the core of this thesis falls together with Computational Linguistics (CL) and speech technologies under the umbrella term of *Language technologies*, often also referred to as human language technologies (HLT). Language technologies revolve around computer-supported methods that enable the analysis, production, or modification of human text and speech. It covers and requires knowledge of a spectrum of different fields, such as linguistics and computer science [Usz00]. Investigating the other sub-fields of language technologies can provide helpful information to support progress in the field of NLP.

4. **Query Generation and Expansion:** Query expansion and generation are processes considered in Information Retrieval that aim to generate search queries to retrieve information or extend these with additional terms to improve the retrieval of information. Additionally, the mismatch or so-called false-positives acquired by the query can be reduced by the refinement through query expansion⁹. As in NLP the retrieval of information from textual inputs can be viewed from a query-based viewpoint; query generation and expansion offer helpful insights in creating information retrieval techniques for texts.

⁹ See <https://nlp.stanford.edu/IR-book/html/htmledition/query-expansion-1.html>, <https://nlp.stanford.edu/IR-book/html/htmledition/query-expansion-1.html> and [MSR08]. Last accessed: 08.12.2020.

3.4 Related Implementations

For the application of NLP methods and techniques, the approach presented in this thesis utilizes already existing external solutions for specific tasks that are to solve for transforming a textual input into a process model in the form of a specialized Petri net, a Horus (Procedure) Model.

The architecture explained in Section 6.2 will employ different supporting libraries. The requirement and reason for employing these will be elaborately explained in Chapter 6.

Grammatical information extraction from a text is nowadays mostly based on machine learning. Different techniques generate models out of training data that describe the structure of a specific natural language, which in turn can annotate each word in a new input text with predictions about their word type, role, and dependencies [Col+11]. Creating such a model requires extensive knowledge about linguistics, statistics, and computer science as Manning and Schütze describe comprehensively in their book [MS99]. These models usually result from complex statistical operations, requiring a considerable amount of input text and computation power.

As this thesis focuses on the automatic extraction of Horus Models as Petri nets from a natural language text, the architecture will utilize publicly available pre-compiled models to provide annotations onto the input text. Pre-compiled models are provided with existing solutions, such as SpaCy¹⁰, CoreNLP¹¹ or NLTK [LB02] and are often trained on publicly available data, such as blog entries,

¹⁰ See <https://spacy.io/>. Last accessed: 08.12.2020.

¹¹ See <https://stanfordnlp.github.io/CoreNLP/>. Last accessed: 08.12.2020.

Wikipedia articles or Twitter messages. The accuracy of correctly performing part-of-speech tagging or dependency parsing for, e.g., the pre-trained model of the English language available for SpaCy, is described as above the percentage of 90.

The main tasks are to extract grammatical and word-based information from the text and establish connections between words, phrases, and sentences. To accomplish these tasks, this thesis exploits a diverse set of tools available with intersecting capabilities. Table 3.2 provides a small overview of sample information extraction related software. The table shows the selection of available tools. Further and more specialized or generalized approaches can be found online of which some find application in specific sub-tasks. Most of the available tools are open source, share similar characteristics, and often include pre-compiled models. They mostly differ in their application domain or the specific sub-field of NLP they address. No significant qualitative advantages for using any tool over another were identified throughout this thesis. However, this might be because results depend highly on usage, implementation of the interfaces, and implementation objectives.

As one of the most used tools in research projects and due to the support of all structures required, a combination of SpaCy, NLTK, and CoreNLP will be utilized for this thesis. All three solutions together provide a portfolio of techniques required to tackle tasks in the field of NLP.

CoreNLP offers easy handling and provides a simple to address API, and is used to acquire constituency trees as well as to annotate coreferences. The constituency parsing enables an alternative approach to the extraction of phrases. The annotation of coreferences is used in the resolution of coreferences as part of the text

pre-processing. The Stanford NLP group provides a tool for the visualization of their different annotations used during the process of implementing different features¹².

Most of the used techniques in this approach stem from the library SpaCy. The different available operations include sentence splitting, tokenization, POS-tagging and dependency parsing based on a set of pre-trained language models. The tokenization, POS-tagging and sentence splitting are used in the identification of sentence parts, such as subordinate clauses. Dependency parsing provides information about the relations between the individual words and is relevant to extract model elements and their connections later. Additionally, SpaCy provides a tool for the visualization of their dependency parsing that is used to support the different functions provided by the implementation¹³.

Next to this, the collection of pre-trained models contain a range of widely spoken languages. Similar to other tools, there are various lexical tools and interfaces available online. SpaCy offers a set of pre-trained models of the English language. Two of them are the `EN_CORE_WEB_SM` and the `EN_CORE_WEB_LG` models. The first is based on a smaller data set, thus faster to load, but still provides a satisfying level of accuracy. The second is based on a more extensive data set, requires consequently more time to load, but has higher accuracy than the first model. As both models are operating with an accuracy level of around 90% for syntax tagging, they are both acceptable to use in this approach. When run time is preferred, `EN_CORE_WEB_SM` should be used and when accuracy

¹² See <https://corenlp.run/t>. Last accessed: 08.12.2020.

¹³ See <https://explosion.ai/demos/displacy>. Last accessed: 08.12.2020.

is prioritized, `EN_CORE_WEB_LG` is the preferred choice. Switching between models requires the adjustment of a few lines of codes and thus is possible with minimal effort.

The NLTK library covers most of the CoreNLP features but provides these in a Python environment and enables an easier integration into the prototype based on Python. Furthermore, NLTK offers the integration of WordNet, which is used to acquire synonyms of words and perform lemmatization and stemming of words in certain functions.

Table 3.2: Information Extraction in existing Solutions.

Name	Language	Features
NLTK [LB02]	Python	Stemming and Lemmatizing Tokenization Constituency Parsing Named Entity Recognition Sentiment Analysis Text Classification
spaCy	Python	Sentence Splitting Tokenization Part-of-speech Tagging Dependency Parsing Visualization
Stanford NLP [DMM08]	Multiple	Part-of-speech Tagging Tokenization Part-of-speech Tagging Coreferences Resolution Dependency Parsing Constituency Parsing Name Entity Recognition Word Segmentation
Gensim [ŘS11]	Python	Topic Modelling Document Indexing Similarity Retrieval Text Summarizing
TextBlob [Lor18]	Python	Noun Phrase Extraction Part-of-speech Tagging Lemmatizing Sentiment Analysis Constituency Parsing

Part II

Automated Model Creation from Text

4 Problem Specification

In this chapter, the research gap and the problem addressed in this thesis are described. Related work is introduced, and aspects related to the problem are emphasized to provide a localization of this work among the current research. Then, the challenges of automatic process model generation from the text are explained. Lastly, the contribution of this work to the research field is stated.

4.1 Related Work

While the automatic analysis of the natural language used in texts has been researched intensively in the past, its application to process modeling has experienced a relatively scarce attention until recently. Five projects are comprehensively described by Riefer, Ternis, and Thaler in [RTT16], which provides an overview of research projects that mainly deal with the extraction of BPMN models from texts. At the same time, other publications address this field from a less modeling language-focused viewpoint.

In the following, the existing approaches dealing with text-to-model transformation or model-to-text transformation are introduced, followed by publications that address specific problems and challenges of these transformations. The selection of publications

is based on the similarity to and their relevance for this work. An overview of the investigated publications is provided in Table 4.1, while the relevant topics of touched in the individual publications are listed in Table 4.2.

4.1.1 Existing Approaches

Different approaches address the task of text-to-model transformation directly of which some are already summarized and analyzed in [RTT16]. Similar to Riefer, Ternis, and Thaler related approaches are described in the following based on three characteristics of text-to-model transformation approaches, namely **Textual Input**, **Text Analysis** and **Model Generation**.

Process Model Generation from Natural Language Text [FMP11]

Friedrich, Mendling, and Puhmann introduce an automatic approach to generate BPMN models from natural language text to tackle the time-consuming step of Business Process Modeling in the context of Business Process Management. Their approach combines existing tools from natural language processing and adjusts them with a suitable anaphora resolution mechanism.

• Text Input

The authors declare the required input as textual without further specification of the format. The input is restricted by structures that cannot be processed, such as questions, the sequence of the described process, which should be sequential and without non-

Table 4.1: Overview: Related Work.

Title	Authors	Date
Processing Natural Language Requirements	Ambriola, Gervasi	1997
Process Discovery from Model and Text Artefacts	Ghose, Koliadis, Chuang	2007
Generating Natural Language Specifications from UML Class Diagrams	Meziane, Athanasakis, Ananiadou	2008
Tell us your process: A group storytelling approach to cooperative process modeling	Santoro, Borges, Pino	2008
Business Process Mining from Group Stories	Goncalves, Santoro, Baião	2009
Use Cases to Process Specifications in Business Process Modeling Notation	Sinha, Paradkar	2010
Text2Test: Automated inspection of natural language use cases	Sinha, Sutton Jr., Paradkar	2010
Semi-automatic generation of UML models from natural language requirements	Deeptimahanti Sanyal	2011
Process Model Generation from Natural Language Text	Friedrich, Mendling, Puhlmann	2011
Supporting Process Model Validation through Natural Language Generation	Leopold, Mendling, Polyvyanyy	2014
Zur Nutzung von Techniken der Natürlichen Sprachverarbeitung für die Bestimmung von Prozessmodellähnlichkeiten	Niesen, Houy	2015
Automatic Process Model Discovery from Textual Methodologies	Epure, Martín-Rodilla, Hug, Deneckère, Salinesi	2015
WoPeD goes NLP: Conversion between Workflow Nets and Natural Language	Freytag, Allgaier	2018
Assisted Declarative Process Creation from Natural Language Descriptions	Lopez, Marquard, Muttenthaler, Stromsted	2019
Extracting Declarative Process Models from Natural Language	van der Aa, Di Ciccio, Leopold, Reijers	2019

sequential references, as well as the point of view of the description, which should be from the actors perspective.

• **Text Analysis**

The core of the syntactical analysis is based on the Stanford parser [DMM08] and the work of the Stanford NLP Group, while the semantical analysis makes use of tools such as WordNet [Mil95] and FrameNet [BFL98]. They provide their own solution for an anaphora resolution mechanism and a keyword-based approach to identify the control flow of the described model. A detailed overview of the approach and included text analysis and model generation is shown in Figure 4.1. Based on a combination of the mentioned tools, references are resolved, and actors, verbs, objects, and conjunctions are extracted from the text. The individual elements are then combined according to the identified relationships between them.

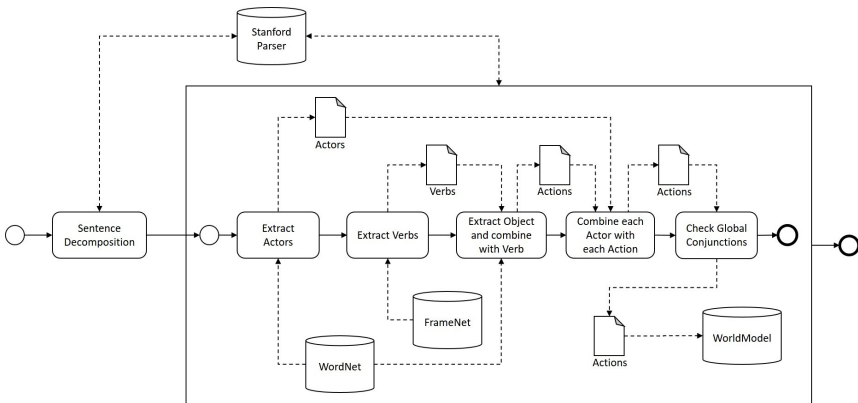


Figure 4.1: Steps of the Sentence Level Analysis. Source: based on [FMP11].

- **Model Generation**

The approach is designed towards a subset of the BPMN elements that can be identified and then generated. First, swim lanes are created based on the identified actors. Second, activities and events are created. Third and lastly, the nodes are connected over the identified control flow. As a result, a complete BPMN model is generated. In their evaluation, around 77% of their models were generated correctly.

Process Discovery from Model and Text Artefacts [GKC07]

Ghose, Koliadis, and Chueng propose a framework, the Rapid Business Process Discovery (R-BPD), and prototype tool to query heterogeneous information resources and rapidly construct process models in a process discovery context that are intended to be incrementally adjusted to correctness by an analyst.

- **Text Input**

The input is declared as a text, but no further details or restrictions about the format are given. Furthermore, the described approach should be able to process other existing models of other formats than BPMN for a model-to-model transformation step and cross-validate two existing BPMN models.

- **Text Analysis**

R-BPD makes use of two types of analysis. First, text patterns are identified that indicate process structures. Second, a text analysis is performed with the help of the Natural Language Toolkit (NLTK) [BKL09] from Bird, Klein, and Loper. The analysis using Natural Language Toolkit (NLTK) builds on POS-tags that are annotated to

the text and a syntax tree parsed. The combination of POS-tags and syntax trees is used to extract activities and objects with the help of patterns. The control flow of the described model is identified with predefined keywords that are indicators for relevant text structures.

• **Model Generation**

Ghose, Koliadis, and Chueng focus on identifying distinct BPMN model parts rather than generating a complete and sound BPMN model. Connections between identified model elements are not generated if not found in the given input.

Business Process Mining from Group Stories [GSB09]

Gonçalves, Santoro, and Baião describe an approach that explores text mining techniques and natural language interpretation for the automatic generation of process models to tackle the time-consuming task of process modeling and improve process mining approaches by capturing and including information that is not present in, e.g., event logs in process mining solutions.

• **Text Input**

The approach in [GSB09] uses the collective narrative technique of group stories to extract relevant information from people involved in the corresponding process. Under the assumption that a jointly written process description is likely to contain more useful information than a combination of individually written statements, the authors focus on such group stories as input. The format of the input is not further specified.

• Text Analysis

The textual input is tokenized, POS-tags are annotated and parsed by a shallow parser¹ to create syntax trees. A restricted set of BPMN elements are identified over templates based on verb- and noun-phrases. The elements include activities, actors, actions, and parameters. The control flow between elements is identified over keywords. A semantic analysis regarding synonyms or other semantic characteristics is not performed. The extracted information is stored based on a variation of the CREWS scenario model and stores the information based on concepts and their hierarchy.

The steps followed by the authors are shown in Figure 4.2. The gathering of data about the process as a text is performed using group stories. The collected stories are then processed with mentioned NLP techniques to extract information and model elements. Lastly, the extracted elements are combined into a general workflow and result in a formal representation of the described process.

• Model Generation

The extracted information is mapped to BPMN models and connected as far as stated in the textual input. Generated models are not necessarily complete but serve as help for the process modeler or analyst and have to be completed manually in most cases. Thus, the model generation is not considered fully automated but semi-automated.

¹ "[S]hallow parsing techniques focus on specific parts of the text and predefined, handpicked verbs and nouns" [LCM03].

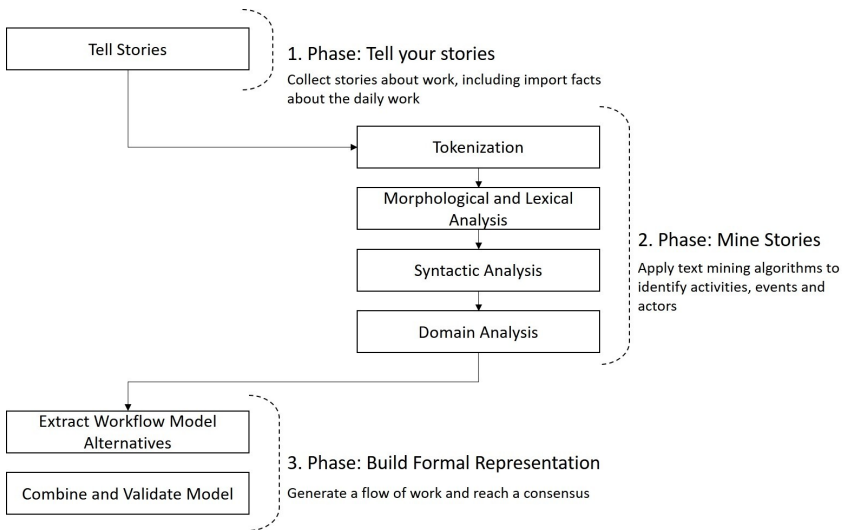


Figure 4.2: Process Elicitation. Source: based on [GSB09].

Automatic Process Model Discovery from Textual Methodologies [Epu+15]

Epure et al. address the problem of mining process models from unstructured, text-based process descriptions. They build their approach on natural language processing with a focus on verb semantics. The result is an unsupervised technique named TextProcessMiner that can discover process instance models by mining activities and their relationships.

• Text Input

The approach described by Epure et al. focuses on a textual description of archaeological methodologies. To simplify the task, only

inputs are used that describe a methodology that consists of a single process instance.

- **Text Analysis**

At the beginning of the analysis, punctuation and not required phrases are removed from the textual input. With the help of the Stanford Parser [Man+14] and the Stanford Tagger in combination with the NLTK Tagger [LB02], a syntax tree is created from the reduced text. Afterward, the textual input is analyzed sentence by sentence. Transitive verbs are identified and extracted using WordNet [Mil95], and VerbNet [PK04], as they are most likely to represent activities. The relationship between the activities is extracted according to keyword-based patterns and domain-specific rules.

- **Model Generation**

No sound or complete process model is generated in the process. Activities and the relationship among these are extracted and together with the textual representation provided as output. Moreover, a visual representation of the process model is not created. An example output is *Start -> (ACT 1 || ACT 2) -> ACT 4 -> Stop*.

Use Cases to Process Specifications in Business Process Modeling Notation [SP10]

Sinha and Paradkar present a technique to semi-automatically transform use cases in unstructured textual formats into business processes and create a mapping between them to enable the generation of process model elements.

• Text Input

The authors focus on use cases as textual input for their approach to identify specifications in Business Process Models. The applied use cases consist of a sequence of statements that describes a sequence of activities. Additionally, conditional statements can be processed as long as they are associated with the nearest preceding statement. The input format is otherwise not further specified.

• Text Analysis

Similar to other approaches, the textual input is first tokenized, lemmatized, annotated with part-of-speech tags based on an approach by Zhang, Damerau, and Johnson [ZDJ02] and parsed by a shallow parser based on the work of [BN08] [BN08] for a syntax tree. Semantic analysis is performed through an anaphora resolution that is built on manually created domain-specific databases. Keywords identify the control flow elements. All gained information is stored in a use case description meta-model containing a domain model with actors and related business items and a use case model that includes actions and actors for every sentence.

• Model Generation

For every actor identified, a swim lane is created. Based on these swim lanes and corresponding actors, the process model is build sentence by sentence from the identified process model elements. The approach supports a limited set of available BPMN elements, shown in Figure 4.3, and focuses on basic BPMN models. Additionally, to achieve a complete process model as output, a method is introduced, enabling the automated combination of single process models.

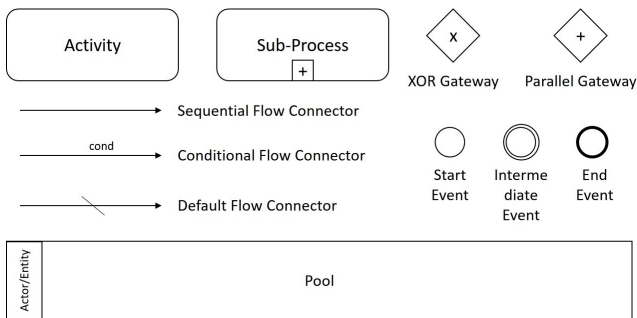


Figure 4.3: Subset of BPMN Elements. Source: based on [SP10].

The implementation of the proposed concept is presented in [SSJP10]. The authors provide the Text2Test environment that aims to generate abstract use case models from text and enables the validation, analysis, and review of the use cases and the respective models. It enables the automated extraction of use case models from the textual descriptions of these following a keyword- and template-based approach to extract use case model elements. Textual inputs are enriched with semantic and syntactic information. Based on the attached information, text parts and words are mapped onto the set of use case model elements. The connection between the individual elements is identified based on extracted references using techniques such as syntax tree and anaphora resolution.

WoPeD: Conversion between Workflow Nets and Natural Language [FA18]

WoPeD is an open-source tool for designing business processes with workflow nets, an extension of Petri nets². In [FA18], the authors extend their software with features that enable the bidirectional translation between process model and text. The Text2Process module can translate textual inputs into Petri nets building on an algorithm proposed by Friedrich, Mendling, and Puhlmann in [FMP11].

• Text Input

The input is required to be plain text without any meta information. The tool provides a text field for the user to provide the description. Features that support import from other formats, such as XML, are not mentioned.

• Text Analysis

The foundation of the text analysis is building on the approach of Friedrich, Mendling, and Puhlmann [FMP11]. First, textual inputs are split into sentences and words following the underlying grammatical rules. Second, a semantic analysis using tools, such as WordNet³ or FrameNet⁴, is performed to enrich the text with information about its linguistic features. The analysis results are then used to extract actors, actions, and business objects based on matching structures in the text.

² See https://woped.dhbw-karlsruhe.de/?page_id=271. Last accessed: 08.12.2020.

³ See <https://wordnet.princeton.edu/>. Last accessed: 08.12.2020.

⁴ See <https://framenet.icsi.berkeley.edu/fndrupal/>. Last accessed: 08.12.2020.

- **Model Generation**

All extracted information is then stored in a meta-model, the so-called WorldModel, and then translated into the Petri Net Modeling Language (PNML) to prepare the transformation into a workflow net, a type of Petri nets. The placement and arrangement of elements in the resulting workflow net have to be done manually in the workflow net editor.

Extracting Declarative Process Models from Natural Language [Aa+19]

As most of the existing approaches focus on the extraction of imperative process models from texts, Aa et al. address the extraction of declarative process models from natural language in their approach. With the extraction of declarative process models, the authors expect to capture complex process behaviors effectively. An automated approach is provided, which extracts declarative process models from text-based on identifying activities and their interrelations.

- **Text Input**

The described approach uses constraint descriptions as input. The constraint description consists of one single sentence. The input format is shown as a simple plain text but is not further specified otherwise.

- **Text Analysis**

The authors use different general-purpose NLP techniques to identify and assess linguistic patterns. They annotate the textual input with, e.g., POS-tags or syntactical dependencies. Following, they

conduct steps of linguistic processing. The verb in a sentence is extracted together with the related subject and objects. The extracted information is then extended by additional specifiers, such as negations, modal verbs, or prepositions. Subsequently, adverbial clauses and coordinating conjunctions are extracted as indicators for interrelations. Building on the linguistic processing and the identified semantic components, activities are extracted. Activities are extracted based on verb-based and noun-based structures. This extraction of activities makes use of existing approaches described in [FMP11; And02].

• **Model Generation**

Building on the results of the text analysis, declarative constraints are generated based on the extracted activities and their semantic interrelation. The generation process does not provide a complete process model nor individual process model parts, but rather declarative constraints as part of a declarative process model.

4.1.2 Related Approaches

In addition to the mentioned approaches of text-to-model, related projects, especially from the reversed process of model-to-text transformation, offer further insights into sub-tasks and -challenges of the text-to-model transformation. The described projects do not necessarily focus on specific model generation but instead on the extraction of individual elements, such as activities or connected workflows. Techniques and methods revolve around the Stanford Parser, the included NLTK tagger, and are used to derive rules from the identified syntax and grammatical dependencies.

Supporting Process Model Validation through Natural Language Generation [LMP14]

In their approach, Leopold, Mendling, and Polyvyanyy provides a process to infer texts from process models with the intermediate step of creating Petri nets from BPMN models and then text from these Petri nets. The mapping of Petri nets and text can be used bidirectionally and thus also to generate Petri nets from texts. The general architecture is shown in Figure 4.4. In the first phase of *Text Planning*, linguistic information is extracted and annotated to the corresponding BPMN process model elements with the help of tools such as WordNet⁵ or the Stanford Tagger⁶ to identify an initial structure of the intended text. The second phase of *Sentence Planning* then uses so-called deep syntactic trees to refine the text composition. The third phase of *Realization* then focuses on constructing the natural language text.

Tell us your process: A group storytelling approach to cooperative process modeling [SBP08]

Another approach in this category is focused on a specific textual structure in the form of group stories to extend an already existing approach that combines the technique of group storytelling and process modeling. Next to the proposed text mining techniques, additional concepts, such as an ontology or dictionary, are needed for a precise process model extraction from group stories. The use of dictionaries and ontologies supports the interpretation by

⁵ See <https://wordnet.princeton.edu/>. Last accessed: 08.12.2020.

⁶ See <https://nlp.stanford.edu/software/tagger.shtml>. Last accessed: 08.12.2020.

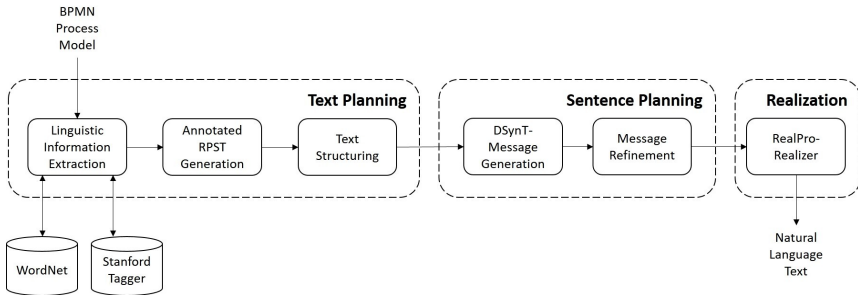


Figure 4.4: Natural Language Generation System. Source: based on [LMP14].

providing on the one side a restricted word corpus represented in the dictionary and, on the other side, a structured representation of it in an ontology. The results of their analysis produce BPMN-like workflow nets.

Semi-automatic generation of UML models from natural language requirements [DS11]

Deeptimahanti and Sanyal describe a semi-automated approach to generate UML models from natural language requirements in a given format. They build upon an intermediate step of creating use-case diagrams and analysis class models as conceptual models with the Stanford Parser [DMM08], and WordNet [Mil95] to then generate the UML model collaboratively. The underlying process architecture of their proposed technique of the UML Model Generator from analysis of Requirements (UMGAR) can be seen in Figure 4.5. Based on stakeholder requests and normalized requirements, use

case models are developed. These are used to generate conceptual models and design class models. The design class models then lead to the construction of code.

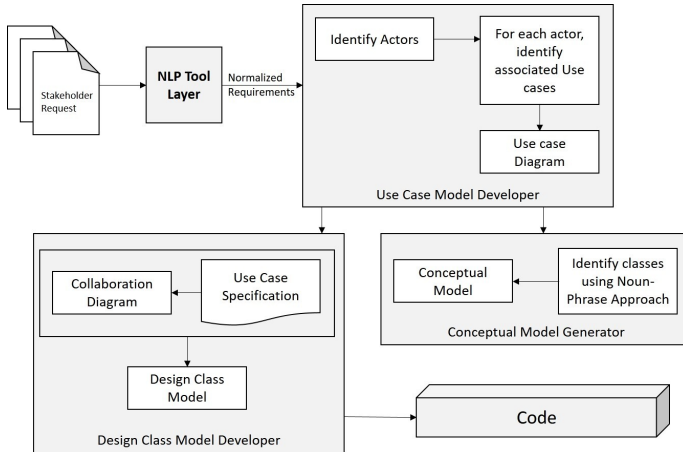


Figure 4.5: Process Architecture of UMGAR. Source: based on [DS11].

Processing Natural Language Requirements [AG97]

Ambriola and Gervasi introduce tool named Circe that provides a web-based environment for natural language requirements gathering, elicitation, selection, and validation. Next to other features, Circe can extract abstractions from natural language texts, generate different kinds of models of the systems described in the requirements documents, and check the validity of these models. The approach applied to recognize and process natural language

includes a combination of canonization and tokenization together with a rule-based approach that follows a fuzzy logic. The used fuzzy logic allows for incomplete matching, weighted independence from the order of the words, multiple matching, and several other imperfect correspondences and enhances the recognition of the tool.

Assisted Declarative Process Creation from Natural Language Descriptions [Lóp+19]

The authors describe an extension of their previously created and in [Lóp+18] described hybrid modeling tool, the Process Highlighter, that supports declarative process generation from natural language descriptions. The Process Highlighter facilitates the manual creation of *Dynamic Response Condition graphs* directly from text documents, supporting non-technical users in adopting declarative process models. Features include the use of Natural Language Processing (NLP) techniques and existing solutions, such as WordNet [Mil95], to support users in the identification of roles, activities, and constraints, as well as means to validate the created models through the automated extraction and provision of text-relevant information.

Generating Natural Language Specifications from UML Class Diagrams [MAA08]

Representing a counterpart to this thesis, Meziane, Athanasakis, and Ananiadou describe an approach to generate textual specifications for Unified Modeling Language (UML) diagrams [MAA08].

The authors analyze the natural language used to name or label components of UML diagrams. Based on the analysis, rules are extracted from the subset of natural language used within the UML diagram. Rules are then used to understand and connect UML model elements' names and labels to transform these into textual descriptions. Among different tools, WordNet [Mil95] is used to verify the generated sentences and ensure meaningful and grammatically correct texts.

Using Natural Language Processing to Determine Model Similarity [NH15]

Niesen and Houy present a concept that leverages NLP techniques to make the content of process models accessible and allows a topic-specific similarity comparison of such models. Additionally, the authors provide a brief literature overview of the state-of-the-art of Business Process Similarity. The used techniques to access content residing in textual descriptions include n-grams, POS-tags, Stemming, Term Frequency–inverse Document Frequency (TF-IDF), and tokenization.

The processing with NLP techniques is split into three phases: Preparation, content-related delimitation, and detailed analysis. First, the model is prepared and enriched with semantic information with previously mentioned tokenization and POS-tagging. Especially the transformation into a vector is described as significant preparation for later processing. Second, each vector of a model is placed into a vector space, where the number of dimensions is defined over the number of index terms of the vectors.

As a first approach to compare model vectors, the authors calculate the distance of two vectors in the vector space and enrich this calculation with the TF-IDF. The result is a collection of similar model vectors. Third and last, the as similarly considered model vectors are used to reduce the number of models that are considered similar and reduce the search space in a detailed analysis on the level of the models themselves. The detailed analysis then takes into consideration not only the meta-data represented in the vectors but the content included in the model features, such as labels and descriptions.

4.2 Challenges

This section introduces the project-specific challenges that are critical for the design and implementation of the transformation approach. The challenges are further categorized into information acquisition, language-related, Petri net-specific, and modeling conventions-related challenges.

4.2.1 Information Acquisition

One of the major challenges in creating meaningful process models includes the extraction of the information that has to be represented. While collecting and retrieving information from documents, interviews, transcripts, descriptions, or other sources of this nature, providing this information is a significant task in itself. The differences between a subsequent manual or automated modeling of a process can be considered marginal, given that the required

data is available. Following this assumption, the challenges for the approach presented in this thesis are focused on the extraction of process model-relevant knowledge from written texts in the English language. As introduced as a shortcoming to process modeling, a human modeler may not be fast or precise in many use cases. However, the modeler has a critical advantage over machines and software used to interpret texts and models, understanding natural language and its specifics.

The use and understanding of natural language are skills every human is practicing since birth. Thus, it is, except for domain-related knowledge, not necessary to explain to an external modeler how to interpret the English language. An internal lexicon, the ability to identify connections between words and sentences, references, related words, and interpret the meaning of these, can be seen as an inherent skill set of every modeler. As these actions are often performed subconsciously, defining and explaining how to accomplish these is difficult.

In consequence, getting a machine to perform these actions is an even more difficult task. A computer uses abstract, precise languages designed to avoid ambiguity and provide sets of instructions that clearly define the possible actions to perform based on the given input. Thus, getting a computer to understand and interpret natural language is a complex task representing a significant part of linguistic research. Nonetheless, initial approaches to this topic are present in research of which one will be used to classify the found challenges. Liddy proposed the notion of seven linguistic levels that were introduced in Chapter 3 and which correspond to distinct levels of understanding of natural language text [Lid01].

A machine designed for a specific translation problem does not necessarily have to be able to address all the mentioned levels but may focus on the levels that are part of the defined scope. For instance, in an approach that aims to translate texts into other languages or representation forms, the phonology might not be of relevance.

In the earlier introduced Georgetown Experiment [Hut04], the prototype did not know the English language nor the Russian language on a human-like level. Instead, it remembered all words from internal storage, the corresponding inflections, and the connection between the same words in English and Russian. The extent to which this prototype captures the seven linguistic levels of Liddy is challenging to quantify and measure, but based on the prototype's capabilities, not all levels are addressed.

When considering analyzing natural language text and transforming gained information into a model, a list of challenges and solutions can be found in the literature related to this topic. For instance, Riefer, Ternis, and Thaler [RTT16] distinguish in their survey mainly the syntactic and semantic analysis. The syntactic analysis focuses on the underlying grammatical rules and consists of three parts: Tokenization, Part-of-speech Tagging, and Parsing. The semantic analysis covers the analysis of the meaning behind the text, sentence, and word, such as synonyms, references, and roles inside a text structure. The techniques of both types of analysis are explained in Section 3.1.3 as part of the foundations of Natural Language Processing. Next to the other introduced tasks and techniques, these analysis views are often used as an aggregated approach to a specific linguistic problem.

An in-detail explained work using this analysis approach can be found in [FMP11] and [Aa+19], which are both part of the related work in Section 4.1.2. As explained earlier, they use four categories, including the analysis of syntax and semantics, to analyze natural language texts and extract different kinds of process models. Both focus on the description order, precedence, and succession and extract proper actions from the text. The two projects emphasize issues relevant to the creation of models with chronological properties, such as having a proper activity order or finding references.

4.2.2 Language Specifics

A significant number of challenges are rooted in the used language. In the following, these are stated for the English language as the chosen language for the transformation approach. However, languages with similar properties, e.g., due to their common historical origin, tend to share the same challenges. A change in the used language can be feasible when switching between languages sharing characteristics. This subsection refers to the challenges rooting in the used English language and challenges in other languages, or language family groups are hinted at. Each language inherits distinct features, such as the used word corpus, the underlying grammatical rules, or pronunciation. Languages of similar nature and origin are summarized in language families defined over features the languages share. Hence, each language or language family provided individual challenges for linguistics and included interpretation, analysis, and processing. An automated translation of a sentence into other forms of representation, such as business process models,

in consequence, has to be developed based on the chosen language to translate.

For this approach, the trade-off between the specific analysis of natural language and identifying a small set of models, elements, and patterns is essential. The modeling language maps natural language to specific patterns and templates. Thus, free use of language makes the precise creation of a model difficult and adds complexity due to, e.g., the ambiguity inherent to natural language.

Referring to the seven layers by Liddy [Lid01], the faced language-related challenges in extracting information from text can be categorized. Even though not every issue can be directly related to exactly one layer, they are assigned to the category that fits them most.

Additionally, many of the issues are related and use similar concepts. While the following list may not be complete, it incorporates the main language-related challenges of creating a text-to-model approach. These challenges are later used to create a concept for the translation and are briefly explained.

Morphological 1 – Lemmata:

A lemma is the base form of a word. These are used in different steps of processing natural language. In modeling processes, elements are often named in a structured manner and following a defined naming convention, e.g., using a verb in the present or past tense and active voice. Hence, lemmata can be used to extract a meaningful evaluation when comparing words or when exploring synonyms as the various forms may result in a different interpretation of the lexical service.

Morphological 2 – Inflections:

In addition to lemmata, the transformation approach needs to provide means to recognize inflections and provide information about deviating forms. Differences in tense, case, voice, or number influence the model mapping and generation process, especially in identifying and distinguishing actors and objects. Additionally, identifying and changing the tense and case of a word is relevant for the generation of labels for model elements.

Lexical 1 – POS Tags:

Building on M.1 and M.2, textual inputs have to be annotated with the individual words' grammatical properties inside the analyzed sentences. This annotation is performed using so-called part-of-speech tags, further explained in Section 3.1.3. The complete and correct annotation of the grammatical properties is essential for the later identification of, e.g., patterns based on keywords and defined indicators.

Lexical 2 – Synonyms:

To not tend to extensive repetitions when speaking or writing, humans often use similar words for the same objects. This phenomenon adds complexity to the transformation process as establishing the connections between different words of the same meaning is difficult. Consequently, synonyms of words have to be considered when referring to domain-related keywords (after, in parallel, while), objects, actors, and verbs inside the process. A correct distinction between words with the same meaning and words of different meaning is crucial to avoid misinterpretation.

Syntax 1 – Atomic Process Steps:

The ideal textual input for the automated transformation process would describe every process step in a separate sentence. An atomic description of one process step per sentence would facilitate the model element identification and reduce the possibility of ambiguity. The main verb, as well as the semantic references to other process steps, would be clearly defined. However, an atomic split of thoughts is rarely the case in natural language texts. Thus, a procedure has to be in place to detect the separate process steps. First, the text has to be split into different sentences. Second, descriptive sentences that do not necessarily provide information about a process step have to be handled differently from sentences describing processes. Some sentences may even inherit multiple process steps, and implications about references from and to sentences with more than one process step have to be taken into account.

Syntax 2 – Actors and Objects:

Actors and objects are often annotated to the single process steps in a process model to provide additional information about the process. Objects are, in general, part of the process steps' label, while actors are annotated. Therefore, this approach includes the extraction of objects and actors from the text to attach them to the process model and relates to L.1, L.2, and D.1.

Syntax 3 – Additional Information:

In addition to verbs, objects, and subjects mapped onto the core elements of the process model, the input text can include additional information about the process. For instance, prepositional phrases or conditional phrases for exclusive provide information about, e.g.,

exclusive splits or inputs and outputs. Both mentioned phrases and the related constructs enrich the model and are important for the complete depiction of the process.

Semantics 1 – Word Relations:

Building on the information provided by the POS-tags, the relationship between the single words have to be interpreted and analyzed. The underlying information about the corresponding process step is highly dependent on the grammatical reference of a word. A typical step includes the creation of a syntax tree for a structured representation of the identified dependencies.

Discourse 1 – Anaphora Resolution:

Next to the analysis of individual sentences and process steps, references within the text have to be considered. Anaphora resolution is considered an integral part of process model translation in several already established works [RTT16; FMP11]. Per definition, an anaphora is "the use of a word referring to or replacing a word used earlier in a sentence, to avoid repetition"⁷. In this thesis, anaphora resolution will primarily include the links between pronouns and their references within the text to clarify, especially actors involved in the process steps.

Discourse 2 – Further References:

Other references within the text, besides anaphoras, are of relevance for the translation. While references are often declared as a sub-category of anaphoras in literature, these references to other process steps are listed explicitly in this thesis as the challenges

⁷ See <https://www.lexico.com/en/definition/anaphora>. Last accessed: 08.12.2020.

and meaning differ in the context of the text-to-model transformation. In case of a reference to another sentence, which may indicate a relation between two steps and patterns, such as loops, the referenced process step has to be identified in the process model. Other challenges, such as mentioned synonyms in *Lexical 2*, impact the analysis of references and, in general, add complexity to the identification of the right referenced aspect.

Discourse 3 - Control Flow Indicators:

Regarding the previous challenge, one has to consider explicitly described sequences and the relations between sentences. A keyword, such as, e.g., "after", "subsequently", or "before", is used in the text to clearly define the order of execution.

Discourse 4 - Non-Sequential Descriptions:

Process models are intended to be structured and described sequential but may split at certain process steps. These splits have to be identified, mapped to the process elements, and inserted into the right place in the process model. Identifying the location of such a split in the process model and where it refers to relies on keywords that have to be identified beforehand, such as, e.g., "meanwhile" and "if", as well as predefined constructs.

Pragmatics 1 - Implied Sequences:

A textual description may not depict all information about final model explicitly but may have some implicit, hidden information. For instance, in the description of a sequential process model, process steps may be executed one after another without explicitly stating this in the description. These implicitly stated and assumed

sequences have to be identified and connected to the other process steps.

Pragmatics 2 - Fill Phrases:

With reference to *Syntactical 1*, phrases may not refer to an atomic process step and should be handled differently or even be ignored. These phrases may include no valuable information for the process or references to the process itself, such as the description of the start or end of it.

Depending on the information carried by the phrase, different actions have to be performed. Sentences that are not atomic but include multiple process steps should connect the process steps in question with the ending place without creating another transition.

4.2.3 Petri Net and Horus Model Specifics

After extracting all the required information from the text, it has to be mapped to the model elements of the Petri net-based Horus procedure models. Two properties are considered for the mapping algorithm:

1. The resulting model must represent the statements from the input text in a meaningful way. This requirement includes, additionally, that no information is left out.
2. The mapping is adhering to the syntax and rules of modeling Horus procedure models that are building on the general Petri net notation.

Satisfying both properties can be difficult, as certain information inherent in the text may not be expressed through the available

model elements and patterns. Other modeling languages that provide a wide range of elements, such as the Business Process Model and Notation (BPMN), might allow a more detailed representation of the described business process. When modeling PNs or PN-based models, the number of model elements is rather small, as the basic notation introduced in Section 2.2 includes only places, transitions, and arcs as structural elements. Complex structures have to be described in PNs with a limited set of elements available. All relevant structures, like connectors or events that other modeling languages, such as the Event-driven Process Chain (EPC) [STA05] or the BPMN [Whi04] provided, must be represented through the use of only the available three components in PNs.

Horus procedure models provide some additional capabilities in this regard, as they include additional elements, such as roles, or additional structures, such as exclusive choices within activities. Horus procedure models use an extended set of elements also to express complex structures and promise a more detailed representation than Petri nets.

Even though it is a challenge to express all required structures over a still limited set of elements compared to other process modeling languages, these structures are recurring in models, similar to often-used phrases in the English language. In consequence, standardization and the modular definition of phrases and structures is an often pursued goal.

This set includes, on the example of Petri nets, lists of common patterns, depicting the correct representation of arrangements like splits, joins, or loops. For Petri nets, a set of widely used patterns is established [LVD09]. However, it is not exclusive to PNs as patterns are used in various fields, such as software engineering, to define

and depict recurring structures. Even though these PN patterns are not considered an official standard, they are commonly referred to in research and widely applied in practice.

Van der Aalst and Hofstede introduce a set of patterns they collected over the years while working with PNs. A comprehensive list of these is available online⁸ and split into categories, e.g., control-flow patterns and exception handling patterns.

A fully functional text-to-model approach should be able to generate any of these patterns based on a given textual description of the process. However, complete coverage of all patterns considering a satisfying precision in the transformation is a complex goal challenging to achieve.

Examples of regular and basic patterns are provided by multiple authors, mostly including control-flow patterns. Control-flow patterns describe the sequence of activities and are therefore essential for modeling processes. They are also the initial patterns Van Der Aalst et al. [VDA+11] introduce in their work and are consequently utilized in most Petri nets.

Defining a selected set of patterns serves two purposes in this approach:

1. Individual patterns, such as "Parallel Split" or "Exclusive Choice", indicate how to visualize the underlying theoretical concepts. They support the use of the PN notation and establish a relation between the recurring structures in natural language texts and the corresponding PN structures.

⁸ See <http://www.workflowpatterns.com/patterns/>. Last accessed: 08.12.2020.

2. Splitting the structure of PNs into patterns allows for a modular, rather atomic design of a parser's different features. Keywords and phrases can be used to identify and map individual patterns. The components described in Chapter 6 are designed following this idea.

Encountering a distinct keyword or phrase then enables the parser to identify and extract distinct patterns. However, the translation between text and model is not strictly bound to the patterns. The use of patterns provides instructions on how to translate specific structures found in the process model counterparts' textual description.

The different lists of patterns proposed by Van der Aalst and Hofstede [VH02] or Lohmann, Verbeek, and Dijkman [LVD09] are not considered complete and there may be cases in which no pattern is applicable or existent for a given phrase.

As the English language may also inherit characteristics, such as the size of the word corpus, it is not easy to differentiate between specific patterns due to, e.g., ambiguity. In cases that are not clearly defined, design choices or contextual domain knowledge is required to identify the correct match between text and pattern.

More complex patterns like the "Interleaved Parallel Routing"⁹ or Exception Handling Patterns¹⁰ require a more in-depth analysis of multiple sentences and even introduce new elements to the notation.

⁹ See <http://www.workflowpatterns.com/patterns/control/new/wcp40.php>. Last accessed: 08.12.2020.

¹⁰ See <http://www.workflowpatterns.com/patterns/exception/>. Last accessed: 08.12.2020.

As a result, a parser used for the transformation would not generate these patterns if it operates with the standard notation.

These limitations in mind, a selection of patterns to be implemented, are necessary to benefit from these catalogs. As mentioned, it is unlikely to implement all patterns in the early stages and achieve a satisfying performance of the prototype.

Additionally, beforehand introduced aspects of quality of process models and guidelines to ensure this quality partly also include the complexity of a model as one measurement. Thus, a model of high quality should not be too complex, and patterns that add to this complexity should be avoided. Therefore, the focus of this approach lies in the most prevalent and essential structures.

4.2.4 Modeling Conventions

In this subsection, challenges due to modeling conventions originating from frameworks, guidelines, or best practices, such as partly already mentioned in Section 2.3.1, are described.

In general, models are used to store, visualize, and transfer information between organizational units or stakeholders. However, models based on the same information may differ in size, labeling, or statement depending on the recipient, viewpoint, or personal opinions. The different people, groups, or organizational units within a company usually wield different viewpoints when creating models. While each different version may not be wrong on its own, a meaningful comparison between two models is more complicated by each mismatch of elements. The discrepancies are especially problematic for model validation and, in this context, an automated text-to-model transformation approach. The model generation and

adjustment steps have to ensure a similar quality of the model compared to manually created models. To reduce the diversity of models and provide a certain standard in the process of modeling, guidelines, and frameworks, such as explained in Chapter 2, exist. These primarily aim at quality assurance and consistency [Leo+13]. However, no guideline is enforced or officially standardized, and various aspects have to be clarified while designing a model.

While the different frameworks and guidelines focus on and highlight aspects of different areas, several categories share similarities and distinct characteristics are useful to consider when designing an automated approach for the text-to-model transformation. Several such characteristics offer meaningful guidance in the design of the text-to-model transformation approach. However, other aspects found in the literature may not be possible to integrate, as they, e.g., require a specific domain-related logic.

The models aimed at with this approach and the corresponding prototype, therefore, should adhere to the following rules:

1. Each piece of information related to a process step available in the source text will be processed if there exists an element in the set of model elements designed to hold this information. It may not exclude any process steps fulfilling this requirement.
2. The resulting models must be designed to be as structured as possible. Following the fourth guideline of the 7PMG, every splitting pattern should have a suitable joining pattern [MRA10]. Resolving splits only with the corresponding merge also reduces the risk of creating deadlocks within the model. An exception is given in the beginning and start of a model, e.g., if a process can have two end states or start from two

- different inputs. In contrast to traditional Petri nets, Horus procedure models can have multiple starts and endings.
3. Each process step should only feature one action. If a process description requires, e.g., a document to be saved and printed, two distinct process steps must be generated. Activities are considered atomic and represent just one action.
 4. As the result of the transformation process, the model will be provided in a machine-readable format, a data frame, or a corresponding file. This structured representation can enable a better validation and assessment of this method and serves as an input for possible visualization.
 5. Most of the introduced frameworks and guidelines mention naming conventions. The essential constituents of a process model are the process steps, represented by transitions or activities. As such, these will carry the central processing logic and are labeled with their process step representation. A consistent naming scheme resulted in better readability and better comparability to other models [MRA10]. Additionally, it automatically eliminates the error-proneness through naming inconsistencies between modelers and enforces a standard. In this regard, activities of process steps extracted from the text will carry present-tense verb-object labels with optional descriptive elements afterward, for instance, "send invoice to the customer". This rule is easily applicable in an automated transformation process whenever verbs and objects can be extracted from the input.

4.3 Contribution

In this section, the potential benefits arising in different fields as a result of the automation of creating a process model from a textual input without human interaction are presented. This thesis's contribution to the research field of automated process model generation in the context of process-oriented knowledge management is stated.

Rather than looking at the full automation or complete manual creation of process models, this approach aims at different degrees of automation and the reduction of necessary manual human actions. Thus and even though full automation is the overall goal, human interaction in specific steps is necessary. This approach can thereby be considered as semi-automated at this stage.

The transformation of textual descriptions into process models with the help of NLP is rather unexplored at this point, and just a few publications in the literature address this topic directly. This transformation approach also considers the assessment of the semantic quality of process models as part of the model adjustment step, which is in contrast to the rather scarce attention from literature, a relevant topic for practical tasks for different application fields and businesses due to the widely spread and frequent use of process models as representation form for information and knowledge.

Gained insights from the investigation of existing literature of this topic as well as related topics are contributing to a deeper understanding of the relevance of process models in managing knowledge, the semantic quality of process models, and means to assess and affect these quality aspects over the model generation and transformation from natural language texts.

Building on these insights, a concept to use automated model transformation to address, among other things, the process model quality is provided. With the help and the combination of techniques from the explored fields of *Process Modeling*, *Natural Language Processing* and *Model Transformation and Generation* and the use of an ontology-inspired reference point, the transformation of textual descriptions of processes into a Horus procedure model that represents this process is addressed.

This transformation is based on a rule-based system extended by an adjustment step using an ontology-based concept map, similar to [KG08], which described an approach to automatically create UML models from texts with the help of ontologies. By using this approach, both generated models and manually created models can be compared. The comparison then supports the evaluation and adjustment of either the manually created process model or the textual description in the form of the generated process model.

Lastly, the module contributes to an existing strategy to support users in creating correct and meaningful process models, including their attached elements, such as descriptions and related models. The implementation supports the combination of direct feedback about syntactic and semantic qualities of a model. It can contribute to different further activities, such as teaching users the modeling language or model-oriented thinking.

Regarding the existing and related work done in this field, one major challenge revolves around how to achieve a correct and precise extraction of model structures and the connection of the individual elements from the underlying text. The model control flow expressed in complex sentence structures is a challenge for the analysis and transformation. This challenge is addressed by

the mentioned rule-based approach that uses pre-defined patterns based on common Petri net patterns and grammatical constructs to establish a mapping between both. The mapping enables the identification of model elements and structures residing in parts of the textual description.

Another relevant challenge lies in the implicitly mentioned knowledge about the described process model, which is difficult to assess while analyzing the textual input. An approximate solution for this is provided in the form of reference to a contextual knowledge base. The creation of process models in an automated way can be improved with existing examples that serve as a context used to recognize implicit information.

Relating this thesis to the field of text mining, it can be understood as providing distinct parts of a Text Mining System, as shown in Figure 4.6.

While the information feed normally provides unstructured content from different sources that have to be integrated and mined, this approach focuses on single-sourced and as simple plain text provided process descriptions. The intelligent tagging to enrich the textual input with linguistic information is already provided by different existing solutions that use pre-trained models for different languages to enable, e.g., part-of-speech tagging, entity extraction, dependency analysis. These tools are used for the corresponding tasks where they fit and extended when required. The Business Intelligence Suite is responsible for consolidating the information and analyze the gathered information with a given goal or purpose. This part is provided through the transformation between text and process models.

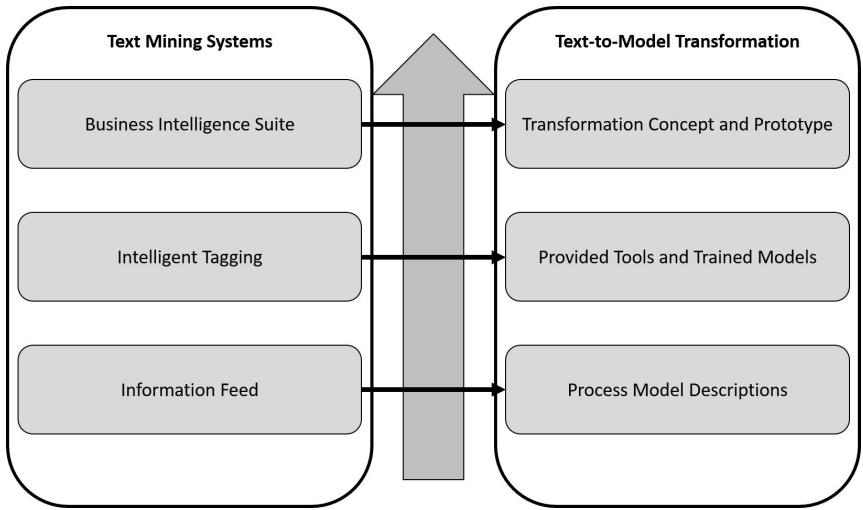


Figure 4.6: Transformation Approach compared to a Text Mining System. Source: based on [BDF05].

Additionally, this work touches but goes beyond the field of process mining. While process mining focuses on extracting information from systems event log to identify, validate and adjust business processes, the extraction here is based on textual input and additional information since a business process incorporates human activities, which will never be present in logs.

Table 4.2: Topics in Related Work.

Publication	Topics
[FMP11]	<ul style="list-style-type: none"> •Generating BPMN models from text using a subset of BPMN elements •Anaphora resolution mechanism •Keyword-based identification of model control flow
[GKC07]	<ul style="list-style-type: none"> •Generating parts of BPMN models •Pattern-based identification of process structures •POS-Tags and syntax trees to extract activities and objects
[GSB09]	<ul style="list-style-type: none"> •Identification of a subset of BPMN elements •Syntax tree and noun-/verb-phrases to identify BPMN elements •Keyword-based identification of model control flow
[Epu+15]	<ul style="list-style-type: none"> •Sentence by sentence analysis •Identification of transitive verbs for activity extraction •Parts of conceptual models are generated
[SP10; SSJP10]	<ul style="list-style-type: none"> •Generating abstract models from use cases •Keyword-based identification of model control flow •Semantic analysis based on anaphora resolution •Meta-model of extracted information •Sentence by sentence analysis
[FA18]	<ul style="list-style-type: none"> •Generating workflow nets from text •Bidirectional translation between model and text •Semantic analysis to identify model elements •Meta-model of extracted information
[Aa+19]	<ul style="list-style-type: none"> •Generating declarative constraints for process models •Subjects, verbs and objects extracted for the main model elements •Clauses and phrases to identify model patterns •Activities extracted from noun- and verb-phrases
[LMP14]	<ul style="list-style-type: none"> •Creating Petri nets from BPMN models extracted from text •Bidirectional translation between model and text
[SBP08]	<ul style="list-style-type: none"> •Process model extraction based on group stories •Combination of text structures, ontologies and dictionaries
[DS11]	<ul style="list-style-type: none"> •Semi-automated approach to generate UML diagrams from text •Automated creation of conceptual model •Manual and collaborative creation of the UML diagram
[AG97]	<ul style="list-style-type: none"> •Extraction of abstraction of natural language texts •Generation and validation of different kind of models •Rule-based approach following fuzzy logic
[Lóp+19]	<ul style="list-style-type: none"> •Declarative process generation from natural language descriptions •Semantic analysis to identify model elements and constraints
[MAA08]	<ul style="list-style-type: none"> •Generation of textual specification from UML diagrams •Analysis of model element labels to extract rules for text generation
[NH15]	<ul style="list-style-type: none"> •Vector-based approach to assess similarity between models •Semantic analysis to enrich vectors with semantic information

5 Context Description and Design

In this chapter, this research project's context is described, as this thesis is the result of a cooperation between researchers and practitioners. This cooperation considers a real-world problem motivated by an industry case that revolves around the use of one modeling method with a specific modeling tool for process-oriented knowledge management. The design of a transformation of textual input into a process model in the form of a specification of Petri nets (Horus model) is described. The general concept, the objectives followed, including the used methods and techniques, the ontology-based reference point for model adjustments based on contextual knowledge, and the steps of the transformation approach itself are explained.

5.1 Horus Method

In this section, the method of internal process-oriented knowledge management used by the business partner is briefly introduced. One main component of their method is their specialization of Petri nets, the Horus procedure models. Potential extensions of the

current solution are hinted at here while picked up again later in the outlook given in chapter 7.3.

The Horus method incorporates practices to support a business process over its whole life cycle. With the first idea, over the design, the implementation, and the ongoing execution, different models and actions are proposed to support each step [Sch+11].

On the top level, the Horus method consists of four phases that are further shown in Figure 5.1:

1. **Preparation:** The phase of preparation includes identifying the organization's parts which will be examined, the budget allocation, the definition of a time frame, and the description of the project goals.
2. **Strategy and architecture:** The modeling of the business process, business objects, and organizational structures is the primary task of this phase to set up a fitting strategy and architecture.
3. **Business process analysis:** The simulation and analysis of the previously modeled concepts is part of the business process analysis. The analysis focuses on different so-called "responsibilities" and includes the analysis of (organizational) structures, procedures, risks, and key figures.
4. **Model usage:** Beyond the documentation and modeling of the business processes, the model usage phase focuses on the process implementation, process execution, and process evolution.

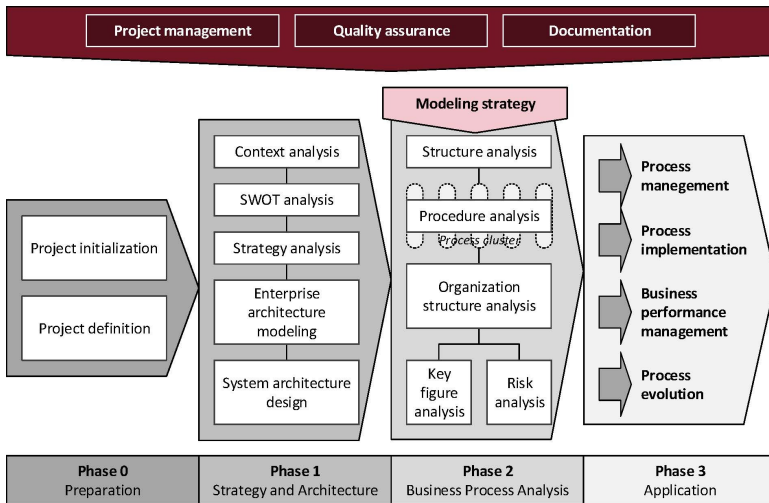


Figure 5.1: Horus Method Phases. Source: based on [Sch+11, p.62].

The Horus Business Modeler focuses on the implementation of the relevant models used in the Horus method. It provides means for systematic business process engineering, including the design, analysis, optimization, and documentation of business activities [Sch+11]. Additionally, social features enable the tool to be used as a Social BPM platform. Incorporating a set of features and models shown in Figure 5.2, the tool is based on three main components:

1. **Procedure models** are used to represent business process models in the form of a specialization of Petri nets based on the utilization of XML as a data structure. They allow for a structured visualization of business activities and correspond-

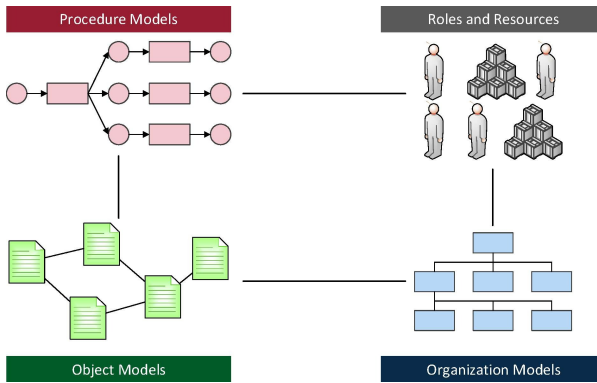


Figure 5.2: Horus Perspectives. Source: based on [Sch+11, p.28].

ing object flows. To deal with the complexity of the numerous business activities included in an organization process landscape, the Horus Business Modeler enables the refinement of tasks in more detailed sub-models [Sch+11].

2. **Object models** represent the objects used in the actual business process and the corresponding process model. They carry information about the used objects and their relationships between each other as well as their connection to the business activities.
3. As a result of the process models and their associated object models a **process-oriented organization structure** can be derived and modeled as well. Additional models are provided and can be used to represent other aspects related to business activities. For instance, it is possible to describe risks,

roles, employees, business rules, or objectives with the Horus Business Modeler in different models.

5.1.1 Horus Business Modeler

In this section, the Horus Business Modeler is introduced as a tool to create the different models used in the Horus method, includes the Horus procedure models, which are introduced afterward together with the significant differences to regular Petri nets. Stated features of this model type and the differences to regular Petri nets are relevant for the later design and implementation. An example of a Horus procedure model is shown in Figure 5.3.

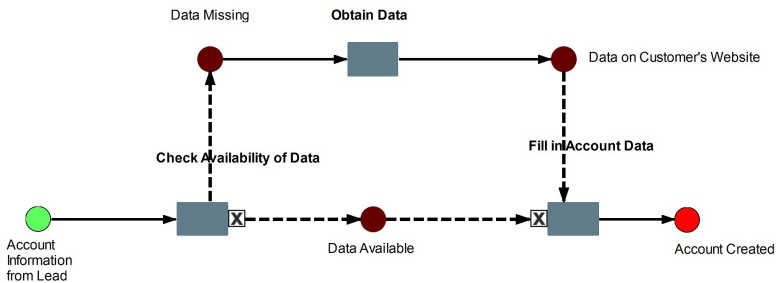


Figure 5.3: Procedure Model created with the HBM.

The Horus software GmbH, the associated Horus method, and as subsequently the Horus Business Modeler, was established based on a cooperation of the Institute of Applied Informatics and Formal Description Methods (AIFB), the Research Center for Information Technology (FZI), which are both parts of the Karlsruhe Institute of Technology (KIT), and the PROMATIS software GmbH [Sch+11].

Seeing knowledge as one of the major drivers of competitive advantages and sustainability, the capability of modeling tools to support the acquisition and the value-adding use of knowledge is an important quality criterion for the Horus Business Modeler¹. With these requirements in mind, the Horus Business Modeler was created as a tool to support the whole life cycle of processes and their process models. The promotion of participation and interaction of members of the entire business community in business process modeling activities was one of the core intentions behind the development [Sch+11]. The later introduced features of Social BPM enable the exchange of knowledge and experience within a community by utilizing modern capabilities of collaboration and communication [PV14; PV13]. Participants in such Social BPM communities can exchange, discuss, and analyze process models within the social network. Furthermore, Social BPM also provides incentives and support for implementing and executing the business processes by involving the different process stakeholders and the individual process users to participate in the process development. This technology-driven communication and collaboration not only utilize the individual and shared knowledge about the field of application to improve the business process but also familiarizes them with alterations of the process to improve the changeability of the organization². Other extensions that are the result of the cooperation between PROMATIS, Horus and different universities include the integration of a gamification module to enhance participation

¹ See http://www.horus.biz/fileadmin/horus/community/wiki/en/about_horus. Last accessed: 08.12.2020. Last accessed: 08.12.2020.

² See http://www.horus.biz/fileadmin/horus/community/wiki/en/social_bpm. Last accessed: 08.12.2020. Last accessed: 08.12.2020.

and training of users [Pfl18], a mobile version of the modeling tool that is location and platform independent [Alp+14] or the use of the Horus Method to establish a meta modeling platform [Fil+13].

5.1.2 Horus Procedure Models

Horus procedure models are one type of model that can be created with the Horus Business Modeler. They are based on Petri nets, use the same elements but are enriched over different extensions. One of these includes connecting to other models used in the Horus method, such as data models, role models, organizational charts, or models representing the system landscape. Roles, organizational units, and objects are attached to the model's activities and objects to provide additional information and reference to the other models. Another significant extension includes the use of types of activities. Activities can be regular or extended with exclusive inputs and outputs. Next to the typical representation of exclusive choices in Petri nets, as shown in Figure 5.4, the extension of activities enables the distinction between active choices over the extensions and passive choices over the use over the regular Petri net pattern. An example of the representation of active choices and the use of the extension of activities in the Horus procedure models is shown in Figure 5.5.

5.2 Concept

This section depicts the concept of the transformation approach. First, the objectives of the transformation approach are defined.

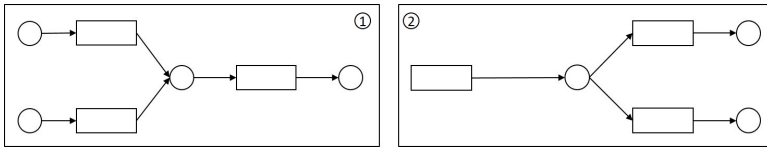


Figure 5.4: Exclusive Choices in a traditional Petri Net. Source: based on [LVD09].

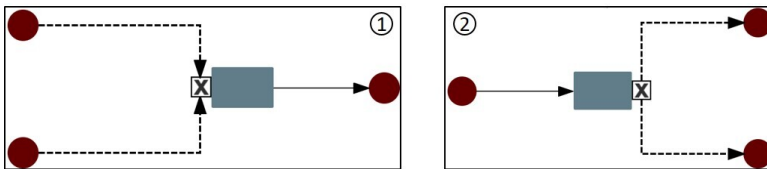


Figure 5.5: Exclusive Choices in a Horus Procedure Model.

Second, an overview of the used methods and techniques is given. Third, the idea behind the ontology-based reference point for incorporating contextual knowledge in the model adjustment step is explained. Lastly, the individual steps of the transformation approach are described.

The concept considers and builds on existing related works, especially those providing solutions for the individual problems, such as the coreference and anaphora resolution or use of patterns and templates described by Friedrich, Mendling, and Puhmann [FMP11].

Figure 5.6 shows an exemplary textual description and the corresponding Horus procedure model. Both will be used together with potentially related processes, such as, e.g., a subsequent production or sales process, in the following to support the explanation of the transformation steps. The process shown revolves around the check

of a lead. A lead activity is first received via email and then further checked. If the lead shows recent activities, it is qualified for further processing. Otherwise, it is disqualified.

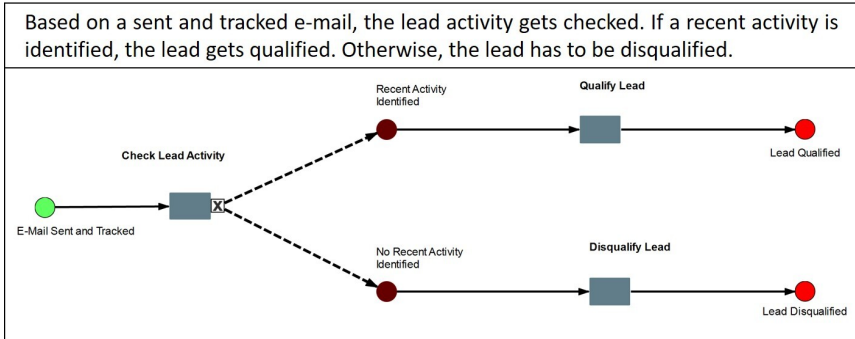


Figure 5.6: Running Example based on the Introduction.

This approach is organized in several steps that are further explained in Section 5.2.4, where the different features are used to provide the desired artifact required for further processing. The individual steps are visualized in Figure 5.7. A plain text is pre-processed with the help of linguistic analysis, and information is annotated. Following this, a set of subordinate clauses and relevant phrases are extracted together with their linguistic features. Then the extracted features are mapped onto the set of elements used in Horus procedure models. Lastly, the created model is enriched and adjusted with contextual knowledge residing in a knowledge base.

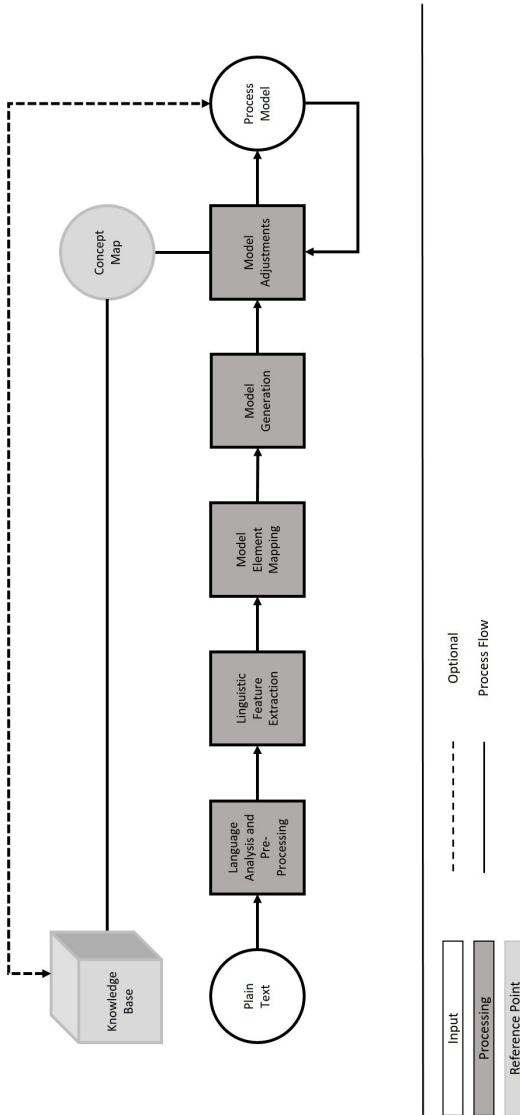


Figure 5.7: Process Steps of the Transformation Approach.

5.2.1 Objectives

This approach aims at three earlier introduced goals, which serve as objectives for the conceptual definition of the transformation process and the implementation of a proof of concept. While the automation of the transformation between texts and process models is the primary goal, a positive impact on the quality of created and adjusted process models focusing on their semantic aspects is another intended outcome. Additionally, the approach is expected to offer insights about process modeling that can support inexperienced process modelers.

Automation

This work's main objective is to explore, explain, and provide an automated transformation of natural language texts into process models. The automation of different steps to create a process model based on a text should, in consequence, reduce the required manual effort, expertise, and time that have to be put into the creation of a process model. As previously stated, a fully automated approach is challenging to achieve. The best-case scenario would require minimal manual interaction, such as just the manual provision of a text. As this is the highest possible achievement, working towards it requires some initial work to focus on distinct and smaller parts of the wanted solution. Thus, different steps are performed to approach the problem and the intended solution.

First of all, the scope of the process has to be defined to clarify which specific parts of modeling a process are to automate. Second, the different process steps and included sub-tasks have to be

identified to structure the problem and search for suitable solutions. Third, the individual sub-tasks have to be grouped and their priority defined based on the impact of possible automation to the modeling process.

In this approach, the lack of available and labeled data leads to a rule-based approach, which would be different if a more extensive set of labeled data would be available or feasible to create. In such a case, techniques and methods from the field of Machine Learning could be applied.

The approach offers the automation of different essential steps in the transformation process. These steps are described on a conceptual level and, first, implemented in modules independent of each other. Second, they are combined to reduce human interaction between the processing steps further.

Quality

Related to the automation of the modeling process, this approach addresses the quality of process models. With the automatic transformation based on defined patterns and rules, the consistency and quality of the produced model are supposed to be ensured or even increased compared to manually created models. Furthermore, the automatic creation of process models should avoid behaviors that lead to manually created process models with low syntactic quality expressed in metrics and their measurements seen in Table 2.2. Process model quality is addressed especially on the syntactic level over the automation feature and on the semantic level over the use of contextual knowledge over a reference point to adjust them manually or automatically created process models accordingly.

Additionally, the earlier introduced guidelines and an established understanding for process descriptions, together with defined rules in creating process models, help establish and ensure desired characteristics of the models leading to a higher quality of expression.

Based on the assumption that using a common way of modeling processes supports the understanding and exchange of information represented in the models, this elevates the semantic quality of a process in the context it is located in. The use of the reference point to access positive examples of process models and their corresponding description provides expectations and points of orientation when it comes to the wanted level of quality. Quality is thus understood as completeness, correctness, and consistency with other related models.

Training

A transparent transformation approach that defines and explains required inputs, performed processing, and the resulting output provides a suitable foundation for inexperienced process modelers to enhance their understanding of process modeling.

The means to acquire a process model or its parts from textual input and being able to retrace model elements back to the corresponding text parts clearly show the connection between both artifacts and how to represent the content both ways.

Consequently, the idea of process modeling over textual descriptions that are automatically transformed into process models can support the learning process of a user if sufficient transparency and explanations are provided. Potential support of training can be provided by communicating and teaching an understanding of

process-oriented thinking and establishing the connection between model, represented knowledge and knowledge management in an organization.

Beyond the transformation process, this approach can contribute to existing approaches, such as Gamification of Business Process Modeling. One approach by Pflanzl [Pfl18] focused on the evaluation of the syntactic quality of Horus procedure models to measure progress and calculate rewards. Rewards were intended to increase motivation and participation among the users and provide feedback about the created process model. This direct feedback provided an opportunity to understand and train process modeling. Extending this approach by providing means to make statements about the semantic quality of process models or recommendations about their elements can lead to a more extensive evaluation and add to the gamified experience and its impact on training inexperienced process modelers.

5.2.2 Methods and Techniques

This subsection provides an overview of the methods and techniques used in the transformation, especially from the field of NLP, and their connection to the single transformation steps. Later, these are realized either by own implementations or by already existing solutions that are integrated and, if necessary, adjusted.

A broad set of text analysis and processing methods and techniques that are partly implemented are combined in the transformation approach. The implementation of a prototype is done in Python. The packages and libraries used are either available in Python or, in the case of the StanfordCoreNLP module, offer interfaces and can

be invoked externally. The tools used show some overlap in their NLP techniques. Techniques are selected based on their fitness for the defined solution approach to keep the implementation effort for a prototype low initially. The features are selected from the portfolio of functionalities mainly offered by NLTK [LB02], spaCy³ and Stanford NLP [DMM08].

Before taking a closer look at our transformation approach, the idea of using a concept map as a reference point for facilitating model transformation in 5.2.3 is introduced. The six steps of the transformation approach are then explained in 5.2.4. This approach involves the task of processing the input text, such as tokenization of the text, splitting into sentences, extraction of model relevant text elements and structures, as well as annotation of dependencies, syntactical structures, and co-references.

5.2.3 Reference Point to Contextual Knowledge Base

A reference point generated from positive examples that already reside in a knowledge base is used in the last step, the *Model Adjustment*. The reference point in the adjustment part enables the described approach to include implicit information that cannot be extracted from the textual description directly. Positive examples include textual descriptions of processes and their corresponding process model.

Even though knowledge is already transformed into the representation of process models, the existing knowledge assessed over the

³ See <https://spacy.io/>. Last accessed: 08.12.2020.

referenced point has to be provided in a structured and organized way that combines insights from all the available process models. Different techniques can be used to assess all available positive examples and leverage the knowledge residing in the process models.

In this context, common techniques are mind maps, knowledge graphs, knowledge maps, concept maps, and ontologies. While mind maps, knowledge graphs, and knowledge graphs are somewhat less structured formats, concept maps and ontologies offer a more structured view of the represented knowledge. The three considered forms of representations are knowledge graphs, concept maps, and ontologies. They are briefly explained in the following, and significant differences between them are pointed out with a small recap regarding knowledge bases beforehand.

One general decision revolves around the underlying approach of representing knowledge. Here, a graph-based approach and two map-based approaches are investigated. While graph-based approaches focus almost solely on data, maps also consider the surrounding environment, e.g., people and applications. Graphs are often used to improve the search for and retrieval of information. In contrast, maps are used to deal with extensive knowledge bases and uncover and present the relationships between the included entities. Constructs such as these can support process modelers or designers in understanding and analyzing complex processes by identifying re-usable parts or generalizing content, e.g., reference processes [Gre+04].

Furthermore, organizing and structuring available information and knowledge about existing processes and their corresponding models can serve as a foundation to further solutions, such as

recommender systems that support the process modeler during the modeling process [KHO11].

The comparability and interoperability of business process models at a semantic level are of relevance in this context. To achieve this, the approach by Höfferer [Höf07] refers to the concepts of ontologies and meta-models and introduces a combination of both to achieve interoperability between business process models.

However, the different ways of representing knowledge are not exclusive and may find application in a combination. For instance, a knowledge graph can be seen as the result of an applied ontology⁴. In the following, methods to structure, organize and represent knowledge that were partly already introduced in Chapter 2, are described in the context of the here intended automated model creation that leverages benefits from an existing set of positive and approved examples of models.

Knowledge Base

As explained earlier in Section 2.4.1, a knowledge base is a structured and organized collection of information. It follows a fact-oriented design, while in comparison, an ontology is rather schema-oriented. The focus lies on the description of the content, with the highest possible expressiveness. An ontology as a counterpart is on the other side, focusing on the concepts behind the content, their interrelations, and attributes.

Managing knowledge residing in knowledge bases often already makes use of templates and standards to establish the ability to share

⁴ See enterprise-knowledge.com/whats-the-difference-between-an-ontology-and-a-knowledge-graph/. Last accessed: 08.12.2020.

and re-use the knowledge [NHLT17]. Such templates, or also often referred to as patterns, can be extracted from existing structural representation forms of the content, such as process models [KR15].

Ontology

An ontology is considered *"a list of concepts and categories in a subject area that shows the relationships between them"*⁵. It includes the representation, naming, and definition of these concepts and categories, their properties and relationships between them.

Ontologies include the description of the semantics of information and sources and are a key component in the integration of information as a resource [NCL06]. However, creating and maintaining ontologies is a time-consuming and challenging task. To facilitate the use of ontologies, the extraction of information from other sources is supported through structured and machine understandable representation formats, such as the Web Ontology Language (OWL)⁶.

OWL is a representation form and format to capture and store ontologies. Especially in the transformation of ontologies into other representations forms, OWL can help with its structured format. Graudina and Grundspenkis describe the use of OWL in the context of creating concept maps based on ontologies. Both represent a certain domain and use classes, concepts, and the relations between them. In addition to concept maps, ontologies express additional information over attributes for classes and their values [GG08].

⁵ See <https://www.oxfordlearnersdictionaries.com/definition/english/ontology>. Last accessed: 08.12.2020.

⁶ See <https://www.w3.org/OWL/>. Last accessed: 08.12.2020.

An ontology resembles, in many ways, a database schema, as they share fundamental similarities. However, and even though significant similarities exist, they are two different forms of representing specific content in a formal and structured way, aiming at different purposes. On the one side, an ontology aims to capture the investigated domain's meaning and understanding. A database schema defines the structure by using a formal language [Usc15], on the other side. The use and combination of both can occur to assess the domain at hand unmitigated and establish a structured provision of the contents together with insights about the meaning they carry.

Furthermore, ontologies find application in different fields, such as the here relevant topic of NLP. An approach by Körner and Gelhausen [KG08] describes how to use ontologies in the process of requirements specification instead of natural language texts. They propose the use of ontologies to recommend missing elements in the requirements specification and thus reduce shortcomings of the process, such as ambiguity.

Ontologies are often used in the creation of knowledge management systems. They serve as structural elements in these systems to establish common grounds to facilitate knowledge creation and sharing, e.g., through enhanced querying of semantic associations [FCMP03]. In contrast to, e.g., data models that depict knowledge in a similar fashion, ontologies are relatively independent from their specific application and consist of rather general knowledge.

An example of an ontology representing content from the domain of healthcare is shown in Figure 5.8.

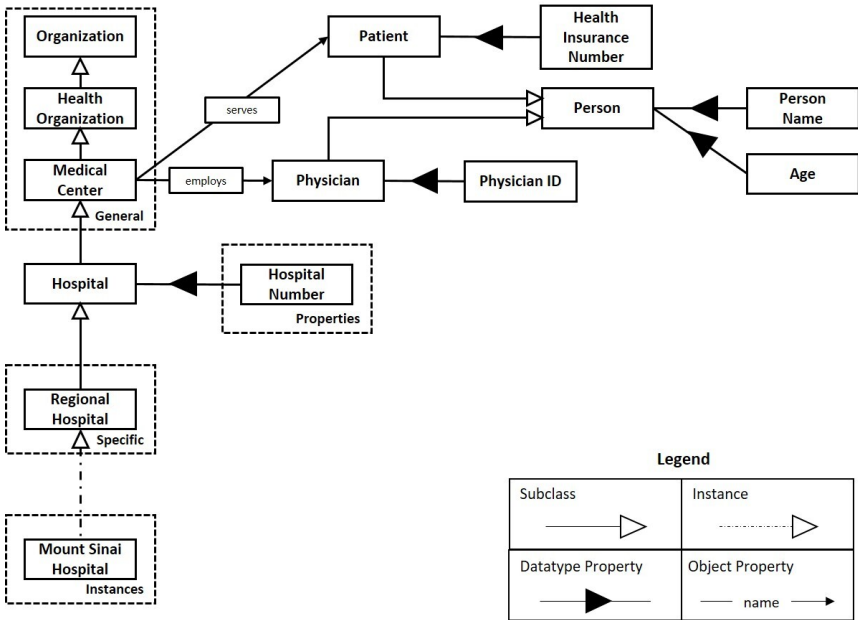


Figure 5.8: Example of an Ontology for the Healthcare Domain. Source: based on [Toc+07].

Concept Map

Concept maps are a widely applied way of representing knowledge and inherent relationships in a graphical and logical connected way and can especially be found in the educational context [Nov91; Nov95; NC06]. They are used in the educational context, e.g., for topics such as adaptive knowledge assessment to assess each student’s individual skills and knowledge level instead of lectures and

exams. The assessment with the help of concept maps enables an individual view on the single student [AGG07].

Concept maps consist of concepts understood as regularities observed in events or objects and their relationships to each other [NC06]. Concepts are part of a hierarchical structure that places the most general concepts on the top of the map. Below, all other concepts correspond to their degree of specificity. Two distinct features are essential to concept maps: The mentioned hierarchical structure and the capability to create queries to search for content and create entirely new links between concepts.

They can thus be of help in the teaching and learning environment. However, the construction and maintenance of these is often performed manually and, consequently, a time-consuming task for lecturers and teachers. The feasibility of their manual use is therefore questionable [ANGS11]. An automated creation would solve this problem, but this again is a time-consuming and challenging task to implement a corresponding system.

In another example investigated by Cañas et al. [Cañ+04], concept maps are used as an approach to establish an environment facilitating the creation and sharing of a common understanding. Experiments show, however, that the assistance provided to the user is not significant, and the effect of the use of concept maps has to be assessed and analyzed further to identify circumstances under which benefits can arise.

Concept maps themselves can occur in different types following different purposes. Thus, they can be linear, circular, a hub or spoke, a tree or a network [Yin+05].

An example of a concept map showing the key features of concept maps is shown in Figure 5.9. The hierarchical structure and

relationships are clearly visible and show the difference to, e.g., the earlier shown ontology.

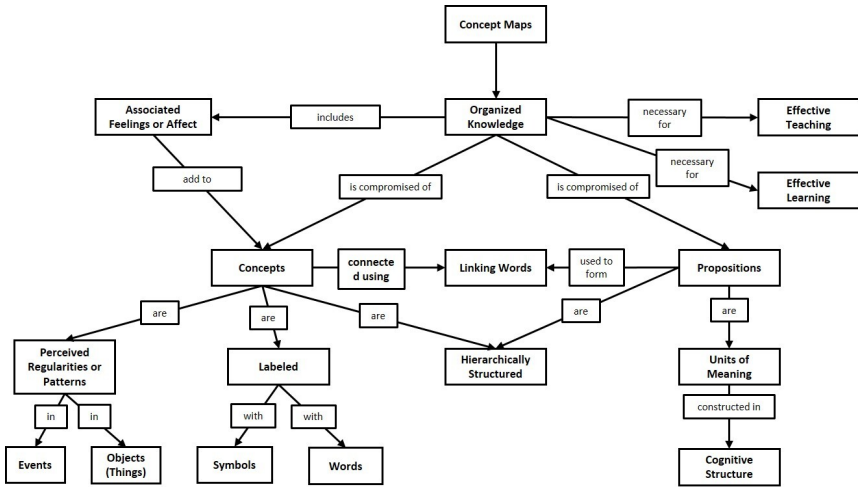


Figure 5.9: A Concept Map showing the Key Features of Concept Maps. Source: based on [NC06].

Knowledge Graph

A knowledge graph describes real-world entities and their relations to a particular topic or domain. It includes classes and relationships between the individual entities following a schema [Pau17]. Knowledge graphs are based on the graph notation but follow a relatively free form of semantics, which provides fewer constraints and less rules compared to other representation forms, such as an ontology [EW16].

In comparison to knowledge maps, knowledge graphs focus on the data at hand rather than the involved actors and their relationships. They are seen as essential to improving search and adding intelligence to applications and systems, such as chatbots or content-based systems. At the same time, knowledge maps are focused on identifying and representing knowledge and its related knowledge carriers⁷.

An example of a knowledge graph can be found in the lexical database of WordNet⁸, whose contents are structured using a knowledge graph. Figure 5.10 shows a simplified case of a knowledge graph that shows different entities and their interrelations.

From another viewpoint, one can consider that a knowledge graph is one possible way an ontology could be represented. An ontology usually deals with concepts, not instances of concepts. It incorporates meta-data and the concepts behind the content, whereas a knowledge graph focuses on the data itself. Using an ontology to represent the mentioned metadata and populating it with dynamic facts using a knowledge graph can be a side-by-side collaborative solution.

Ontology-inspired Reference Point

The individual building blocks of the reference point mentioned before are connected, and the realization, as part of the transformation approach, is explained in the following. The reference point in this approach is realized through an underlying ontology from

⁷ See <http://knowledgemanagementdepot.com/2019/05/31/knowledge-graphs-vs-knowledge-maps/>. Last accessed: 08.12.2020.

⁸ See <https://wordnet.princeton.edu/>. Last accessed: 08.12.2020.

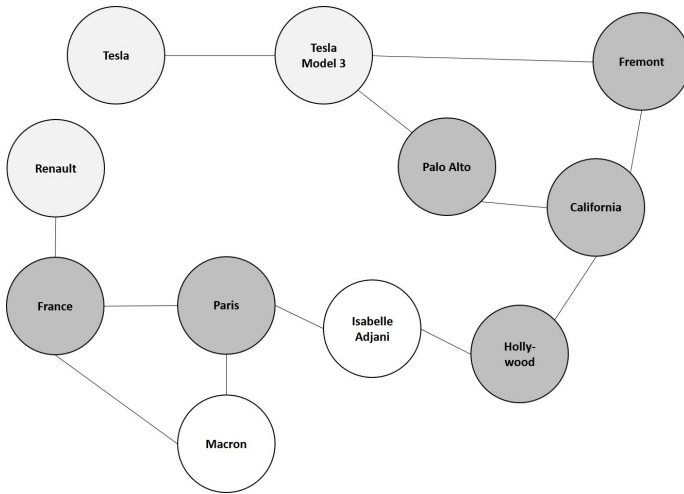


Figure 5.10: Example of a Knowledge Graph. Source: based on <https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/>.

which knowledge graphs are generated incorporating information of different instances that are represented by the individual process models. The reference point's three main building blocks are an ontology, knowledge graph(s), and instances, respectively.

Ontology Backbone: As the underlying representation of the content that resides in the knowledge base, an ontology is chosen. As for now, an ontology with the features described earlier and shown in Figure 5.8 can express more information than actually required in the current approach.

The ontology's purpose can be compared to a database schema or Entity–relationship model (ERM). It is primarily used to describe the

relationship between the data present in this scenario. Additionally to a database schema or an ERM, an ontology provides information about the meaning of the described object.

The use of an ontology leads to a generalized representation on a conceptual or meta-level. In consequence, this design choice is made with the Horus method in mind. As introduced earlier, the Horus method does not only include process or procedure models. However, it uses different kinds of models to represent the different aspects of a process during its life cycle. Thus, keeping the underlying structure abstract enables integrating other model types, such as data models, as part of possible future extensions.

Figure 5.11 shows a simplified example of the ontology in the context of this work. It describes the relationship between the individual model elements as entities in the ontology similar to a meta-model. The high level depiction and abstraction enables possible future extension to other model types and the relationship between their common entities. The individual entities are represented by different shapes and labeled accordingly. The elements of the shown example are:

- Square with sharp edges - Process Description
- Square with round edges - Model
- Circle - Activity
- Pentagon - Object (Input or Output)
- Hexagon - Actor

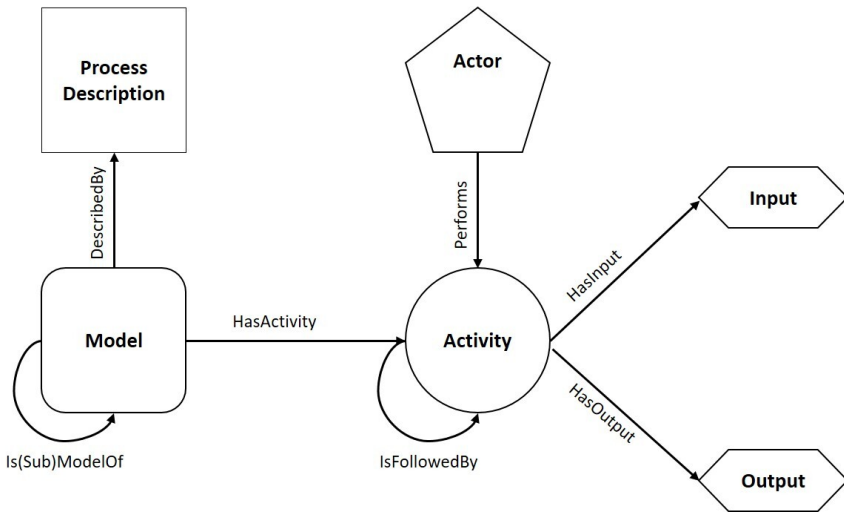


Figure 5.11: Ontology-based Schema of the Knowledge Base.

Knowledge Graph(s): The actual representation of real data described over the ontology is realized through knowledge graphs.

The ontology can be represented through one comprehensive knowledge graph, but more likely will be focused on different graphs that, e.g., incorporate a subset of the ontology, to focus on a particular perspective, scope, or problem. Additionally, the knowledge graph can be seen as a combination of all incorporated instances and process models. Thus, it can also be disconnected, which would lead to multiple graphs.

Regarding future extensions, knowledge graphs can include information that might not be found in the related instances. As instances can represent the different models used in the Horus

method, they also use a different subset of the concepts represented in the underlying ontology.

An example of a knowledge graph based on a small knowledge base is shown in Figure 5.12. Not only does it describe the relationship at the conceptual level but, in comparison to the underlying ontology, incorporates the specific data, e.g., all activities instead of just the concept of an activity.

Instances: While knowledge graphs represent a combination and summary of real data based on the structural definitions from the ontology, *Instances* are individual and real occurrences of a combination of the information of a knowledge graph. Two possible instances of the knowledge graph from Figure 5.12 are shown in Figure 5.13 and Figure 5.14. Both figures can be understood as a specific process model residing in the knowledge base, while the previous knowledge graph can be understood as a combination of both.

Information Extraction

Based on the created and used ontology-inspired background, the knowledge graph(s), and the instances, information is extracted and used to adjust and enhance the corresponding process model. For the process model and the related textual descriptions, information about three main adjustment options are extracted, which are described next. The options are addressed in the model adjustment step and can either be applied to an existing model or the model generated from text. First, the adjustments focus on the existing elements that may have to be changed. Second, elements that are not present in the model but should, be based on the related knowledge

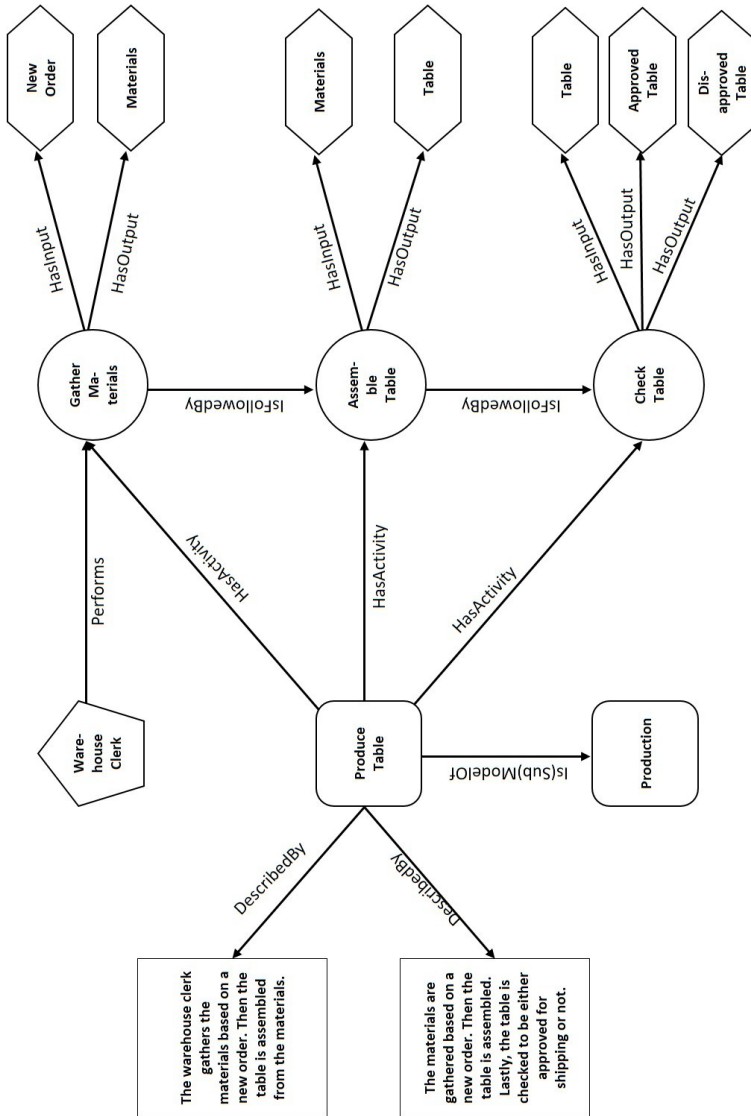


Figure 5.12: Example of a Knowledge Graph based on Figure 5.11.

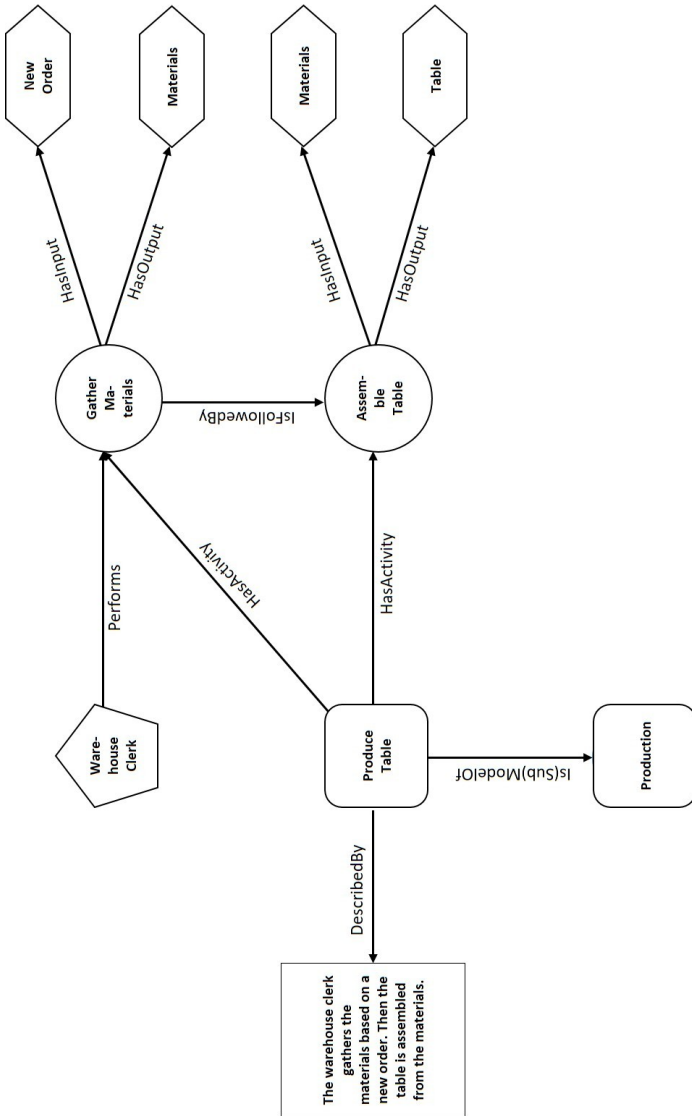


Figure 5.13: One Instance of the Knowledge Graph shown in Figure 5.12.

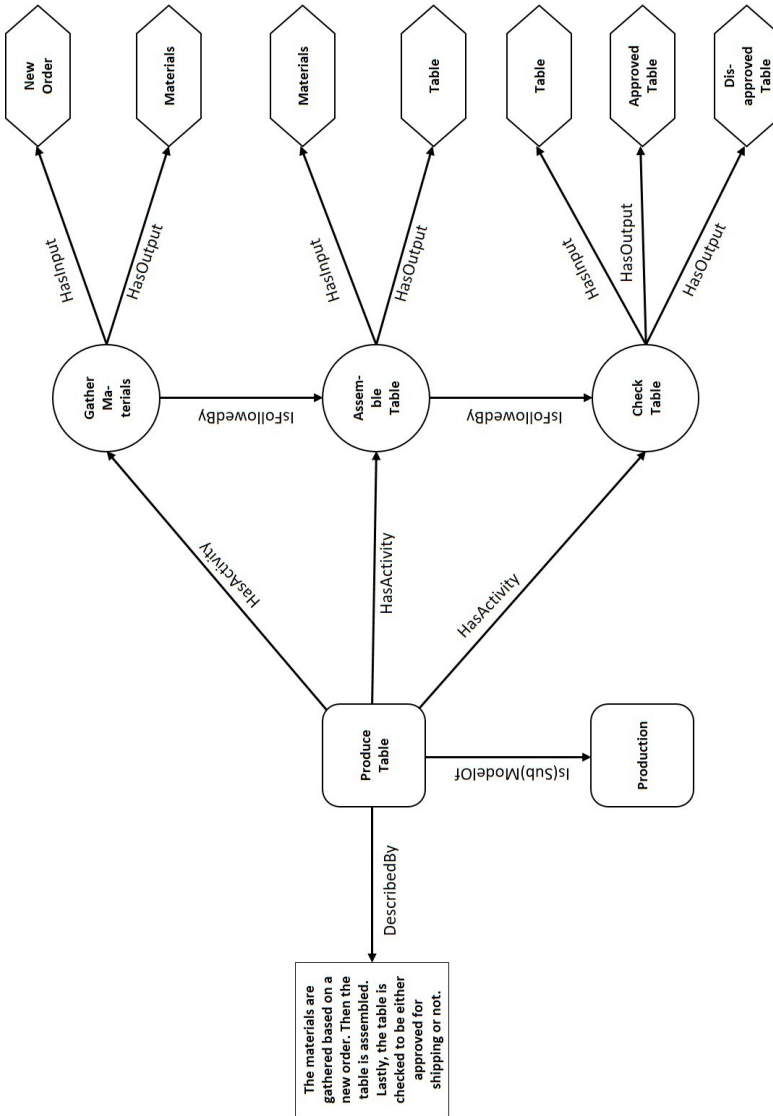


Figure 5.14: Second Instance of the Knowledge Graph shown in Figure 5.12.

graph and instances, are integrated into the model as far as possible. Third, elements that are not identified in the reference point, but are part of the investigated process mode have to be handled as well.

Adjusting the Existing Elements

The different relationships between the elements used in a Horus procedure model are analyzed with the help of a subset of the ontology, corresponding knowledge graphs, and their instances to adjust the model and the already included elements.

1. **The relationship between the Activity and Objects:** To identify patterns based on the relation between verbs as indicators for activities and their related objects as input and output, the relationship between activities and objects is analyzed. The connections between activities and inputs as well as activities and outputs detail for each activity the related objects required.

This assignment enables identifying an activity's input and its output from the relevant knowledge graph either if there is an object given and its purpose is not clearly stated or if no object is associated with the activity and has thus to be identified.

For instance, a model part that is created based on the simple description of *"The accountant prepares the invoice"* will be transformed into an activity *"Prepare Invoice"* with the input *"Invoice to prepare"* and output *"Invoice"*.

Under the assumption that one instance of the creation of an invoice was already modeled in previous projects and is

part of the knowledge base, insights about the model and the description at hand can be gained from this reference. Let the model in the knowledge base consist of the same activity and output but with the input *"Product List"*.

Now, the recently from text generated model can be adjusted based on this information. It can be derived that the input of *"Invoice to prepare"* as an implicitly generated input can be replaced with the input *"Product List"*.

2. **The relationship between Activities and Actors:** To identify patterns based on the relationships between roles and performed actions, an analysis of the relationship between nouns and verbs is performed. In the knowledge graph, the relation between activities and roles details for each role the activities performed. This way, the reference point can provide information about who performs which activity.

Since the knowledge graph is a collection of knowledge residing in the knowledge base in form of an ontology, multiple actors might be identified for an activity. In such a case and the current version of this approach, the actor and related activity from the adjusted model are not compared towards one possible solution. However, rather it is ensured that this pair is part of a set of pairs found in the knowledge graph. However, this leaves room for ambiguity, as not one answer is provided, and no statement is made about which element of the subset fits with the analyzed pair of actor and activity.

Considering the previously described example and the model consisting of an activity *"Prepare Invoice"* and corresponding

inputs and outputs. The reference to a related model and activity in the knowledge base can provide additional information about the acting entity. Hence and in this example, the model stored in the knowledge base has a role attached to the activity "*Prepare Invoice*". Consequently, the model at hand that is processed in the model adjustment step can now be extended by the role attached to the activity and thus enriched with additional information that can be acquired from information residing in the knowledge base.

3. **The relationship between Activities:** To identify the control flow and the interrelations between the single activities, the existing instances are analyzed and found structures and patterns are compared with the provided process model. To identify patterns in the relationships between activities, the relationship between activities in texts is investigated. The relation between activities can be extracted from the knowledge graph and specifies for each activity the related activities that are either performed directly beforehand or afterward, so that common patterns can be identified.

Let there be an extension to the previously described model. After the activity "*Prepare Invoice*" there is an activity "*Send Invoice*" described. Next to this, the knowledge base, in this case, includes a process model that describes a process on the sequentially connected activities "*Prepare Invoice*", "*Check Invoice*" and "*Send Invoice*". This knowledge base entry indicates an extension of the model that is adjusted.

The activity "*Check Invoice*" has now to be inserted in between the other two activities based on the corresponding entry from the knowledge base.

However, it still can be the case that the new model described a process where the activity "*Check Invoice*" does not happen. An adjustment of this type is thus to be treated with caution and requires a human decision to decide for or against the proposed adjustment.

Enrichment of the Existing Model by Additional Elements

The enrichment of the existing model and extension by additional elements is based on the analysis of the model's control flow and how the control flow is described in the knowledge graph and its instances. A comparison leads to the identification of possible missing flows and elements.

Based on the *FollowedBy*-relationships, activity sequences are compared, and missing elements are identified. Based on the similarity, the missing elements are then either added to the model or, in case of low similarity and the subsequent assumption that the created model is of a different kind than the information it is compared to, the missing elements are ignored.

Reduction of the Existing Model by removing Elements

In some cases, model elements might be removed from the created model. On the one side, this scenario can occur in the case of disconnected model parts and in the case of parts that are not relevant and do not belong to the process described. On the other side, elements might not be relevant to mention or should be mentioned as

a combination with other elements, such as, e.g., two activities that are in sequence, might be generally described as one activity.

Another scenario in which model elements might get removed includes the handling of duplicates. While creating the model manually or with this approach in a semi-automated way, differences in the labels or naming of activities, objects, and roles can lead to elements that have the same meaning but cannot be identified as such due to inconsistent descriptions.

Assuming an example in the form of the following description may not be considered a good or precise description in general, but can occur: *A new order is registered. The recent order is then processed.*

In the example, the new order and the recent order refer to the same object. This matching can be identified over synonym detection but can also be identified as two distinct objects, as word compositions increase the complexity to compare two terms towards their similarity. Nevertheless, identifying such overlap is important and should lead to removing one object or merging both objects, respectively.

Even though the acquired information about the process model's domain serves as a positive reference, it cannot be considered a single point of truth. In case a process model for a completely new process and sub-field inside the domain is created, no or few related information will be found in the reference point. Consequently, it would not be correct to discard the whole model since it is not mentioned in the existing knowledge base but instead has to be recognized as newly acquired knowledge and treated as such.

5.2.4 Transformation Approach

In this subsection, the transformation approach that is proposed in this work is described. The different steps of the approach are explained. Required inputs, used techniques, and expected outputs are stated. The related work introduced in Section 4.1, and the often-used keyword- or pattern-based strategies for the identification of textual structures and corresponding model elements are combined and adjusted for the purpose of this approach. The explanation is focused on the conceptual level, while the implementation and realization of the concept in the form of a proof of concept are described in Chapter 6. The transformation approach is organized in five parts, shown in Figure 5.7, that include different smaller steps that come with individual tasks to solve. The five parts are:

Text Pre-Processing

Before analyzing and transforming the textual input, some processing steps have to be performed to adjust the input in a suitable format for the next steps. While texts are often provided with some additional meta-data, such as headings or bullet points, this approach focuses solely on the plain text itself. Any structural elements should be removed beforehand, such as, e.g., headings or bullet points.

After this, the process continues with the plain text reassembling the input from Fig. 5.6 without any meta or structural information.

"While Marc prepares the invoice, Laura collects all the products. She then packs the products together with the invoice for delivery."

To enable the initial preparation, the additional required information is annotated to the text with the help of existing techniques, such as POS-tagging and syntax trees. This information will further help the upcoming steps of *Linguistic Feature Extraction* and *Model Element Mapping*.

Beginning with operations on the whole textual input, coreferences and anaphoras are replaced with the help of StanfordNLPs coreference annotator. With the annotation, referenced pronouns are identified, such as he, she, him, or her, and replace them with the entity they refer to. In our example, the "*She*" from the second sentence is replaced with the entity "*Laura*" it refers to.

*"While Marc prepares the invoice, Laura collects all the products. **Laura** then packs the products together with the invoice for delivery."*

In the next step, the text is split into sentences. The single sentences are then analyzed to identify the main and subordinate clauses in the respective sentences. Each sentence (S) and subordinate clauses (SBAR) with no further nested sentences or subordinate clauses are extracted. The syntax tree used for splitting the first sentence of our example is shown in Fig. 5.15.

Linguistic Feature Extraction

In the *Linguistic Feature Extraction* all the linguistic features required for the later steps of mapping and generating, such as keywords and grammatical structures, are extracted. These structures include:

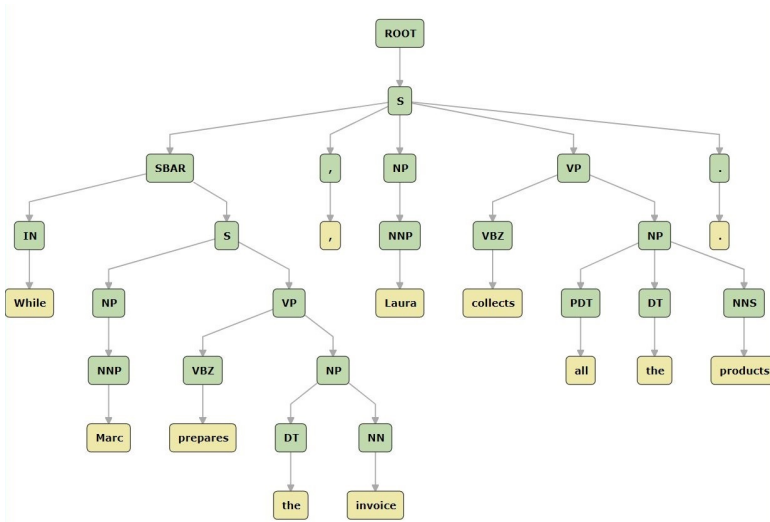


Figure 5.15: Syntax Tree of the Example Sentence.

Subject-Verb-Object Tuples: For every sentence and phrase the Subject, Verb, Object (SVO)-tuple is identified over the corresponding POS-tags and their dependencies. Subjects commonly represent the actor in a sentence and are seen as representations of the role attached to the related activities. Activities are identified over the verbs that describe the actions performed. The object or multiple objects of a sentence represent the object that is processed in or is the result of an activity. In case passive voice is used in the textual description, the subject of that sentence is taking on the role of the object in the corresponding model part. The subject in the tuple remains empty in this case. This is important for the later mapping of these tuples to the set of model elements.

Additionally, the identified verbs are categorized in pre-defined sets of "creating"-verbs and "processing"-verbs to define inputs and outputs with higher precision. This categorization is especially relevant if no further objects are defined over additional clauses in the relevant sentence(s) and if the main object of the core sentence is either an input or output of the described activity.

Figure 5.16 shows an example of the transformation of the sentence "*The table is assembled.*", including the "creating" verb "*assemble*", while Figure 5.17 shows an example based on the sentence "*We send the invoice.*", including the "processing" verb "*send*".

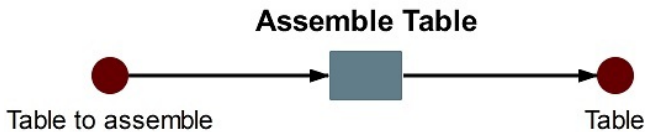


Figure 5.16: Inputs and Outputs based on a "creating" Verb.

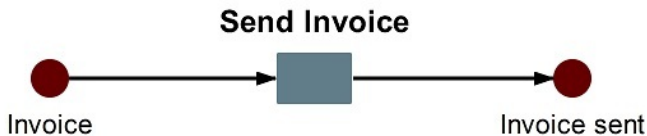


Figure 5.17: Inputs and Outputs based on a "processing" Verb.

The core sentence and subordinate clauses: Identification of the main sentence, e.g., as mentioned over a syntax tree and the constituency label "S" for sentence and the identification of subordinate clauses, e.g., indicated through the constituency label "SBAR".

Subordinate clauses are categorized differently in literature. At this stage, a sub-set of subordinate clauses is considered that contains the following clauses:

- *Relative clauses:* A subordinate clause that is dependent on another part of a sentence can be expressed through a relative clause. This clause offers information about the relationship between the different sentences and corresponding model parts, especially between activities and objects. The identification and interpretation of relative clauses are similar to the other clauses dependent on keywords. For instance, the words *that*, *which* and *who* hint to an activity or objects that are connected to the related main sentence and the there described model part. An example of a relative clause in a process description would be:

"An accountant approves a payment, that was received."

- *Adverbial clauses:* Adverbial clauses are a linguistic construct that holds information about the sequence of the described process parts. These clauses often add additional information about the time, place, cause, or purpose of the aspect they refer to. The different types of information inherent to the clause are classified with the help of a set of keywords, such as, e.g., the word *before* is used to describe the sequence or the word *because* is used to describe a condition and thus a possible split into two branches of the process. An example of an adverbial clause in a process description would be:

"Before the package is sent, it has to be checked for completion."

Instead of whole phrases and adverbial clauses, adverbial modifiers are single, individual words used to describe the control flow of the model parts. These modifiers are commonly used to indicate if a process step is performed before or after the previously in the description mentioned process. The keyword *before* mentioned before that can occur in adverbial clauses can in some cases also be used without a further phrase and thus is understood as a particular keyword without further information about additional elements relevant for the process. An example of an adverbial modifier in a process description would be:

"Afterwards, the logistic department delivers the package."

- *Conditional clauses:* One of the linguistic constructs used to identify elements and especially patterns of the described models are conditional clauses. They are grammatically part of the adverbial clauses and describe a condition but are investigated separately due to their descriptions of specific model patterns, namely exclusive choices. An example of a conditional clause in a process description would be:

"If all products are collected, the warehouse clerk prepares the package for the delivery."

Prepositional Phrases: These are not considered clauses, but are similar to verb- or noun-phrases indicated and labeled in the syntax tree. They often serve as indicators of additional information about objects in a model, such as the distinct description of inputs and outputs. They are thus relevant for the creation of activities'

objects. They are extracted over the syntax tree, and objects referenced in the phrase are combined with the referenced verb into a tuple. The SVO-tuples and PP-tuples are then further merged based on their syntactical dependencies.

Not all possible linguistic features get extracted, but only the ones that are in this approach defined to be relevant for the mapping to model elements. Expressions that are not relevant to the later process model are not extracted and investigated further. Such non-relevant information and features include, e.g., emotional analysis or topic modeling via NLP.

The feature list used here is not considered complete and within the highest degree of detail. Future extensions and further differentiation between model elements, patterns, and structures might require further and a more detailed analysis of the linguistic features. The features were selected based on this project's requirements for the described transformation from text into a Horus procedure model.

Model Element and Pattern Mapping

After extracting the linguistic features from the prepared textual input and providing them in a structured format, a mapping of the different textual parts and elements of the intended model must be established. The intended model to create is a specialized Petri net, a Horus procedure model. This type of model consists of objects, activities, roles or actors, gateways, e.g., XOR-Input, XOR-Output, and connections, or the so-called control flow.

To establish a connection between the linguistic features and named model elements and structures, rules define how each lin-

guistic aspect relates to its model counterpart. At this stage, a rule-based approach provides a relatively static solution. However, the lack of labeled data in the form of "process description - process model"-pairs makes the option to train, e.g., a model for classification, not feasible. Defining rules for the classification of linguistic features into the categories of the corresponding model elements and patterns allows an approach without the availability of training data.

In a possible future case, in which an extensive set of process models and their fitting descriptions are available, texts can be labeled by their reference to the model part they are representing. Based on these labeled sets of pairs, a model can be trained that would be able to automatically map parts of the text onto model elements and structures based on the existing set of labeled examples. Similar to the previous step, the mapping is performed sentence by sentence. Consequently, the elements extracted from the linguistic features of each sentence can be considered parts of a sub-model of the intended model described in the textual input.

Structures are generated based on activities and their attached elements, such as objects and roles. The result is structured in the format of *[Activity|Input|Output|Role]*, and a possible, here simplified, output would be, for instance, *[Create|Product List, Customer Data|Invoice|Accountant]*. Inputs and outputs are further distinguished between inclusive and exclusive objects. A single input or multiple inputs required together for the next activity are labeled with *AND*, while exclusive inputs and outputs are labeled with *OR*. The activity labels are generated as a combination of the verb's infinitive and the related object noun. The input and output labels are defined over the object name and depending on input or output

over a combination of object and verb, e.g., *object to create* or *created object*. Splits, such as exclusive choices or parallelism, are identified using a keyword- and pattern-based approach.

The result for our sample textual input after this step is shown in Table 5.1 and Table 5.2 on the example of two sentences that are variations of the earlier introduced running example.

In the following, structures found in a Horus procedure model and examples of these are shown as they are used in this approach.

Assuming that in a company products are sold, and sales orders are received as well, the processing of these sales orders has to be modeled. Figure 5.18 shows an example of a sequence of process steps that are likely to be performed in a sales process. The excerpt focuses on the structure of a sequence and does not represent the whole extend of a possible sales process.

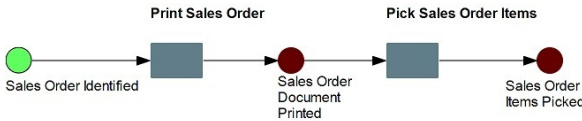


Figure 5.18: Example of a Sequence.

As part of a sales process, a newly received order is often processed, and relevant data for further steps are extracted. Figure 5.19 shows an example of parallel execution of different process steps in this excerpt of a sales process. The step *Processing Order* leads to the two outputs of *Customer Data* and *Product List* that initiate two new branches of process steps.

Figure 5.20 shows the exclusive decision of following one of two branches of process steps based on a decision that is part of the

Table 5.1: Example of identified Model Elements from the first Sentence.

Sentence
After a new order, the order is processed for customer data and the product list.
Linguistic Features
SVOs: (- process order)
PPs: (customer data, product list process out)
AdvCl: (new order process in)
Element(s)
(Process Order (New Order,AND) (Customer Data,AND),(Product List,AND)) -)

Table 5.2: Example of identified Model Elements from the second Sentence.

Sentence
From the customer data the accountant retrieves the invoice information.
Linguistic Features
SVOs: (accountant retrieve invoice information)
PPs: (customer data retrieve in)
Element(s)
(Retrieve Invoice Information (Customer Data,AND) (Invoice Information,AND) Accountant)

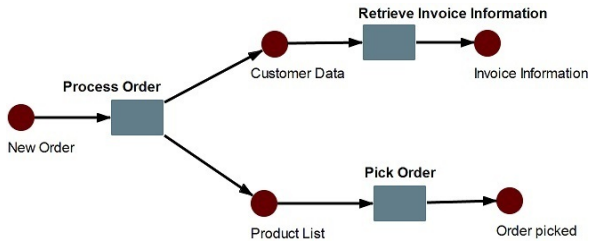


Figure 5.19: Example of a Parallel Split.

activity *Check Lead Activity*. As part of the initial example shown in Figure 5.6, a lead is received from an email, and it is checked for recent activities. Such activities are either identified or not.

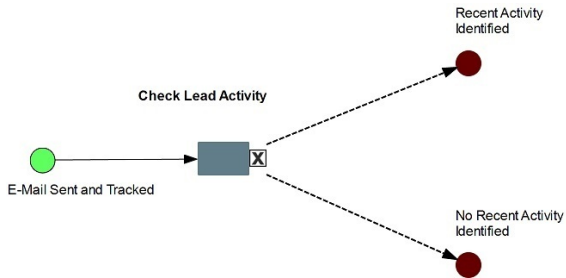


Figure 5.20: Example of an Exclusive Choice.

Figure 5.21 shows the synchronization of two previously initiated branches of process steps. Considering a process in a later stage of an order process, the two processes of *Pick Products* and *Create Invoice* both have to be completed to complete an order.

Figure 5.22 shows the merge of two previously initiated branches of process steps. In contrast to parallel execution and synchroniza-

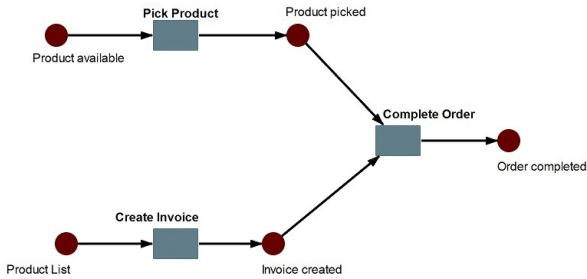


Figure 5.21: Example of a Synchronization.

tion, not both process branches have to be executed. In the shown excerpt, a lead can be approved either from an existing lead that is updated or based on a new lead created in the CRM system.

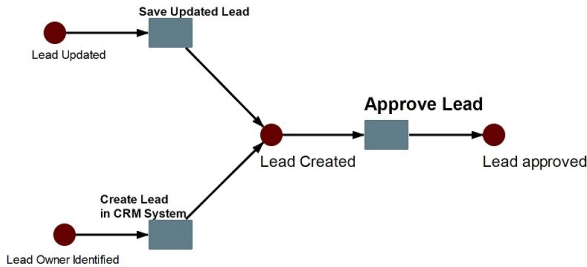


Figure 5.22: Example of a Simple Merge.

Another structure that is often mentioned when discussing patterns in process models is a loop. Even though they represent a unique structure in these models, they are not explicitly handled in this approach. Loops are established over the backtracking to a previous object identified based on the activities and their matching

inputs and outputs. However, the textual description of such loops includes additional challenges for the transformation as these often tend to present a certain ambiguity in their possible interpretation.

Model Generation

The result of the *Model Element Mapping* often includes disconnected model parts that have to be combined into a sound process model. The resulting data frame provides a set of model elements and structures for each processed sentence. These now have to be combined to generate a continuously connected process model. However, if a fitting process description is provided as an input, the result can consist of only connected model parts. In such a case, the step of connecting disconnected model parts becomes obsolete.

In case disconnected model parts are present, the "sub-process-models" of each sentence have to be combined. The combination is based on shared inputs and outputs, as the control flow is indicated over these matching inputs or outputs. Of these objects, the referenced ones were already identified in the step before

Referring to the example of a parallel split shown in Figure 5.19, the combination of extracted model elements from the two sentences that were shown in Table 5.1 and Table 5.2 as well as describing parts of the model excerpt are explained. After the mapping to model elements and structures, the two sentences can be combined over the common object *Customer Data* that is the output of the activity described in the first sentence and at the same time the input of the activity described in the second sentence. The resulting model data frame represents a Horus procedure model, as shown in Figure 5.23.

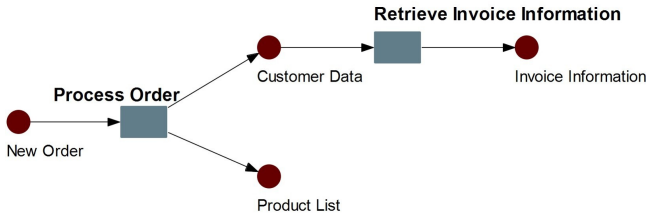


Figure 5.23: A Data Frame of a Horus Procedure Model.

One crucial challenge occurs when the generated model parts cannot be connected in the first try over the shared inputs and outputs and cannot be combined into a sound process model. This can also be the case when objects are similar but cannot be identified as the same. In such a case, disconnected models are generated, which can be assumed to be connected based on the fact that they are described in one textual description. If such cases would lead to detached model parts in the final model, a way to connect these has to be found. The first and most straightforward strategy to resolve this is to connect the model parts based on their occurrence sequence in the textual description.

Connecting disconnected model parts with different outputs and inputs based on their chronological order in the text leads to the loss of one output or input as just one is taken as a connecting element between the two sub-models. The question of which object to keep and which one to drop. In this case, a solution is in place that is based on a defined priority of objects. If an object is explicitly mentioned in the corresponding text part, it will be preferred over an implicitly from the related activity derived object. In case both objects are either explicitly mentioned or implicitly derived, the role

of the verb the activity is based on is of relevance. If the first activity is a creating activity, the output of this activity is prioritized over the input of the activity it connects to. However, if the second activity is a processing activity, this activity's input should be prioritized. This priority rule can lead to a conflict if the first activity is creating while the second activity is processing the individual objects. In such a case, either both objects are used for the connection, or a third rule comes into place, which is prioritized as last resolution strategy outputs over inputs.

Measurements to connect initially disconnected model parts might require further refinement in the future. Nevertheless, they already benefit from the contextual knowledge that is accessible through the reference point. Solving disconnections inherits the danger of losing information or adding wrong information by introducing connections that are not there and not correct. Thus, the problem is to be treated carefully due to the quickly rising complexity in more extensive texts and the described processes.

Figure 5.24 shows an example of the connection of two model parts and the resolution of conflicting objects based on the priority of explicitly stated objects over implicitly derived objects. In the example, the object *"Customer Data"* is explicitly mentioned in the textual description and is thus favored over the implicitly generated object *"New Customer registered"*.

Figure 5.25 shows an example of the connection of two model parts and the resolution of conflicting objects based on the type of action performed. In the example the activities *"Create Delivery"* and *"Create Shipment"* are both creating activities. In consequence the object *"Delivery Note"* is prioritized over the general object *"Shipment to create"*.

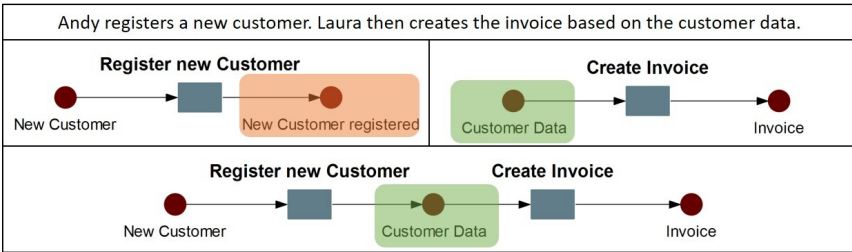


Figure 5.24: Example of a Conflict Resolution based on Object Occurrence.

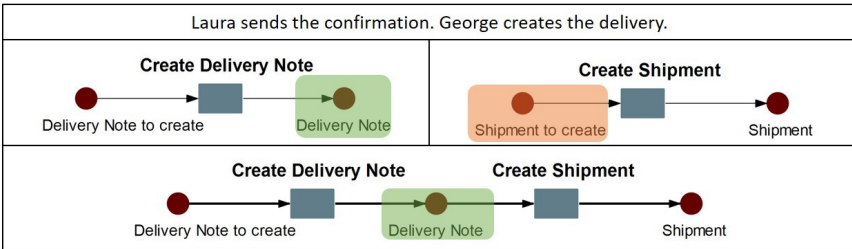


Figure 5.25: Example of a Conflict Resolution based on Activity Types.

Model Adjustment

After the model generation, a reference point is introduced in the form of an ontology-based knowledge graph consisting of multiple instances that are described in 5.2.3. Insights can be gained about the relationships between certain entities via the knowledge graph. In case an object is missing or cannot be identified as input or output, a reference to the knowledge graph can be helpful, where it can be looked up over the referenced activities and objects. If

no connections are found in the model generation step and model parts are disconnected, the reference point supports identifying possible connections. Extracted model elements identified based on the mapping of text, sentences, and words onto model elements are compared to the knowledge base entries. If an element is already found in the knowledge base or similar to one entry, it may be approved or adjusted according to the knowledge base.

The model adjustment includes a matching process between the process model, generated or manually created, and the reference point. However, process model matching, comparison, and alignment is an open issue with complex challenges that still await a satisfying solution. Therefore, the focus in this work does not lie in the realization of this comparison and alignment. The idea and theoretical concept are defined towards a possible application and expected benefits. Nevertheless, the possible adjustment of the model can be categorized into three categories:

1. **Resolution of Objects:** In some cases where the process description does not provide all detailed information about the described process, the contextual reference point can provide this information. For instance, a process step is described simply with "*The package is prepared for shipment.*". This description does not state any information about the input to the activity that is performed but solely desired output. Assuming this activity is described in more detail, including a distinct input, already in another previous model that is part of the knowledge base, additional information might be available over this reference. Regarding the activity residing in the knowledge base, the process model at hand can be

adjusted accordingly. While the original sentence would lead to a model that includes one activity, namely "*Prepare Package*", one output "*Shipment*" and an input "*Package to prepare*", the reference point might offer us the information that this input was already defined as "*Ordered Products*" in another model describing the same activity. With this information, the model can now be adjusted, and the generalized input can be replaced with the distinct input of "*Ordered Products*".

2. **Establishment of a connected Control Flow:** The control flow between and the connection of the individual steps of the process model are established over the connection of matching inputs and outputs in this approach. In case this connection cannot be established, and the model consists of disconnected model parts, the reference can again provide some helpful information based on previously created models it enables access to. For each activity, the reference point provides information about related activities based on sequences of activities found in previously created models.
3. **Extension by missing Activities:** Assessing not only model parts but the model as a whole with help of the reference point increases the complexity and computational effort required. It has to be acknowledged that model comparison, alignment, and compliance are topics that still state challenges that are not solved feasibly at this moment. Nevertheless, the inner structure and sequence of activities of a model is used and compared against similar models residing in the knowledge base.

In case missing structures are identified, these are based on a probability calculation added to the model or ignored. In this approach, the comparison and resulting adjustments are focused on single activities that might be integrated between two existing activities. The identification and integration of whole model structures, such as a process branch as part of a split into parallel or exclusive execution, inherits a complexity beyond this thesis.

Assuming a description of a process model that states two activities and their inputs and outputs correctly and sufficiently and let these two activities be "*Prepare Package*" and "*Send Shipment*". The description and the corresponding model may be correctly formulated and transformed but might miss process relevant aspects found in similar models found over the reference point. Imagining that in this case, multiple models already exist that describe such a process with an additional activity of "*Check Package*" that is performed between both earlier mentioned activities. In such a case, the reference point provides the information that a crucial element generally connected to the described two activities is missing and might have to be added. Consequently, the activity "*Check Package*" is inserted into the created model. However, the decision about such an extension and adjustment of the created model is highly dependent on the intention and purpose of the process. It should require human approval to be performed. Therefore, the adjustment would rather serve as a recommendation to the modeler than an automatic and invasive change to the model.

6 Implementation and Evaluation

In this chapter, the implementation and the performed evaluation of a proof of concept are explained. First, the foundations, including the scope, architecture and key components, are described. Second, the implemented features and their functionality are presented. The explanation of features is structured based on the individual transformation steps as they are described in the concept. Third, the evaluation and the results are discussed.

The implementation is used as proof of concept for the core features of the proposed transformation approach. As some tasks faced in this endeavor are already tackled in different existing solutions, the implementation's description focuses on the parts contributed by this thesis.

The different features that are explained the following, as well as the corresponding code and the data set of model-description pairs used for the evaluation can be found in a GitHub repository and accessed at https://github.com/felixrnlte/PhD_Submission.

6.1 Scope

The implementation is intended as a proof of concept that is independent of any other tool. Consequently, the intended Horus procedure models are not generated and displayed in the Horus Business Modeler but instead provided as data frames that include all the necessary information to create and import these.

The implementation is done in Python due to the availability of a broad spectrum of libraries and packages that provide existing solutions for the set of faced problems in the area of NLP. Additionally, specific solutions are not available in Python, such as the StanfordCoreNLP suite that is only implemented in Java. These solutions are integrated using the provided interfaces.

The implemented features focus on the transformation approach to generate a model from textual input. With the priority put on the model creation steps, the subsequent model adjustment based on the reference point is not part of the implementation. This decision is based on the fact that in-depth knowledge of different fields is required to achieve meaningful results. Practical solutions in both areas are scarce and the corresponding problems addressed are mostly considered highly complex.

For the visualization and better understanding of different created artifacts that hold information about the linguistic features of a text, the individual solutions were accessed and used over web services and demo environments. Such include the CoreNLP online version¹ and the SpaCy Dependency Visualizer².

¹ See <http://corenlp.run/>. Last accessed: 08.12.2020.

² See <https://explosion.ai/demos/displacy>. Last accessed: 08.12.2020.

6.2 Architecture

In this section, the architecture of the implementation building on the concept and the described transformation steps is explained. This explanation includes the general idea of a transformation pipeline and used artifacts.

6.2.1 Pipeline

The implementation follows the concept of a pipeline that incorporates features in a modular way and enables the later exchange of components. This approach can facilitate, e.g., the adjustment to another language. As for now, just one module for each task and step is established. For future extensions, e.g., an alternative approach to anaphora resolution or mapping to additional modeling languages, the modules for the specific affected tasks can be exchanged without impacting the other modules. This way, multiple pipelines can get invoked for different languages, models, or used techniques in the future.

The described, implemented and tested pipeline modules are provided in an overview in the following and later explained in further detail. Figure 6.1 shows the visualization of the current pipeline design.

The different modules of the pipeline are:

1. Anaphora Resolver as part of Pre-Processing. Linguistic Analysis is performed here already as this is required for the task. The gained information is annotated and used in the later modules.

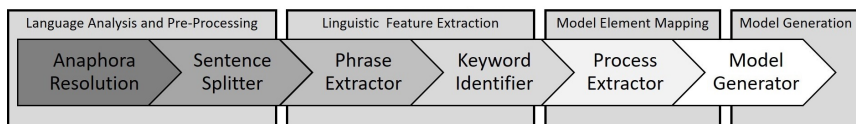


Figure 6.1: Implementation Pipeline.

2. Sentence Splitter as part of Pre-Processing.
3. Phrase Extractor as part of Linguistic Feature Extraction.
4. Keyword Identifier as part of Linguistic Feature Extraction.
5. Process Extractor as part of Model Element Mapping.
6. Model Generator as part of Model Generation.

The pipeline requires a java-based StanfordCoreNLP server with the current version 4.2.0 running in the background³.

6.2.2 Input Requirements

The expected input has to fulfill a set of requirements for our current approach to work properly. At this stage, the input has to be plain text without any structural features, such as headings or bullet points. The text has to be written in English and for the successful processing correctly formulated. Spelling or grammatical mistakes can lead to serious misinterpretation during text analysis. Specific text structures, such as activities included in a noun combination,

³ Download Link: <https://stanfordnlp.github.io/CoreNLP/download.html>. Last accessed: 08.12.2020.

e.g., *the creation of the table*, are not included so far. Additionally, the entire input is analyzed, and any additional information, such as *"Next I will describe(...)"*, will be processed too. In consequence, text parts not related to the model per se, such as *"The process start with (...)"* or *"The process ends here"*, would be identified but add noise to the final model. These statements would lead to identifying and creating model parts, such as an activity *"Start process"*. The text should thus focus purely on the process it intends to describe.

All information about the model elements and structures can be represented in either active or passive voice. Additionally, the set of possible subordinate clauses includes adverbial clauses of which conditional clauses are treated individually, and relative clauses. Subordinate clauses are bound to defined sets of keywords, which include words such as "for", "before", and "based on" at the moment.

6.2.3 Artifacts

Iterating through the different steps of the transformation process in the different modules, the process input, acquired information, extracted linguistic features, and the identified model elements and parts are carried by different artifacts throughout the process.

The input, provided as plain text, is processed as a string. Different libraries available for Python, such as Pandas⁴, enable the import of data from different data sources using single functions. The current implementation imports text from files either in .txt or .csv formats and allow the direct definition of an input string in the code itself.

⁴ See <https://pandas.pydata.org/>. Last accessed: 08.12.2020.

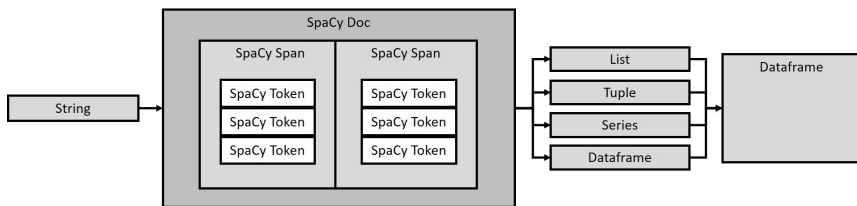


Figure 6.2: Object Types and Flow.

Additional information acquired in the *Information Enrichment* phase using mainly SpaCy is carried by the so-called *Tokens*. These are part of a SpaCy structure similar to a sentence, a *Span*. Furthermore, a *Span* is part of a higher organizational unit, the document *Doc*. Each of these three SpaCy objects includes information about linguistic characteristics and is used throughout all transformation steps.

Extracted linguistic features and model parts identified in the model element mapping are stored inside data frames handed over to the next processing step. Information used inside the individual functions is mostly carried by lists, tuples, and series that are in the end combined into a data frame.

The final result, the model generated, is then provided again in a data frame that depicts the individual model elements, such as activities and their associated objects and roles.

A rough depiction of the set of artifacts is shown in Figure 6.2.

6.3 Features

Throughout the transformation and adjustment steps, different selected features defined in the concept were implemented. These features are categorized into the five categories that were established earlier in the design and concept of this approach of which the first four are part of the transformation approach and implemented in the proof of concept:

1. Natural Language Text Pre-Processing
2. Linguistic Feature Extraction
3. Model Element and Part Mapping
4. Process Model Generation
5. Process Model Adjustment

The description and explanation of the features are supported by examples that emphasize each step's individual tasks.

6.3.1 Natural Language Text Pre-Processing

Pre-Processing features are used before the actual transformation steps to align the textual input with the next steps' input requirements. Furthermore, the textual input is enriched with relevant information, such as POS-tags.

Input Cleaning

Transform input into plain text and remove structural information, such as headings or bullet points. Unlike other approaches where NLP is used, this approach does not include a typical removal of stop words, as the standard corpus of these includes relevant words for our approach. However, a limited set of not required or wanted symbols can be used to check the text and remove any occurrence of them. A simple plain text is expected in the proof of concept imported from a .csv or .txt file or provided directly as a string in the code. Further integration of other inputs and more extensive steps of transforming and cleaning data into the required format is, in the end, a data integration problem that is itself independent of the transformation approach. In consequence, the individual data integration steps are not further investigated here.

Input Information Enrichment

The upcoming steps require further information than just the plain text to perform. Libraries, such as SpaCy, NLTK, and Stanford-CoreNLP offer a variety of methods to annotate additional information to the text. This approach focuses on SpaCy and the use of its tokens that include most of the required information already. However, NLTK and StanfordCoreNLP are used later on for specific information, such as, e.g., to create a specific syntax tree of a sentence. A java-based StanfordCoreNLP server has to be running simultaneously to enable access over an API.

The information annotated and provided at this stage is part of the *Tokenization of SpaCy* and include⁵:

- `TOKEN.POS` and `TOKEN.TAG` - Two different types of POS-tags.
- `TOKEN.DEP` - The syntactic dependency relation.
- `TOKEN.HEAD` - The syntactic parent of this token.
- `TOKEN.LEMMA` - Base form of the token, with no inflectional suffixes.
- `TOKEN.I` - The index of the token within the parent document.
- `TOKEN.SPAN` - The sentence span that the token is a part of.
- Different attributes that indicate a certain type of the token, such as, e.g., a digit over `TOKEN.IS_DIGIT` or an url over `TOKEN.LIKE_URL`

Resolving Coreferences and Anaphoras

As one step of processing the textual input previous to the transformation, the library `StanfordCoreNLP` and the included coreference resolution provided by the Stanford NLP Group⁶ is used. It enables the identification of coreferences in the text and the replacement with their first occurrence. This information will be important in the later processing, as roles are identified over subjects and replace pronouns with the related coreference. Thus, the approach does not identify "he" or "she" as a role, but rather the referenced entity.

⁵ See <https://spacy.io/api/token>. Last accessed: 08.12.2020.

⁶ See stanfordnlp.github.io/CoreNLP/coref.html. Last accessed: 08.12.2020.

```
1 def resolve(corenlp_output):
2     for coref in corenlp_output['corefs']:
3         mentions = corenlp_output['corefs'][coref]
4         antecedent = mentions[0]
5         for j in range(1, len(mentions)):
6             mention = mentions[j]
7             if mention['type'] == 'NOMINAL' and
8                 ↪ mention['text'] != antecedent['text']:
9                 antecedent = mention
10            if mention['type'] == 'PRONOMINAL':
11                target_sentence = mention['sentNum']
12                target_token = mention['startIndex'] - 1
13                corenlp_output['sentences']
[ target_sentence -
  ↪ 1 ]['tokens'][target_token]['word'] =
  ↪ antecedent['text']
```

Listing 6.1: Coreference Resolution.

Listing 6.1 shows a method that is part of the anaphora resolution using the coreference annotations. For each coreference identified over the coreference annotations (line 2), the first occurrence is taken as so-called "*Antecedent*", and this replaces each following occurrence if it has the type "*Pronominal*". If an occurrence has the type "*Nominal*", it is replacing the first occurrence as "*Antecedent*" and following coreferences are replaced with this one from that point on.

Splitting Sentences

In this approach, each sentence of an input text is processed individually. The input requires that the process's activities are described atomically and that each sentence generally focuses on one activity.

```
1 def _get_sentences_spacy(text):
2     sentences = []
3     nlp = spacy.load("en_core_web_sm")
4     doc = nlp(text)
5     for sent in doc.sents:
6         sentences.append(sent.text)
7     return sentences
```

Listing 6.2: Sentence Splitting.

Due to this, in this step, the textual input is split into sentences with the help of the document structure of SpaCy.

Another module is realized by the same function using the library Stanza by the Stanford NLP Group⁷. However, the approach using Stanza often provides not only splitting into sentences but also identifies phrases and subordinate clauses as sentences. Nevertheless, it can help the translation when considering more complex sentences in the future that, e.g., also describe more than one process step.

Listing 6.2 shows an excerpt of code used for the sentence splitting. First, the SpaCy pipeline with the specific English language model is loaded. Second and based on the SpaCy-specific object *document*, the individual sentences are extracted from the text and stored in a list. This extraction happens in order with the sequence of the sentences as it is present in the input. It is essential to ensure the order of occurrence for later steps.

⁷ See <https://stanfordnlp.github.io/stanza/>. Last accessed: 08.12.2020.

6.3.2 Linguistic Feature Extraction

In this subsection, features that enable the processing of the individual sentences are explained. First, subordinate clauses are extracted from each sentence. Each identified subordinate clause is removed from the original sentence, and the leftover sentence is used as the core or main sentence. Second, for each sentence and the included subordinate clauses, the relevant linguistic features are extracted.

Possible nested structures of subordinate clauses, including further clauses, are not considered in this approach due to, on the one side, computational capacity and feasible runtime, and on the other side due to assumed and expected characteristics of the textual inputs. A textual input consisting of multiple nested clauses is not considered realistic. These would result in sentences that extend the length and complexity found in the regular and everyday use of written natural language.

As the phrase extraction and further extraction of linguistic features are closely related, they are explained together in the corresponding parts for each sentence structure. However, in the implementation, these are split into three distinct modules: The Phrase Extractor, the Keyword Identifier, and the Model Element Mapping.

Phrases and Features are extracted and provided as a data frame in which columns represent the different sentence structures. An example of the header of the resulting data frame is shown in Table 6.1.

The extraction of linguistic features builds on the analysis of more detailed linguistic aspects, such as information about individual words. For this, functions to extract tuples consisting of the subject, verb, and objects are provided together with functions to analyze

Table 6.1: Output Data Frame Header of the Linguistic Feature Extraction.

Core	Adverbial	Conditional	Relative
------	-----------	-------------	----------

further phrases that hold relevant information, such as prepositional phrases. These supportive functions are briefly explained in the following before the focus is turned back onto the analysis of each sentence and included subordinate clauses.

Extraction of Subject, Verb and Objects Tuples

The first and most relevant function for later steps is the extraction of SVO-tuples from a sentence, excluding additional phrases⁸. Under the assumptions and requirements for this approach, only up to one subject and one verb are extracted from each sentence part, while there can be multiple objects. In the case of multiple objects, the conjunction between both is already carrying valuable information for the later model mapping. If the conjunction is an "or", this is already an indication for a later exclusive split. To avoid a loss of this information, objects are extended here with an attribute describing their conjunction. For this the values "AND" and "XOR" are possible. This attribute is attached to all objects identified in this main sentence or the subordinate clauses.

⁸ The actual implementation used in this work is based on <https://github.com/peter3125/enhanced-subject-verb-object-extraction>.


```

1  (...)
2  svos = []
3  verbs = [tok for tok in tokens if _is_non_aux_verb(tok)]
4  for v in verbs:
5      verb_phrase = ' '.join([item.text for item in
6          ↪ v.subtree])
7      verb_phrase = nlp(verb_phrase) # Tokenization with
8          ↪ Spacy
9      is_pas = _is_passive(verb_phrase)
10     subs, verbNegated = _get_all_subs(v)
11     if len(subs) > 0:
12         (...)
13         v2, objs = _get_all_objs(conjV, is_pas)
14         for sub in subs:
15             for obj in objs:
16                 objNegated = _is_negated(obj)
17                 if is_pas:
18                     svos.append((" ",
19                         "don't " + v.lemma_ if verbNegated or
20                         ↪ objNegated else v.lemma_,
21                         to_str(expand(sub, tokens,
22                             ↪ visited))))
23                 else:
24                     svos.append((
25                         to_str(expand(sub, tokens, visited)),
26                         "don't " + v.lemma_ if verbNegated or
27                         ↪ objNegated else v.lemma_,
28                         to_str(expand(obj, tokens,
29                             ↪ visited))))
30             else:
31                 (...)
32                 v2, objs = _get_all_objs(conjV, is_pas)
33                 for obj in objs:
34                     objNegated = _is_negated(obj)
35                     if is_pas:
36                         svos.append((" ",
37                             "don't " + v.lemma_ if verbNegated or
38                             ↪ objNegated else v.lemma_,
39                             ""))
40                     else:
41                         svos.append((" ",
42                             "don't " + v.lemma_ if verbNegated or
43                             ↪ objNegated else v.lemma_,
44                             to_str(expand(obj, tokens, visited))))
45                 (...)

```

Listing 6.3: Extraction of Subject, Verb and Objects.

Listing 6.3 shows an excerpt of the function `_GET_SVOS_SENT`⁹. First, all verbs are extracted that are not auxiliary verbs (line 3), such as in *"The invoice was sent"*. Auxiliary verbs are ignored, and the verb that relates to it is placed as the main verb of this sentence. For the following steps, each extracted verb is extended with its corresponding phrase (line 6). Each Verb Phrase (VP) is then processed further. A check for the use of passive or active voice in the sentence is performed (line 7). After this, all subjects related to the verb phrase are identified and extracted (line 8). If the list of subjects is not empty (line 9), all objects relating to the verb are extracted (line 11). The verb, related subjects, and objects are combined into the "SVO"-tuple (lines 15-24). In case a negation is identified (line 13), the verb is added with the additional term of *"don't"* to indicate the negation in the tuple. If the list of subjects is empty, the same steps are performed, but the subject in the tuple is left empty (lines 32,37).

For the identification of SVO of a sentence, the used voice type is relevant. In case a sentence is written in active voice, the extraction matches the annotations provided by the POS-tags and the dependency parser. However, if the sentence is written in passive voice, the provided information and annotations can be ambiguous. In a passive sentence, the object is labeled as the subject, and no subject in the sense of an actor is present. In this case, the (empty) subject and object in the SVO-tuple are switched (lines 18,23,33,38).

Table 6.2 shows an example of an extracted SVO-tuple from a sentence.

⁹ The described function can be found in the folder `HELPER` and in the file `SUBJECT_VERB_OBJECT_EXTRACT.PY`

Table 6.2: Extraction of SVO-tuples from a sentence.

Input Sentence: The accountant sends the invoice
Output - Subject, Verb, Object of the Sentence: [[('accountant', 'send', 'invoice')]]

Extraction of Prepositional Phrases

Similar to SVO-tuples, prepositional phrases are used to extract further information about the sentence or subordinate clause. They are extracted and used to identify objects that serve as input or output. To distinguish between input and output, a keyword-based approach is defined. While a certain set of words or word combinations indicate an input, such as *"with"*, *"from"*, *"based on"* and *"out of"*, another set indicates outputs, such as *"for"*, *"towards"* or *"into"*. Objects of prepositional phrases that cannot be categorized into one of the two categories are then checked for information about actors over *"by"* and *"through"*. These categories are defined in the code in sets of words and word combinations indicating the corresponding type of prepositional phrase.

Listing 6.4 shows an excerpt of the function `_GET_PREP_SENT`¹⁰ used to extract the prepositional phrases from the textual input. The example shows how tokens of a sentence are checked for their syntactic dependency (lines 2,3). In case a dependency with the label *"prep"* (*preposition*) is found, the sub-tree of this token is extracted as the linguistic structure of the prepositional phrase (line 4).

¹⁰ The described function can be found in the `LANGUAGEOPERATIONS.PY` in the folder `HELPER`.

```

1  (...)
2  for token in tmp:
3      if token.head.dep_ == "prep" and
   ↪ token.head.text.lower() != "of":
4      pp = ' '.join([item.text for item in
   ↪ token.head.subtree])
5      for toki in nlp(pp):
6          if toki.text.lower() == "or":
7              role = "XOR"
8              reference = _get_head_verb(token.head)
9              tmp_pps.append([pp, reference])
10 (...)

```

Listing 6.4: Extraction of Linguistic Features from a Prepositional Phrases.

In case multiple objects are identified, their relationship has to be clarified. In this regard, the whole prepositional phrase is checked for conjunctions (lines 5,6). If the conjunction "or" is found, the label "XOR" is assigned to clarify that the objects are exclusive. Otherwise, the standard connection "AND", defined previously to this code excerpt, is kept. An additional condition is checked, which is shown in Listing 6.5. To decide about the mentioned type of a clause, the first token of that clause is compared (lines 5-11) to the indication sets (lines 1-3). It is labeled according to the set it belongs to, and thus one of the types of input, output, or role is assigned.

Table 6.3 shows an example of the extraction of relevant information from a prepositional phrase. The preposition "from" indicates the prepositional phrase and its role as input. The object "product list" is then stored together with the verb "create" it relates, its role as "input" and the connection or gateway as "AND". In case of just one input or output, the gateway is always provided as "AND".

```
1 INPUT_IND = {'from', 'with', 'on'}
2 OUTPUT_IND = {'as', 'into', 'for', 'towards', 'over'}
3 ROLE_IND = {'by', 'through'}
4
5 for token in pp_nlp:
6     if token.text.lower() in INPUT_IND:
7         type = "input"
8     if token.text.lower() in OUTPUT_IND:
9         type = "output"
10    if token.text.lower() in ROLE_IND:
11        type = "role"
```

Listing 6.5: Extraction of Objectroles from a Prepositional Phrases.

Table 6.3: Extraction of PP-tuples from a Sentence.

Input Sentence: The invoice is created <u>from the product list</u> .
Output - Prepositional Phrase Features: [['create', 'product list', 'input', 'AND']]

Extraction of Adverbial Clauses and their Linguistic Features

The information about the described process residing in the adverbial clauses provides insights mainly about process steps' control flow.

Adverbial clauses are extracted either over the corresponding sub-tree of the syntax tree of the sentence or a set of keywords, including words such as "*before*", "*after*" or "*while*". Identifying the set of keywords is necessary since the used tools and models to create the syntax tree do not provide a sufficient level of precision

in identifying adverbial clauses. This precision can vary between the different available language models. However, a significant percentage of misinterpretations regarding adverbial clauses was observed in tests conducted the implementation.

However, regardless of the approach used to extract the clause, it and the set of keywords support the identification of patterns such as parallelism and sequence. The relevant keyword is checked for its referenced verb to identify the related activity in the core sentence. With pre-defined roles of keywords, the relation between the adverbial clause's activity and the core sentence can be identified in the later step of the **Model Element Mapping**. Mentioned roles of keywords include, e.g., the indication of a previous process step over words, such as "*after*", or a subsequent process step over words, such as "*before*". Table 6.4 shows an example sentence, the extracted adverbial clause and its linguistic features.

The identification and extraction of adverbial clauses and conditional clauses as sub-category of these, are shown in Listing 6.6. With the help of the dependency annotation of SpaCy, the dependency "*advcl*" or the token dependency label "*prep*" together with the POS-tag "*ADP*" identify the adverbial clause (line 2), while the keyword "*if*" then further provides identification of the conditional clause (line 5). The tokens sub-tree is then joined into a combined tree (line 3). Lastly, the clause is removed from the initial sentence (line 9).

Listing 6.7 shows the function `_GET_ADVCL_FEATURES`¹¹. As features between adverbial clauses and included conditional clauses

¹¹ The described function can be found in the `INFORMATION_EXTRACTION.PY` in the folder `LINGUISTIC_FEATURE_EXTRACTION`

```
1 for token in nlp_sent:
2     if token.dep_ == "advcl" or (token.dep_ == "prep" and
   ↪ token.pos_ == "ADP" and token.text.lower() in
   ↪ ADVINDS):
3         advcl_tmp = ' '.join([item.text for item in
   ↪ token.subtree])
4         advcl_nlp = nlp(advcl_tmp)
5         if advcl_nlp[0].text.lower() == "if":
6             condcl.append(advcl_tmp)
7         else:
8             advcl.append(advcl_tmp)
9         sent_tmp = sent_tmp.replace(advcl_tmp, "")
```

Listing 6.6: Extraction of Adverbial and Conditional Clauses.

differ, they are processed individually. First, the SVO-tuple is extracted (line 5). Second, prepositional phrases are identified and extracted (line 6). Third, the role of the adverbial clause is determined (line 7). Lastly, all three attributes are combined into one list that is returned (lines 8,9).

```
1 def _get_advcl_features(df_row):
2     info = []
3     if df_row['Adverbial']:
4         for x in df_row['Adverbial']:
5             svos = _get_svos_sent(x)
6             pps = _get_prep_sent(x)
7             role = _get_adv_role(x)
8             info = [svos, pps, role]
9     return info
```

Listing 6.7: Extraction of Linguistic Feature from an Adverbial Clauses.

Table 6.4: Extraction of Linguistic Features from an Adverbial Clauses.

Input Sentence: Before Laura sends the invoice for the notice, Marc approves the invoice.
Output - Adverbial Clause Features: [[('Laura', 'send', 'invoice')], [['send', 'notice', 'output', 'AND']], 'successor']

In some cases, just a single adverb, an adverbial modifier, is stated without using a whole clause. In this case, the adverb will remain as part of the core sentence as the carried information directly relate to it and is processed accordingly. However, the adverb references no longer to activities or objects stated in a clause but refers to the previously mentioned process. Thus, the previously mentioned process step outputs are stated as inputs for the currently investigated process step to establish the connection between both.

Extraction of Conditional Clauses and their Linguistic Features

Even though part of adverbial clauses, conditional clauses are treated individually in this approach. They are an indicator of exclusive splits and represent a distinct structure in the intended process model. Over the keyword, *it*, exclusive splits related to a previous process step are identified in the corresponding clauses. With the help of the POS-tags and the annotation of constituency dependencies, the keywords are identified, and the related clause in the form of the syntax tree sub-tree are extracted. The extracted phrase is

processed further and checked for objects that have to be added to the activity from the core sentence.

A conditional clause often describes a condition for the current sentence's activity but refers to a previous sentence.

```
1 def _get_cond_features(df_row):
2     info = []
3     if df_row['Conditional']:
4         for x in df_row['Conditional']:
5             refsvos = _get_svos_sent
6                 ↪ (df_row['Core'])[0])[0]
7             ref = refsvos[1] + " " + refsvos[2]
8             obj = _get_objects(x)
9             info = [ref,obj]
```

Listing 6.8: Extraction of Linguistic Features from a Conditional Clauses.

Listing 6.8 includes the function to extract the linguistic features from an identified conditional clause. First, the SVO-tuple of the core sentence is extracted to establish the reference to the activity of the sentence (lines 5,6). Second, all objects from the conditional clause are extracted (line 7). Third, reference and object(s) are combined into a list and returned (lines 8,9).

In Table 6.5 an example sentence including a conditional clause based on the word "if" is shown. In the example, the keyword "if" relates to the verb "send". The relation between both is identified by looking at the tokenized version of "if": `token.head()`. The verb "send" is then later used to map to the activity of SEND INVOICE. The conditional clause "If the check is positive" is then checked for verbs. If no verbs are present, it is assumed that only objects are

Table 6.5: Extraction of Linguistic Features from a Conditional Clauses.

Input Sentence: If the check is positive, the accountant sends the invoice.
Output - Conditional Clause Features: ['before', 'send invoice', 'positive check', 0]

described over the clause. In the example, an object is extracted from the clause.

Extraction of Relative Clauses and their Linguistic Features

Relative clauses are another part of the textual input that is extracted and used for further analysis. Relative clauses contain elements which interpretation is provided by an antecedent element on which the clause depends on. Keywords that indicate a relative clause include terms such as *that*, *who*, *which* and *whose*. The clauses are identified over the set of keywords. Like the conditional and adverbial clauses, the keyword's sub-tree is used to extract the relative clause. Linguistic features are then identified in the clause, and information relevant for the transformation is returned.

```

1 for token in nlp_sent:
2     if token.dep_ == "relcl":
3         relcl_tmp = ' '.join([item.text for item in
4                               ↪ token.subtree])
5         relcl.append(relcl_tmp)
6         sent_tmp = sent_tmp.replace(relcl_tmp, "")

```

Listing 6.9: Extraction of a Relative Clause.

An excerpt of the function `_GET_PHRASES_OF_SENTENCE`¹² is shown in Listing 6.9. First, every token in a sentence is checked for its dependency earlier annotated through the SpaCy dependency parser (line 1). If a token with the dependency *"relcl"*, which is indicating a relative clause, is found (line 2), the sub-tree with the root of this token is extracted. The labels of all tokens of this sub-tree are combined again into the relative clause's textual representation (line 3). The extracted clause is then appended to a list (line 4), and the clause is removed from the main sentence (line 5).

The function to extract the linguistic feature from a relative clause is shown in Listing 6.10. As the relative clause references a previous or current object, it is assumed that the information also refers to this object. This is defined in a role with the label *"before"* (line 5). However, in future extensions, the used tense of a relative clause must be incorporated to identify references to upcoming objects. Even though the description of the information about objects that are not yet mentioned is rather uncommon in process descriptions, they might occur and be relevant when extending the use case. Next, the core sentence's verb is extracted and used as an indicator for the referenced activity (line 6). The same is done for the object of the core sentence (line 8). Besides, the SVO-tuple of the relative clause and the tuple of the prepositional phrase are acquired (lines 7,9). Lastly, all acquired information is combined and returned as a list (lines 10,11).

In Table 6.6 an example sentence including a relative clause based on the word *"that"* is shown. In the example, the term *"before"*

¹² The described function can be found in the `PHRASE_EXTRACTION.PY` in the folder `LINGUISTIC_FEATURE_EXTRACTION`.

```

1  def _get_relcl_features(df_row):
2      info = []
3      if df_row['Relative']:
4          for x in df_row['Relative']:
5              role = "before"
6              ref_v = _get_svos_sent
7                  ↪ (df_row['Core'])[0])[0][1]
8              svo = _get_svos_sent(x)
9              ref_o = _get_svos_sent
10                 ↪ (df_row['Core'])[0])[0][2]
11                 pps = _get_prep_sent(x)
12                 info = [role, ref_v, svo[0][1], ref_o, pps]
13     return info

```

Listing 6.10: Extraction of Linguistic Features from a Relative Clause.

Table 6.6: Extraction of Linguistic Features from a Relative Clause.

<p>Input Sentence: Laura sends the invoice <u>that</u> was approved.</p>
<p>Output - Relative Clause Features: ['before', [(('Laura', 'send', 'invoice')), 'prepare', 'invoice', ([], [])]]</p>

indicates the role and that the clause relates to an object previously mentioned in the process. The verb *"send"* is later used to map to the activity of SEND INVOICE from the core sentence. The keyword *"that"* relates to the object *"invoice"*. The word *"approve"* represents the activity that was performed with the referenced object *"invoice"*. In case the relative clause includes any prepositional phrases, these are stated at the end of the list. As this example does not include any prepositional phrases, an empty list is added.

```
1 def _get_core_features(df_row):
2     info = []
3     if df_row['Core']:
4         for x in df_row['Core']:
5             svos = _get_svos_sent(x)
6             pps = _get_prep_sent(x)
7             advmod = _get_advmod_sent(x)
8             info = [svos, pps, advmod]
9     return info
```

Listing 6.11: Extraction of Linguistic Features from the Core Sentence.

Extraction of the Core Sentence and its Linguistic Features

The core sentence is considered the initial sentence, excluding all identified subordinate clauses. Thus, direct extraction of a text string is not performed. However, the extraction of features is performed similarly to the procedure performed for the subordinate clauses.

Listing 6.11 shows the function `_GET_CORE_FEATURES` that extracts all linguistic features from the leftover core sentence. First, the SVO-tuple is extracted (line 4), followed by objects from any prepositional phrases (line 5). The sentence's sequential role is extracted then by either an existing adverbial modifier or without such a modifier, considered according to the sentence's occurrence in the text (line 7). The sequential role can thus be either "*successor*" or "*predecessor*". Lastly, the gathered tuples are combined into a list and returned (lines 8,9).

Table 6.7 shows the result of extracting linguistic features from the core sentence. First, the SVO-tuples is shown. As no entity is present to replace the pronoun, the subject is stated as "*We*", the

Table 6.7: Extraction of Linguistic Features from the Core Sentence.

Input Sentence: We send the invoice for the payment.
Output - Core Sentence Features: [[('We', 'send', 'invoice')], [['send', 'payment', 'output', 'AND']], 'successor']

original subject of the sentence. Second, one object is defined in a prepositional phrase. The object "*payment*" is an output of the to the verb "*send*" related activity. Third, no statement about the sequence is given in the form, e.g., an adverbial modifier. Thus, the sequential order of process steps is assumed and indicated by the label "*successor*". Consequently, this sentence represents a process step that happens after the process step is extracted from the previous sentence.

6.3.3 Model Element Mapping

After the previous extraction of linguistic features, these features have to be mapped onto the set of model elements used in a Horus procedure model. The model elements used in a Horus procedure model are:

- **Activities:** In process models and thus in Horus procedure models, activities represent the performed action in the corresponding step and the input's specific processing.

Table 6.8: Output Data Frame Header of the Element Mapping.

Activity	Inputs	Outputs	Role
----------	--------	---------	------

- **Objects:** Same as for other process models, objects represent the artifacts given to an activity as input and created by these as output.
- **Roles:** Actions include optional information about the entity performing them. This information is acquired by attaching a role to the activity that shows the acting entity.
- **Control Flow:** The control flow connects the single elements and indicates the flow of objects throughout the process.
- **Gateways:** An indication of exclusive splits and merges is realized through attachments to activities. Activities are extended by a box representing either an exclusive input or exclusive output.

The linguistic features are mapped to these individual elements but already combined into a process model part representing a sub-process including the activity, required inputs and outputs, as well as attached role. This way, single elements are already combined into model parts and do not have to be stored with all their relations to other elements and then combined later. This combination should reduce the error-rate and loss of precision by avoiding an additional step of matching and merging artifacts. The resulting output data frame is structured as shown in Table 6.8.

Different sub-tasks are performed in the individual mapping steps of which some include the labeling of the generated elements. The nature of labeling activities, objects, and roles in this approach is explained briefly and can then be found later in the different mapping steps.

Generation of Activities

Activities are identified and based on the main verb of a sentence. However, the label of such an activity is composed of more than just the performed action. An activity is commonly described as a combination of the performed action and the object processed by this action.

The labeling of activities in process modeling usually follows one of two standards. The first standard defines an activity's label as a combination of a verb followed by the object, such as "*Deliver package*". In contrast, the second standard reverses this sequence. It uses a label consisting of a noun combination that includes the object and the noun related to the verb, such as "*Package Delivery*". In this approach, the first standard is used, as the verb is already present. The transformation between verb and noun presents an additional possibility for ambiguity and thus a loss of information.

Another aspect to consider is the handling of noun compounds. The identification of those is not trivial and had to be defined specifically in the implementation. For instance, the object *list of suppliers* would be identified in the SVO-tuples just as "*list*", while "*of suppliers*" would generally be recognized as a prepositional phrase.

However, the word "*of*" was explicitly stated as excluding criteria in the identification of prepositional phrases. Consequently, these have to be identified and combined again to refer to the real object

stated in the description. In the rare case that no object is present in the core sentence, the activity is generated with just the verb used as a label.

Generation of Inputs and Outputs

The same design choice for labels of activities has to be made for objects. As objects are not always defined precisely but rather are labeled using a combination of the performed action and the object, such a combination can be done in two ways again. On the one side, the object can be labeled over the object followed by the inflected verb, e.g., "*Invoice sent*", or the object follows the inflected verb, e.g., "*Sent Invoice*".

The design decision made here favored the second option since the business partner prefers it, and the majority of the acquired models for the evaluation follow this design. In consequence, this enables the use of models created by the business partner for testing.

An additional constraint present in the creation of inputs and outputs and their corresponding labels is given by the type of the verb. In case the verb is considered a *processing verb*, the output at hand is a combination of the performed action and the object. The verb describing the activity is inflected and combined with the object text. For instance, the activity *Send Invoice* would lead to the output "*Sent Invoice*". If the verb is considered a *creative verb*, the object at hand is seen as the output and is labeled with the original object text, e.g., based on an activity "*Create shipment*" the output is "*Shipment*".

Generation of Roles

As part of the extracted SVO-tuples, the subject of each active sentence is considered the acting entity in the described process step. Next to the SVO-tuples, actors can be described in prepositional phrases using keywords, such as *by*.

In the context of Horus procedure models, these identified actors are annotated to the activities as so-called roles. The subject from the SVO-tuple or the stated noun from the prepositional phrase is taken and associated with the activity according to the shared verb they relate to.

Support Functions

Next to previously introduced rules or constraints, several functions are defined that directly aim at solving specific sub-tasks, such as transforming objects used in the main functions. These include, for instance:

- In some cases, especially when a sequential order cannot be assumed, the activity to which an extracted subordinate clause relates has to be identified. To accomplish this and ensure a correct mapping, later on, the verb representing that activity and which a clause or word relates to has to be identified. As this relation in the form of the next syntactic parent of this token is not always pointing to a verb and thus activity, a recursive function checks for the next higher syntactic parent until a verb is found that expresses an activity. For this the function `_GET_REFERENCED_VERB` is implemented.

This step is essential for the later mapping and ensuring possible future extensions by, e.g., non-sequential descriptions of

processes. It is performed during the extraction of linguistic features. Each clause includes a verb representing an activity taken from a core sentence that it relates to.

- Some cases require the transformation of plural to singular nouns. In case the plural of a noun is used, this has to be transformed into the singular form for further use in the earlier mentioned combination into activity and object labels. This transformation is performed using the package INFLECT¹³, which offers different functions for the conversion of words, e.g., also the conversion of digits into written numbers.
- In several tasks, the conjugation of verbs in creating labels has to be performed. The package MLCONJUG3¹⁴ provides different approaches to conjugate verbs in different languages, including English. The conjugation is especially relevant when creating beforehand described labels for generated objects that consist of the objects name and the conjugated verb, such as "*Product selected*". As in these cases, just the activity is stated in the text, the verb this activity originates from has to be conjugated to fit the object label's requirements.
- The acquisition of synonyms and antonyms of words is provided by another function. For the list of synonyms and antonyms of words, the lexical database and Python library of WordNet¹⁵ is used. Antonyms are required when processing conditional sentences where just one possible outcome in the

¹³ See <https://pypi.org/project/inflect/>. Last accessed: 08.12.2020.

¹⁴ See <https://pypi.org/project/mlconjug3/>. Last accessed: 08.12.2020.

¹⁵ See <https://wordnet.princeton.edu/>. Last accessed: 08.12.2020.

form of an object is stated, and the opposing object has to be created.

Listing 6.12 shows the function to get the antonym of a word. First, a check is performed to clarify if the token is negated in the sentence already (line 3). If that is the case, the token itself is returned as an antonym (line 24). If the token is not negated and in case an antonym is required for a word that is an adjective, WordNet is used to find an antonym of this adjective (lines 8-13). In case antonyms are found, just the first suggested one is selected and returned as an antonym to provide a single result and not a set of possible results (lines 12,25). If the word is a verb or no antonyms are found in the previous case, the original word is used in combination with the negation "*not*" (lines 17-21).

Additionally, a set of functions is used that enables the reverse approach, such as finding the phrase a token belongs to or finding all verbs and objects a particular token refers to. These are mainly similar to "*getters*" in standard programming and are not further explained here as they do not directly contribute to the transformation approach.

The stated methods and the underlying rules are used to create the model parts from the model elements identified based on the linguistics features.

Mapping of the Core Sentence

The extracted features of the core sentence, a SVO-tuple, object-tuples of optional prepositional phrases, and the position in the

```
1 def _get_antonym(token):
2     antos = []
3     is_neg = False
4     for left in token.lefts:
5         if left.dep_ == "neg":
6             is_neg = True
7     if is_neg == False:
8         if token.pos_ == "ADJ" or token.pos_ == "NOUN":
9             for syn in wordnet.synsets(token.text):
10                for l in syn.lemmas():
11                    if l.antonyms():
12                        antos.append (l.antonyms()[0]
13                                     ↪ .name())
14                antos = list(set(antos))
15            if token.pos_ == "VERB":
16                to_append = "not " + token.text
17                antos .append(to_append)
18            if len(antonyms) == 0:
19                to_append = "not " + token.text
20                antos .append(to_append)
21        else:
22            antos .append(token.text)
23    return antos [0]
```

Listing 6.12: Acquisition of the Antonym of a Word.

order of process step, are mapped onto the set of model elements. A core sentence can include information about all relevant elements, namely activities, objects, and roles. From the SVO-tuple, the verb in combination with the object is mapped onto the activity, while either the object or the objects defined in prepositional phrases are taken as inputs and outputs. The type of the verb, "*processing*" or "*creating*" is considered when generating inputs and outputs when no further objects are defined. The subject of the SVO-tuples or the role defined in the prepositional phrase is mapped onto the role or acting entity of the process.

Listing 6.13 shows an excerpt of the function `_MAP_CORE`¹⁶. It is used to map the extract linguistic features of a core sentence onto the set of model elements. First, the information gained from the SVO-tuple is assigned. The verb as the main activity, the object as the main object, and the role itself (line 2-4). Second, the verb type is identified for the activity by comparing the verb with a set of defined verbs (lines 6-8). The activity itself is stored together with the type label in a list for further processing (line 10). Third, as the main object can consist of multiple objects, e.g., connected by a conjunction, these have to be identified and split into individual objects (lines 11-13). If the main object consists of just a single word, this is kept (lines 14-15).

After processing the SVO-tuple, possible tuples extracted from prepositional phrases have to be processed. For these, the object role determines if the stated object is added to the list of inputs, outputs, or roles (lines 17-24). If no inputs or outputs were defined by prepositional phrases, the object from the SVO-tuple is taken as input or output, based on the type of the verb (lines 25-31). Depending on the type of the verb and if the object is used as input or output, the counterpart has to be generated (lines 27,33-36). In the end, a list consisting of the activity, the inputs, and outputs, and the role is returned (lines 37-38).

Table 6.9 shows the output of the linguistic feature mapping onto model elements for the core sentence. From the verb *"send"* and the object *"invoice"*, the activity *"send invoice"* is generated, which is labeled based on the set of indication words as *"processing"*. As no

¹⁶ The described function can be found in the `MAPPING.PY` in the folder `MODEL_ELEMENT_MAPPING`.

```
1 svo, pps = tmp[0][0], tmp[1]
2 role = svo[0]
3 main_activity = svo[1]
4 main_object = svo[2]
5 (...)
6 vtype = "processing"
7 if main_activity in CREAT_VERB:
8     vtype = "creating"
9
10 activity = [activity_label, vtype]
11 main_objects, inputs, outputs = [], [], []
12 if len(main_object.split()) > 1:
13     (...)
14 if not main_objects:
15     (...)
16
17 if pps:
18     (...)
19     if pp[2] == "input":
20         inputs.append([pp[1], pp[3], 'defined'])
21     elif pp[2] == "output":
22         outputs.append([pp[1], pp[3], 'defined'])
23     elif pp[2] == "role":
24         role = pp[2]
25 if not inputs:
26     if activity[1] == 'creating':
27         input = obj[0] + " to " + main_activity
28         inputs.append([input, obj[1], 'generated'])
29     else:
30         input = obj[0]
31         inputs.append([input, obj[1], 'defined'])
32 if not outputs:
33     if activity[1] == 'creating':
34         (...)
35     else:
36         (...)
37 mapping = (activity, inputs, outputs, role)
38 return mapping
```

Listing 6.13: Mapping the Core Sentence onto Model Elements.

Table 6.9: Mapping the Core Sentence onto Model Elements.

Input Features: [[('accountant', 'send', 'invoice'), [], 'successor'], [], []]
Output - Elements: (['send invoice', 'processing'], [['invoice', 'AND', 'defined']], [['sent invoice', 'AND', 'generated']], 'accountant')

other prepositional clause was identified, the inputs and outputs have to be generated based on the SVO-tuple. Hence, because the activity is based on a processing verb, the input *"invoice"* is generated. Since there is only one object, there cannot be an exclusive choice, and the label *"AND"* is assigned to the input. Furthermore, the label *"defined"* is added to the input additionally, as the input is stated directly in the text. The output is generated as *"sent invoice"* and gets consequently the label *"generated"*. Lastly, the subject *"accountant"* of the SVO-tuples is assigned as the role.

Mapping of Conditional Clauses

Conditional clauses include the description of inputs of the currently investigated activity and at the same time insights about the outputs of the referenced activity. The stated input object of the current activity results from a decision made in a previous activity. Thus it can be assumed that this object is one of multiple possible outputs of this previous activity. The mapping process extracts different attributes and elements from the corresponding row and column in the linguistic feature data frame. First, the type of the condition that determines if the condition refers to an activity *"before"* or *"after"*, is determined. Second, the referenced activity is

generated based on the same SVO-tuple and in the same way as it was done to map the core sentence. Third, based on the stated object, at least two conditions, and thus two objects are defined. One object is based on the stated one, while the other represents the possible counterpart and uses the negated object, the antonym of the object. Lastly, the index of the row the referenced activity is residing in is stored for the later combination of all mapping steps.

Listing 6.14 provides an excerpt of the function `_MAP_COND`¹⁷. As more than one conditional clause can occur in sequence, a counter is defined previous to the code excerpt to track the initial activity. Then, the SVO-tuples of the referenced core sentence is taken as the referenced activity (line 2). The index of the referenced activity the conditional clause relates to the regarding outputs is extracted from the features tuple (line 3). Following this step, the adjective or negation of the object is selected as the adjective to find an antonym for, while the first word after the adjective is selected as the noun as this resembled the typical structure of, e.g., "*positive check*" or "*declined payment*" (line 4-5). Then the two possible options, the known object label as "*syn*" and the counterpart as "*ant*" are defined and generated (lines 6-7). The known object is then stored as input to the current activity and together with its counterpart as exclusive outputs for the referenced activity (lines 9-10). The type, inputs, outputs, and the index to the referenced activity are combined into a list and returned (line 11-12).

Table 6.10 shows an example of an input in the form of a row from the feature data frame and the generated output of model

¹⁷ The described function can be found in the `MAPPING.PY` in the folder `MODEL_ELEMENT_MAPPING`.

```

1  (...)
2  refactorvitiy_in = tmp[0]
3  xor_ref = tmp[3]
4  adj = nlp(tmp[1])[0]
5  noun = nlp(tmp[1])[1]
6  syn = adj.text + " " + noun.text
7  ant = _get_antonym(adj) + " " + noun.text
8  type = "before"
9  input = [syn, 'AND', 'defined']
10 outputs = [refactorvitiy_in, [syn, 'XOR', 'defined'],
11            ↪ [ant, 'XOR', 'generated']]
12 elements = [type, input, outputs, xor_ref]
13 return elements

```

Listing 6.14: Mapping a Conditional Clause onto Model Elements.

Table 6.10: Mapping a Conditional Clause onto Model Elements.

Input Features: ['before', 'send invoice', 'positive check', 0]
Output - Elements: ['before', ['send invoice', ['positive check', 'AND', 'defined']], [['positive check', 'XOR', 'defined'], ['negative check', 'XOR', 'generated']], 0]

elements. First, the type of occurrence is stated with *"before"*. The object *"positive check"* is described as input for the activity *"send invoice"* as well as combined as outputs together with the object *"negative check"*. To relate the outputs to an activity, the index of the referenced activity is attached.

Mapping of Adverbial Clauses

Adverbial clauses provide information about the sequence and connection between two activities, the one stated in the adverbial clause and the one stated in the core sentence. Thus, the mapping for the adverbial clause is primarily similar to the mapping performed with the core sentence. However, the information about the sequence of these two is then established over adjusting the inputs and outputs, respectively.

Listing 6.15 shows a code excerpt that deals with the mapping of linguistic features extracted from adverbial clauses onto the model elements and parts. The interrelation between adverbial clauses and other clauses or the core sentence adds complexity to this mapping.

First, all adverbial clauses are selected related to the verb, activity, respectively, of the activity analyzed. This is again, similar to the other clauses, achieved over matching the common verb in the SVO-tuples of both. Second, depending on the label, the adverbial clause was assigned based on the included keyword. It is declared as input or output. This assignment relates to adverbial clauses that solely describe objects. In contrast, the ones describing activities are assigned the labels of *"before"* and *"after"* to indicate the sequence and if the activity is placed before or after the related main activity. The different inputs and outputs are later then matched and merged, respectively.

Listing 6.15 shows an excerpt of the function `_MAP_ADVCL`¹⁸. The adverbial clause features are treated like features from a core sentence and mapped accordingly (line 4). The results are returned

¹⁸ The described function can be found in the `MAPPING.PY` in the folder `MODEL_ELEMENT_MAPPING`.

```

1 def _map_advcl(row):
2     advfeat = row[1]
3     if advfeat:
4         tmp = _map_core(row)
5         mapping = (advfeat[2], tmp[0], tmp[1], tmp[2], tmp[3])
6         return mapping
7     else:
8         return None

```

Listing 6.15: Mapping a Adverbial Clause onto Model Elements.

Table 6.11: Mapping a Adverbial Clause onto Model Elements.

<p>Input Features: [[('Laura', 'send', 'invoice')], [['send', 'notice', 'output', 'AND']], 'successor']</p>
<p>Output - Elements: ['successor', ['send invoice', 'processing'], [['invoice', 'AND', 'defined']], [['sent invoice', 'AND', 'generated']], 'Laura']</p>

together with the indicator for the sequence and returned as a list (lines 5,6).

Table 6.11 shows the output of the previous linguistic feature extraction of an adverbial clause as input for this step and the output of this step. Similar to the core feature mapping, the SVO-tuple and tuples of objects from the prepositional phrases are processed. The resulting activity, inputs, outputs, and roles are then combined with the adverbial modifier type. The adverbial modifier type in the example is "successor" and indicated that the activity mentioned in the adverbial clause happens after the activity from the core sentence.

Mapping of Relative Clauses

Relative clauses mainly state information about objects, roles, and the control flow in a model. Information about activities can be present as well, but are relatively uncommon and often an indicator for unstructured and complex process descriptions.

The extracted linguistic features from a relative clause thus refer to objects or roles from the core sentence. They can have a SVO-tuple and state an activity connected to one of the mentioned objects or roles from the core sentence. In case no SVO-tuple is part of the clause, it most likely contains attributes of the referenced object or role. This object is further mapped onto the adjusted inputs, outputs, or roles of the activity the referenced object belongs to.

Listing 6.16 shows the function to map relative clauses, their linguistic features, and the model elements and parts¹⁹. First, the referenced activity is combined from the SVO-tuple (line 4). The object is then created based on the previous activity by combining the activity's inflected verb with the object label (lines 4-8). The referenced activity, the new object with the labels "*defined*" and "*AND*" are stored in a list and returned (lines 9-10).

In Table 6.12, the output of the mapping is shown based on the previously extracted linguistic features. The first entry, the activity the object serves as input for, is generated based on the second entry of the input, the `glsacr:svo-tuple`. The newly generated object is stated, which is based on the third entry of the input, the referenced previous activity.

¹⁹ The described function can be found in the `MAPPING.PY` in the folder `MODEL_ELEMENT_MAPPING`.

```

1  def _map_relcl(row):
2      tmp = row[3]
3      if tmp:
4          refractivity = tmp[1] + " " + tmp[3]
5          obj = default_conjugator.conjugate(tmp[2])
6              ↪ .conjug_info['indicative'] ['indicative
7              ↪ present perfect']['1s']
8          + " "
9          + tmp[3]
10         elements = [refractivity, [obj, "defined", "AND"]]
11         return elements
12     else:
13         return None

```

Listing 6.16: Mapping a Relative Clause onto Model Elements.**Table 6.12:** Mapping of a Relative Clause onto Model Elements.

Input Features: ['before', [(('Laura', 'send', 'invoice')), 'prepare', 'invoice', ([], [])]]
Output - Elements: ['send invoice', ['prepared invoice', 'defined', 'AND']]

Combination of Mapping Results

In the last step of mapping, the resulting features of the individual sentences and clauses are merged into process parts already.

Directly described and stated model elements are always prioritized over objects generated from the implicit information gained from the core sentence and the corresponding activity. In case objects are described in subordinate clauses, these are prioritized over the general object taken from the core sentence. Furthermore, activities extracted from adverbial clauses are inserted as additional rows in the data frame.

Listing 6.17 shows the function `MODEL_ELEMENT_MAPPING`²⁰. The combination is first, and for each row, mapping the individual sentences and clauses using the previously described functions (line 3). All elements from the core sentence are initially used as activity, inputs, outputs, and role (line 5).

In the next step, a check for elements from conditional clauses is performed (line 6). In a part not shown in this excerpt, the clause's input replaces the input taken from the core sentence (line 8). If elements were extracted from this clause, these are used to adjust the inputs and outputs accordingly. In case the last row that was processed already included a conditional clause, the first output defined in the current clause is added to the outputs of the referenced row (lines 9-11). This way, the referenced row's outputs are the stated optional object from the conditional clauses. If no conditional clause precedes the current one, both outputs, the stated one and the created counterparts, are used as outputs of the referenced activity (line 12). After this, elements identified from relative clauses are processed. In case the relative clause referenced activity matches the activity from the core sentence, the relative clause's object is used as input for the activity (lines 14-16). The set of elements is then added as a row to the model data frame (line 17).

Lastly, elements from adverbial clauses are integrated. A variable is defined that holds the activity described in such a clause (line 18). Then and if an adverbial clause is present, the type of the clause is checked (lines 21,24). In case the type is "*precessor*", the activity and corresponding elements are directly added as a row to the model

²⁰ The described function can be found in the `MAPPING.PY` in the folder `MODEL_ELEMENT_MAPPING`.

data frame (lines 22-23). If the label is *"successor"*, the activity and elements are stored in the earlier defined variable (lines 25-26). Not shown in this excerpt is the final processing of the activity extracted from the core sentence that is, corresponding to the type of the adverbial clause, either inserted before or after the activity described in the adverbial clause, or in case no adverbial clause is present, inserted as row into the data frame.

The output is then represented in a data frame, as shown in Table 6.8. Each row represents one activity and the corresponding inputs, outputs, and roles.

6.3.4 Process Model Generation

The generation of a connected and complete process model is the goal of the **Process Model Generation** step. The main tasks revolve around connecting the extracted model parts with the help of the identified control flow and resolving any conflicts that arise due to mismatches or missing information.

First, all extracted elements that are present in individual lists or data frames have to be combined. This is performed using the function `_CONNECT_PART(MODEL_DF)`, shown in Listing 6.19. This function merges all extracted linguistic features that relate to each other. The merging is based on the matching verb or activity. Thus, all features belonging to a certain activity are combined into one data frame row to process each activity one by one in the next steps. The Listing and shown excerpt is further explained later on.

Different scenarios are possible when it comes to connecting the individual model elements extracted from each sentence. First, these can be connected over matching inputs and outputs. Second,

these can be connected based on the defined priorities introduced in the concept.

```
1  (...)
2  for index, row in feature_df.iterrows():
3      core, advcl, cond, relcl = _map_core(row),
4      ↪ _map_advcl(row), _map_cond(row), _map_relcl(row)
5      (...)
6      activity, inputs, outputs, role = core[0], core[1],
7      ↪ core[2], core[3]
8      if cond:
9          for con in cond: (...)
10             if condindi > 0:
11                 temp_out = mapping_df.at[ref, "Outputs"]
12                 mapping_df.at[ref, "Outputs"] =
13                 ↪ [temp_out[0], ref_outputs[1]]
14             else:
15                 mapping_df.at[ref, "Outputs"] =
16                 ↪ ref_outputs[1:]
17         if relcl:
18             for rel in relcl:
19                 if rel[0] == activity[0]:
20                     inputs = [rel[1]]
21         mapping_df = mapping_df.append({'ID': id, 'Activity' :
22         ↪ activity, 'Inputs': inputs, 'Outputs': outputs,
23         ↪ 'Role': role}, ignore_index=True)
24         (...)
25         if advcl:
26             (...)
27             if advcl[0] == "predecessor":
28                 mapping_df = mapping_df.append("AVERBIAL
29                 ↪ ROW")
30                 inputs = adv_outputs
31             if advcl[0] == "successor" and core:
32                 act_suc = [id+1, adv_activity, outputs,
33                 ↪ adv_outputs, adv_role]
34             (...)
35     return mapping_df
```

Listing 6.17: Combination of Mapping Results into Model Parts.

In the first case, no further adjustments have to be made. In the second case, however, disconnected parts have to be identified. This identification is performed by checking for each input and output of an activity if it can also be found in another activity. If just the input or output is connected to another activity, it is assumed that this activity is either a starting or ending activity in the process. If no object of an activity is found in any other activity, it is considered disconnected. The chronological predecessors and successors are identified for the disconnected model parts based on their occurrence in the text.

Listing 6.18 shows a function that checks for a row in a model data frame if it is disconnected. All Inputs and Outputs of the other rows are combined into a list (lines 3-8). These inputs and outputs of the row that is checked are combined into another set (lines 2,10-17). If an entry in the list of the row, an entry in the list *checkset*, is found in the list of the other rows, *rowset*, the row is connected and returns the value *False* as it is checked for disconnection (line 20). In the opposite case and if an entry is found in the other list, the row is connected, and the function returns *True* (line 22). However, the function is solely checking for disconnected rows but does not provide any information about whether a connection is correct.

In case disconnected rows are identified, these have to be connected to the rest of the model. As previously mentioned, different rules and priorities are defined to solve any conflicts. Listing 6.19 shows a function that, after the check for disconnected rows (line 4), provides a straightforward approach to connecting these rows with another model part. Considering the third priority, a disconnected part gets connected to the previous model part by replacing the input with the process step's output before (lines 8,9). This replace-

```
1 def _check_disconnect_row(model_df, drow):
2     checkset = []
3     rowset = []
4     rowindex = drow['ID']
5     for x in drow['Inputs']:
6         rowset.append(x[0])
7     for x in drow['Outputs']:
8         rowset.append(x[0])
9     ind = 0
10    for index, row in model_df.iterrows():
11        if ind < len(model_df):
12            if ind != rowindex:
13                for x in row['Inputs']:
14                    checkset.append(x[0])
15                for x in row['Outputs']:
16                    checkset.append(x[0])
17            ind = ind + 1
18    out = any(check in rowset for check in checkset)
19    if out:
20        disconnected = False
21    else:
22        disconnected = True
23
24    return disconnected
```

Listing 6.18: Checking a Generated Model for Disconnected Parts.

ment is done with the assumption that the process is described sequentially, and thus the model data frame is created accordingly. However, if the row that has to be connected is the first row in the data frame, it can be assumed that it is the first process step in the represented process model. As such, there is no previous output that can be replacing the current input. Thus, the output gets replaced by the input stated in the next row of the data frame (lines 6,7).

```
1 def _connect_parts(model_df):
2     df = model_df
3     if len(model_df) > 1:
4         for index, row in df.iterrows():
5             if _check_disconnect_row(model_df, row):
6                 if index == 0:
7                     df.at[index, 'Outputs'] =
8                         ↪ df.at[index+1, 'Inputs']
9                 else:
10                    df.at[index, 'Inputs'] =
11                        ↪ df.at[index-1, 'Outputs']
12
13     return df
```

Listing 6.19: Connecting Model Parts.

6.4 Evaluation

To evaluate the current state of the proof of concept, tests were conducted. These tests are described in the following, and a preliminary evaluation of the approach based on a pool of 40 cases is discussed. All cases are taken from real-world projects and represent processes from practice. The textual descriptions were formulated with the earlier stated input requirements in mind.

The evaluation focuses on the correct extraction of the core model elements and structures mentioned in the input text. Roles are not part of the set of evaluated model elements at this stage, as these, compared to the other elements, provide rather additional information less essential to create a correct process model. The bottom line of evaluating the approach focuses on creating a sound and correct process model without losing quality compared to manual creation. Based on this viewpoint, different questions are investigated by the evaluation:

- Can the approach identify all implicitly and explicitly in the text mentioned model elements and parts?
- Can these parts be connected correctly and combined into the overall process model that is described?
- How do wrongly identified model parts impact the established transformation process?

Considering these question, tests were conducted as described in the following. The results are then discussed in Section 6.4.3.

6.4.1 Experimental Setup

The experiment was conducted with students from bachelor and master program of information systems to obtain model-description pairs. The students were provided with each a set of process models taken from the project partner's knowledge base or other previously conducted projects and a set of process descriptions. The students formulated the textual process description for the provided models and created Horus procedure models from the provided textual descriptions.

A small set of input requirements was provided and no templates or sentence building blocks to guide the participants. On the one side, further restrictions were not as well known and clear as they are now after the evaluation. On the other side, the small set of requirements should provide an impression of how the approach performs in a "free" and less restricted environment. The participants were familiar with the modeling method and tool, as these are

part of different lectures in both programs. However, they cannot be considered experts that model processes in this manner regularly.

Feedback was given regarding the results to solve critical issues that would exclude a pair of model and textual description from the evaluation, such as violations of the modeling language's underlying rules. Each textual input was checked for grammatical correctness and spelling mistakes, as the transformation approach, especially the used language models, are sensitive to the incorrect use of the language. Additionally, all text structures that were specifically excluded were removed. This includes the use of meta-descriptions, such as "*the process starts with*" or "*the process ends here*".

Data and Measurements

For the evaluation, the original and manually created model is compared to the model data frame produced as the output of the *Model Generation* phase. It has to be acknowledged that the connecting of model parts can include drawbacks, as objects are sometimes falsely dropped for the sake of connectivity even though they are required and stated in the description. Thus, a comparison with the model data frame after the *Model Mapping* phase might hold insights about possible information loss or gain.

A set of numbers and measurements is used to assess the transformation approach towards the three mentioned questions. These include different descriptive numerical indicators of the process description and the corresponding created model, as well as calculated measurements.

In total, 40 models and their descriptions were tested. These included 765 model elements, consisting of 162 activities, 234 objects, and 396 connections. The average model consists of around 19 elements: 4 activities, 6 objects, and 9 connections. The smallest considered model consists of 9 elements, while the biggest model consists of 39 elements.

The collected example process models are taken from different projects and were created by different people in different contexts. In consequence, inherit some inconsistencies. These inconsistencies are considered in the evaluation to ensure a meaningful analysis even though the example text-pairs are not entirely fitting.

Furthermore, the definition of the measurements is essential for the intended evaluation. In this regard, the following paragraph briefly states what is considered correct, incorrect, and missing on the example of a process description, a manually created model, and an automatically generated model.

Considering the textual process description "*We approve the invoice. Then, we send the package for the delivery.*" and the original model Figure 6.3, as well as the automatically generated model shown in Figure 6.4, the selection of correct, missing, and wrong elements, is explained.

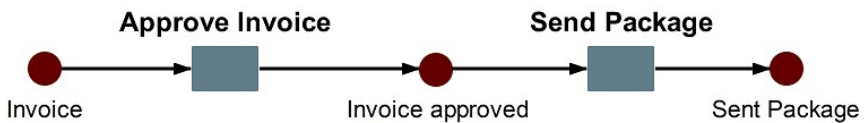


Figure 6.3: Example Test Model for Evaluation.

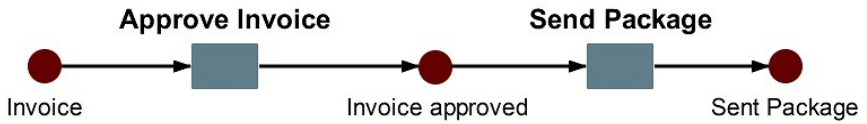


Figure 6.4: Example Generated Model for Evaluation.

The original and manually created model shows some inconsistency in the modeling style, for instance, in labeling the object *"invoice approved"*. The automated transformation process creates this object label as *"approved Invoice"* in the second model due to the defined style of labeling model elements. Nevertheless, it can be, without a doubt, be assumed that both labels describe the same entity. In such a case, the generated element *"approved Invoice"* is considered correctly generated.

Furthermore, the connection between *"Send package"* and *"sent Package"* would initially be identified as incorrect, as the connection between them is not found in the generated model. However, the control flow resembles one of the manually created models. Thus, the connections are considered correctly identified, while the object is considered incorrectly identified.

At the end of the shown process, the object should correctly be *"Delivery"*. As this is not the case in the generated model, where the object is labeled as *"sent Package"*, the object *"Delivery"* is missing, while the object *"sent Package"* is considered incorrectly generated. In this case, the object *"sent Package"* logically fits the process but is not correctly labeled. Thus, the object is considered incorrectly generated, but the object *"Delivery"* is not considered missing. The

same rule applies to the labels of activities. For instance, an activity "send the invoice" would be considered the same as "send invoice".

6.4.2 Summary of Results

The test results of the tests run with the implementation are summarized in the following. To be able to make statements about the accuracy of the transformation, the evaluation metrics *Precision* and *Recall* were used. These ratios are used for all model elements, as well as activities, objects, and connections. An overview of the results is shown in Table 6.13.

$$\text{Recall} = \frac{\text{Correct Elements}}{\text{Correct Elements} + \text{Missing Elements}} \quad (6.1)$$

$$\text{Precision} = \frac{\text{Correct Elements}}{\text{Correct Elements} + \text{Incorrect Elements}} \quad (6.2)$$

Among the total number of 40 models, 679 out of 744 elements were correctly identified with a precision of 89,3% and recall of 96,3%. An overview of the results is shown in Table 6.13. Additionally, two subsets of the model were investigated, representing small and more extensive models.

- Among the 19 models that consist of 18 or more elements, the identification of elements was achieved with a precision of 90.4% and recall of 96.1%.
- Among the 21 models that consist of 17 or fewer elements, the identification of elements was achieved with a precision of 90.4% and precision of 97.7%.

Table 6.13: Recall and Precision of the evaluated Models.

Metric	Value
Total Number of Models	40
Total Number of Elements	765
Total Number of Elements (excl. Connections)	396
Total Number of Activities	162
Total Number of Objects	234
Total Number of Connections	369
Average Number of Elements per Model	19.1
Average Number of Elements per Model (excl. Connections)	9.9
Average Number of Activities per Model	4.1
Average Number of Objects per Model	5.9
Average Number of Connections per Model	9.2
Total Number of correct Elements	679
Total Number of correct Activities	141
Total Number of correct Objects	194
Total Number of correct Connections	344
Total Number of incorrect Elements	81
Total Number of incorrect Activities	17
Total Number of incorrect Objects	46
Total Number of incorrect Connections	18
Total Number of missing Elements	26
Total Number of missing Activities	2
Total Number of missing Objects	11
Total Number of missing Connections	13
Total Recall	0.963
Total Precision	0.893
Total Recall excl. Connections	0.963
Total Precision excl. Connections	0.842

Example Results

Two example outputs of the transformation approach are shown in the following of which one represents a depicts the successful transformation. In contrast, the second example points out a current weakness of the approach.

This first example revolves around a process and includes a loop as a special pattern and construct. The input text and the generated model are shown in Fig. 6.5. The described process steps are connected via common input and outputs. While described independently in the textual description, the activities *Check Order* and *Open Order* can be connected with the help of the matching inputs and outputs. The common objects *Order* of the two activities *Check Order* and *Open Order* are then leading to the loop shown in the figure.

The second example is shown in Fig. 6.6 and emphasizes one of the difficulties this approach currently faces, namely the correct identification of exclusive splits. Exclusive splits are identified in the text via conditional clauses and the keyword "if" and the reference to the activity these conditions refer to leaves room for misinterpretation. This room for misinterpretation is especially the case when two exclusive splits follow each other. In the textual input, conditional clauses are identified three times by the "If..." structures. As all of these are stated without the interruption of a sentence without a conditional clause, they are all related to the first sentence. Thus, the third conditional statement is incorrectly matched to the first sentence.

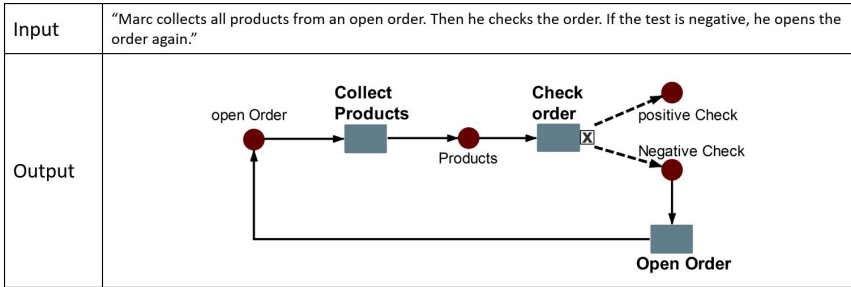


Figure 6.5: Petri Net Generation including a Loop.

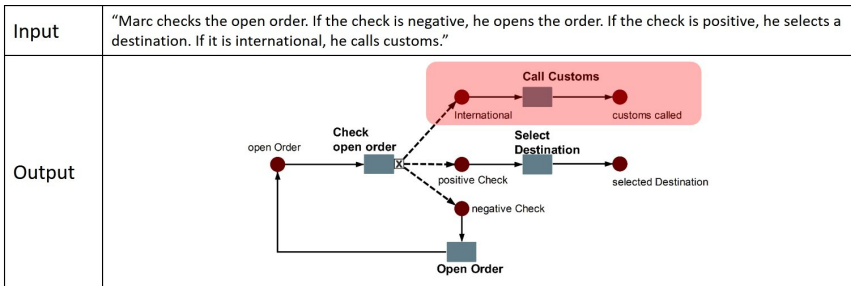


Figure 6.6: Incorrect Identification of an Exclusive Split.

6.4.3 Discussion

In this section, the results of the tests are discussed. In the first part, the results are summarized and assessed towards the three earlier defined questions and the implementation’s initial objectives. In the second part, remarks on the current state of the implementation are made.

The correct identification of elements, especially activities, shows a good recall and precision for all models. Contrary to the initial

assumption that more extensive models are more likely to consist of wrong elements or miss in the text described elements, the transformation approach provides a satisfying recall and precision for both small and big models. However, investigating the individual models' results shed light on the translation difficulties and where misinterpretations occur. Significant challenges and a higher error-rate are present in models that incorporate splits, specifically exclusive splits. The connection between two activities among sentences and stated conditions is complex and leaves room for ambiguous descriptions. However, in more extensive and complex models that include different patterns, such as splits and merges, the elements of these are often identified correctly but then incorrectly connected. Consequently, this leads to the wrong connection of parts and inevitable to a loss of information. An example of such a misinterpretation due to ambiguity is shown in Figure 6.6.

Connecting the different model parts and establishing a correct control flow thus present additional challenges, and the current state of the approach leaves room for improvement. The focus lies in creating a wholly connected model and comes at the price of wrong connections or losing information from unidentified elements.

Furthermore, while the content is often identified and transformed semantically correct, it shows some smaller differences. For example, in a text, the *list of suppliers* is mentioned, which is modeled in the process model as the object *suppliers*, our approach identifies the *list of suppliers* as an object in the model. It does not reduce it to just *suppliers*. Based on the explanation of correct, incorrect, and missing elements, such differences are considered acceptable as both labels logically represent the same object.

Another difficulty relates to the language of the text input mentioned before. Every language provides challenges to the precise identification of words and their grammatical dependencies. For example, the English word "*ship*" is identified by the most prominent and available models as a noun instead of a verb, which has implications on the further processing steps. The word's implicated use as either a noun or verb cannot be identified correctly by the used techniques. Consequently, the models generated from textual input, including such ambiguous words, can be inaccurate and require human interaction to confirm or adjust the generated model.

Considering the application of this approach and the intended benefits for process modelers, it has to be acknowledged that the mentioned difficulties in the analysis of the chosen language lead to a limitation of the potential textual input. Nevertheless, the transformation aims towards a realistic application scenario and either contributes to this already or provides structures to adjust or extend the pipeline approach's modules to fulfill individual projects' requirements.

As a consequence of points stated before, models may not be transformed from text and entirely match the original model. However, the core elements and contents are identified with a satisfying recall and precision. The difficulties in connecting the individual model elements correctly still require human interaction and decision making. The described concept and the implemented proof of concept, the evaluation of the implemented features, and the theoretical concept allow statements about the three earlier stated questions and objectives of this approach. Regarding the three stated questions that frame the evaluation, insights were gained that provide some initial answers.

- Can the approach identify all implicitly and explicitly in the text mentioned model elements and parts?

The high recall shows that a high percentage of described elements are extracted correctly. Implicitly stated elements are additionally created based on different rules, e.g., the generation of inputs and outputs based on the verb type or a conditional clause. The phase of connecting the different model parts and generating the model can lead to a loss of elements and reduce the recall.

- Can these parts be connected correctly and combined into the overall process model that is described?

The Model Generation phase focuses on connecting all parts of the model that are not yet connected over matching inputs and outputs. However, and as mentioned, the current generation includes the risk of connecting elements not as described.

- How do incorrectly identified model parts impact the established transformation process?

Model elements that are incorrectly generated, e.g., attached with a wrong label, but logically fit the description, do not significantly impact the model and other model elements. On the contrary, elements that are not logically fitting and are not just incorrectly labeled often lead to a disturbed control flow and related elements.

While the focus was on automating the model creation from the text, insights about the other two objectives of ensuring model quality and using this approach in a learning and training environment were gained.

Automating Model Creation

As stated in the objectives, a fully automated translation is difficult to achieve. However, this approach provides an automation of crucial steps in creating a process model based on text. The transformation process can identify and extract model elements and parts, which are ideally combined into a complete process model, similar to already existing works from Friedrich, Mendling, and Puhlmann [FMP11], Ghose, Koliadis, and Chueng [GKC07] and Epure et al. [Epu+15].

Human interaction for confirmation and adjustment is still beneficial and recommended to acquire satisfying results for more complex models. The transformation approach works well on smaller and fewer complex models and can transform these without significant human interaction.

For more extensive and complex models, this approach can produce a starting point and support the modeling process as model elements as well as initial connections can be proposed to the process modeler.

In Figure 6.7 a textual input in the form of a process description is shown on the left, while on the right, the corresponding and automatically generated process model is shown. The activities and objects described in the textual input are extracted and transferred into a data frame representing the shown process model. Missing objects and labels are inferred from the activities in this step.

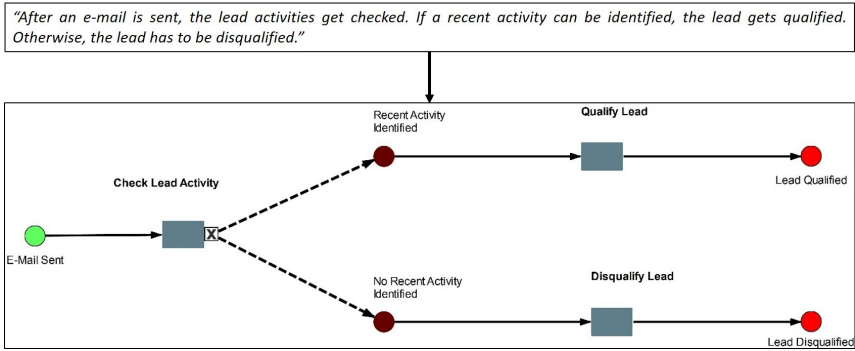


Figure 6.7: Textual Input and Automatically Generated Process Model.

Improving Model Quality

Process model quality is often perceived from a subjective viewpoint and highly dependent on the model's purpose.

However, recalling the introduced aspects of model quality taken from the guidelines and frameworks that were introduced in Section 2, some implications about the impact of an automated transformation approach on these aspects can be drawn.

Syntactic metrics are in the Horus Business Modeler's case solved by the automatic adjustment and are thus just of little interest. Only aspects such as the extent of the model, e.g., the number of elements, and the number of distinct patterns, can provide feedback about the complexity. With bigger models, the corresponding tool's adjustment algorithm reaches its limits. On the contrary, semantic quality aspects of a model or its capability to sufficiently represent

the corresponding process, meaning first and foremost completely and correctly and in a consistent manner, are of high interest.

To be able to make a more sophisticated statement about the quality of the models, a more extensive data set has to be compared. Here, the assumption that all elements present in the manually created model are also in the automatically generated model serves as orientation in comparing both models. Based on this comparison, the approach can be assessed towards its ability to achieve at least the same model quality as the manual creation.

To better understand the quality of models and practical requirements, a statement about the definition of the quality of models would have to be acquired from experts. However, this would still represent subjective opinions depending on the expert to ask as well as the practical context.

Supporting Training Participants

With the here proposed transformation approach, the connection between text and process model is automatically provided. The transformation provides precise and accurate results, which are important for an efficient training environment. These results enable new process modelers to start from simple textual inputs to familiarize them with process-oriented methods and process models. The different artifacts that serve as inputs and outputs for the distinct processing modules allow the user to follow the creation process and enable them to observe which text structures certain elements come from and how textually described structures are represented in the process model.

Additionally, the modular structure and the different outputs of each step can be used to explain each processing step individually. The presentation and explanation of each step's intermediate results can focus on explaining distinct aspects, such as the more modeling relevant parts or the rather NLP and text mining-focused parts. For instance, the indication of structures found in a text and the mapping onto model elements provides inexperienced users with examples of how specific text structures and phrases are represented in a model and establish a connection between and understanding of both.

Remarks on the Implementation

In this chapter, a proof of concept of the proposed transformation between textual inputs and process models has been described. The implementation shows that key aspects of the approach can be realized with satisfying results under some restrictions.

Although this chapter and the described proof of concept solely focus on a selected set of features relevant in a specific scenario, the implementation and the results of the evaluation enable an analysis of the current version and shed light on open potentials and issues. While future extensions of the approach as a whole are hinted at in the concluding remarks of this work, potentials and open issues specifically relevant for the implementation are briefly pointed out in the following.

First, the textual inputs are expected to fulfill a set of requirements so far to ensure a satisfying precision and recall in the transformation. Ambiguous expressions regarding language or in the text-model mapping have to be clearly defined, and a decision

towards one meaning was made. As for now, a subset of English grammar and vocabulary can be used in the written process descriptions. Even though entirely unrestricted use of natural language cannot be considered for the transformation due to its complexity and naturally inherent ambiguity, the restrictions can be reduced. The use of a so-called controlled natural language represents a compromise in this regard. Standards and joint agreements on the form of communication and thus also naming, labeling, and structuring documents, models, and other artifacts are already established to enable precise and unambiguous communication. For instance, these are used nowadays in military or air-traffic communications. Referring to an already defined controlled natural language or creating one specifically for this purpose promises a more precise definition of processing rules and, in consequence, can increase the performance of this approach.

Furthermore, any input will be processed at this stage, regardless of its correct use of language or respect paid to the input requirements. Validation of the input and processing of just validated inputs is part of an open discussion. On the one side, the consideration that only valid inputs also lead to correct process models supports the need for a validation mechanism. On the other side, even incomplete and not correctly stated inputs can be processed, and parts of models can be extracted. A partial extraction can still be of use and help the user in the modeling process. A decision would have to be made in favor of one of the two stated scenarios when shifting the attention towards the validation of the input.

This approach is constructed for and based on the English language. Rules and keywords are selected based on the underlying English grammar and vocabulary. The described pipeline archi-

itecture enables the exchange of individual modules already but requires further adjustments to integrate modules using other languages and language models. However, the adjustment to other languages is dependent on the language specifics, and the degree of necessary adjustments varies between languages and language families.

In this regard, the annotation of additional linguistic information is relying on the mentioned language models. Even though the most commonly used models provided by groups such as the Stanford NLP group²¹ or the software company Explosion²², promise a high accuracy, e.g., in annotating the correct POS-tags to the text, there are still between five to ten percent of incorrect annotations.

Furthermore, using different models to acquire additional information about the textual inputs can hold other benefits than the currently used models provide. For instance, the linguistic feature extraction and included clause extraction are based on the dependency parser of SpaCy. An alternative approach would be using the constituency parsing provided by the NLTK library. On the one side, this alternative might promise higher precision in identifying subordinate clauses but is missing the dependencies between the single text parts on the other side. A combination of both is the topic of several investigation and implementations for some time now, such as in [RCK13; GŽ12], but still often not provided beyond a prototype solution. In case a combination of constituency and dependency parsing into one set of annotations is possible and feasible, better results can be expected.

²¹ See <https://nlp.stanford.edu/>. Last accessed: 08.12.2020.

²² See <https://explosion.ai/>. Last accessed: 08.12.2020.

A feature introduced in the concept but not included in the implementation scope is the adjustment of models based on the reference point to contextual knowledge. This decision was made based on an initial investigation and tests of simple and prototype procedures of adjustments. These tests showed a reduction of the model quality based on a tendency to add wrong elements or remove elements that should be part of the model.

When considering the implementation, the adjustment process could be structured similarly to the levels of the evaluation. First, the adjustment towards model completeness might be a promising step to provide recommendations about missing or wrongly included elements. This adjustment could be achieved by combining all labels of the individual model elements of the initial model as well as the models from the knowledge base into a list of terms. With this list of terms, the Term Frequency–inverse Document Frequency (TF-IDF) can be calculated for each list. Based on the TF-IDF the lists can then be compared and based on a defined threshold, similar models can be identified. Elements that are present in models considered similar to the analyzed model can be recommended to add. In contrast, elements present in the analyzed model but not in similar models can be recommended to remove.

Nevertheless, an adjustment beyond the described model parts, e.g., the connections and control flow between the model elements, significantly increases complexity. This step would benefit from a further investigation of topics, such as model comparison and alignment.

The implementation of the proof of concept focused on selected features of the model creation process and the provision of these features was of the highest priority. However, considering the ap-

plication of this approach in real use cases, further attention has to be paid to complexity and runtime. The individual model creation steps and the respective implementation are at this stage not optimized towards their runtime. The next development steps should focus on the reduction of complexity of the individual used functions. Even though the current stage leaves room for optimization of single process steps to reduce the complexity, the required nested loops to, e.g., iterate through the sentences, phrases, and words, potentially inherit an exponential runtime.

However, the provided input is expected to include just a limited range of sentences, phrases, and described model elements. This is based on the expectation of process-oriented descriptions that describe models that are not contrary to common conventions and guidelines. Process descriptions thus are describing one model that typically consists of a limited number of elements. In general, a fitting size of a process model is assumed when the model consists of five to fifteen activities [KM96]. In this regard, the input is not limited through the implementation but can be expected to not exceed a certain size. Consequently, complexity and runtime are bound by this soft restriction on the input.

Part III

Closing Remarks

7 Conclusions

To conclude this thesis, Section 7.1 briefly recapitulates its underlying motivation and provides a concise summary of its most significant outcomes. Section 7.2 proceeds to discuss the implications of its results for research and practice. Finally, the most promising opportunities to continue the work presented in this thesis are laid out in Section 7.3.

7.1 Summary

This work is based on and motivated by a use case from industry related to process-oriented Knowledge Management and the use of process models. Process models are used as knowledge carriers intended to provide information for and at the distinct processes. These models are part of the Horus method that incorporates other model types, such as data models or organizational charts.

A crucial shortcoming of process modeling is its time-consuming nature of it. Often stakeholders have to work together who share different views, and to create process models that capture all relevant information and are understood by all participants is a challenging task. Even though current alternatives, such as process mining, often open discussions about the necessity of process modeling,

it stays a relevant task, e.g., due to the fact that process models are carrying knowledge and enable the exchange of knowledge in different scenarios.

The problem at hand has been specified based on the connection of the topics Knowledge Management, Business Process Modeling, and Natural Language Processing. Focus has been put on tackling the creation of process models by analyzing their textual descriptions and automatically deriving the model and inherent structures with the help of a rule-based NLP-approach.

One field to contribute to a possible solution is NLP, as often processes are described in user stories or requirements at the beginning of a project, and domain experts tend to write text instead of creating process models due to their lack of experience in process modeling. NLP offers a portfolio of techniques that enable the analysis of texts and text structures for further processing.

The main contribution of this thesis is the provision of an automated transformation of textual inputs into process models. This goal has been approached using a set of suitable techniques stemming from the field of NLP.

To achieve the intended goal, individual sub-tasks and steps of such a transformation have been automated to reduce the human effort that has to be put into these steps of modeling a process. The current implementation of the transformation approach includes a set of these sub-tasks; however, human interaction is still required in some parts as misinterpretations are still possible.

Overall, the proposed approach defines a process to automatically generate one type of process models, Horus procedure models, from natural language texts. The means of analyzing texts were researched extensively in the past and found their way in various

application areas. Methods of natural language processing have been applied to the textual descriptions of processes. The spectrum of available NLP-tools enables the use of existing solutions for specific tasks that, in most cases, just required minor adjustments. Building on these solutions, a rule-based system has been specified to identify patterns in the syntactic structure of texts and map them to process model elements. The elements are then extracted as model parts and further combined into one connected process model.

A first evaluation has been conducted based on a set of model-description pairs gained from the business partners. The results have shown that the automatic transformation is possible for a restricted set of models. Identifying complex structures is possible, but a challenge is still present in the correct connection of identified model parts. An increase in the complexity of the underlying and textual described process leads to a lower implementation efficiency. Different alternatives for the individual processing modules are already known and are to explore in the future to receive better results, e.g., by a combination of approaches.

7.2 Implications

The findings presented in this thesis have implications for Business Process Modeling research, research in the field of NLP, Business Process Modeling practitioners, and vendors of business process modeling tools. The most significant consequences for these parties are outlined in the following paragraphs.

Implications for Business Process Modeling

The task of Business Process Modeling remains a tedious and time-consuming one but can be facilitated with approaches described here. Full automation will stay a challenging task due to the nature of written and spoken natural languages. Additionally, the translation of textual inputs into a process or any other kind of model remains highly dependent on the used language and the availability of pre-trained models required for the different tasks.

Regarding the choice of process modeling language, not only Horus procedure models or Petri nets are suitable for this kind of transformation, but also any other modeling language, such as e.g., BPMN. However, attention has to be paid to the complexity of transforming text into a process model. Using another modeling language can increase the number of included modeling elements and possible structures, such as swim lanes in BPMN. These elements and structures have to be identified in a textual input and mapped to the corresponding parts.

With regards to Business Process Modeling as a tool in process-oriented knowledge management, the automation of process modeling reduces required effort, time, and resources that can be redistributed to other knowledge-relevant tasks. Furthermore, the facilitated and improved transformation between representation forms of knowledge can be beneficial for exchanging knowledge. The defined strategies for creating labels of activities and objects enforce labeling standards and promote consistency.

Implications for Natural Language Processing

The majority of the features are highly dependent on the used pre-trained models. Incorrect annotations can result in low pre-

cision and, in consequence, incorrect models. However, a set of publicly available models provide already a good accuracy in their annotation of POS-tags or the creation of correct syntactical trees.

Nevertheless, the diversity of languages and their distinct individual characteristics lead to different sets of challenges for each language. While some are more suitable for the described approach, others may even be impossible to incorporate due to their complexity in, e.g., their grammatical structure. The broad spectrum of Natural Language Analysis and Processing tools that already exist leave room for further advancement in the automated processing of texts in the context of Business Process Modeling. Even though a variety of solutions exists, these have to be adjusted for the distinct purpose at hand most of the time.

Implications for Practice

Even though automation is not completely achieved, and human interaction is still required, this thesis takes a step towards potential solutions that support the modeler in creating a meaningful, correct, and complete process model. The quality of created models or at least model parts can match manually created models and can even be potentially increased by using a reference point and contextual knowledge. Additionally, the mentioned standards ensure that models created with this approach are consistent in characteristics, such as labeling elements. Furthermore, this approach contributes to systems such as recommender systems for business process modeling or the reverse approach of the process model to text transformation.

Consequently, the integration of this approach as a module in existing process modeling can extend these tools by an additional feature that promises meaningful benefits.

7.3 Outlook

While the concept and implementation already provide meaningful insights into automated model generation, several aspects are not covered within this thesis. The most significant aspects and promising opportunities for further research and extension of the current implementation are stated in the following.

Improving the Text Analysis

In the case of a more extensive pre-existing data set of positive examples, other techniques can support the translation, such as machine learning approaches or artificial neural networks. The described approach is focused on a rule-based system, but a truly "intelligent" system requires the mentioned data set.

For a supervised learning approach, one would have to acquire a data set consisting of models, their descriptions and the mapping of both. In contrast, for an unsupervised learning approach, the mapping and underlying rules can be a possible result.

Next to using other techniques, the text analysis can benefit from the previously mentioned use of controlled natural language. A controlled language is a subset of a language, most prominent English, and is used, e.g., in air traffic or in the military to reduce ambiguity in communication. Such a reduction of ambiguity can

improve the precision of the text analysis, the identification of text structures, and consequently the mapping onto model elements.

Extending the Implementation

The implementation at this stage serves as a proof of concept and does only include one possible option for each module of the transformation pipeline. Potentials and open issues of this specific realization are stated in the remarks to the implementation in Chapter 6.

As NLP is a rapidly developing and still intensively researched topic, new approaches, algorithms, models, and solutions are emerging or entering a mature state that allows their efficient use. These arising solutions and those not incorporated in this implementation might provide benefits over the current modules in this scenario or be more suitable in a changed setting, such as for a different kind of language. For instance, the different available language models could be incorporated as an optional choice in the pipeline to provide the option to switch between these models and even enable a comparison of their performance. Ultimately, an improved implementation aims to achieve higher precision in the identification of the model described in a text, while on the other side can extend features, such as the mostly excluded pre-processing steps.

Transferring the Use Case

The use case followed in this thesis can be transferred to other domains. Instead of Business Process Modeling, the included discovery of processes from textual inputs can be used in other application scenarios. These scenarios may include written and spoken language,

as the latter is in many scenarios already recorded and transformed into a text in a preparation step.

One potential application scenario was hinted at earlier already and relates to the model adjustments. Model adjustments may not have to be performed automatically, as ambiguity in the interpretation requires human interaction to capture the users' intend. However, the model adjustment step can still support the process modeler. By providing recommendations of model elements, a set of possible solutions can already be provided to the user. Furthermore, Recommender systems are already part of different modeling tools, such as in Rapid Miner ¹, a tool to create models for Process Mining.

Considering research projects, such as the earlier introduced PropStop or PAMBot projects, this approach can provide benefits by enabling text mining or process discovery through improving the detailed identification of the relationship between words, sentences, and whole texts. In this regard, e.g., the recognition and analysis of texts for dialogues in chatbots can be improved to enable even more interactive and realistic communication. In another case, providing more detailed information about texts originating from, e.g., Twitter tweets, can improve techniques, such as sentiment analysis, and contribute to topics such as hate speech detection.

In Process Mining, this approach can potentially provide an alternative to log-based process mining. Data in textual form as addition to acquired event-logs can enlarge the data set that can be mined. Available textual descriptions can provide additional information not present in the log-based data. Thus, it is worth

¹ See <https://rapidminer.com/>. Last accessed: 08.12.2020.

exploring a combined approach to improve the next higher task of process discovery.

Finally, this thesis's interdisciplinary nature proves to provide incentives to explore a potential contribution of this approach to different other scenarios and incorporate insights and results from other related fields to achieve better results in the scenario at hand.

Bibliography

- [Aa+19] Han van der Aa, Claudio Di Ciccio, Henrik Leopold, and Hajo A. Reijers. “Extracting declarative process models from natural language”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2019, pp. 365–382.
- [Aal98] Wil M.P. Van der Aalst. “The Application of Petri nets to Workflow Management”. In: *Journal of circuits, systems, and computers* 8.01 (1998), pp. 21–66.
- [ABE90] Linda Argote, Sara L. Beckman, and Dennis Epple. “The persistence and transfer of learning in industrial settings”. In: *Management science* 36.2 (1990), pp. 140–154.
- [Ack89] Russell L. Ackoff. “From data to wisdom”. In: *Journal of applied systems analysis* 16.1 (1989), pp. 3–9.
- [AG97] Vincenzo Ambriola and Vincenzo Gervasi. “Processing natural language requirements”. In: *Proceedings 12th IEEE International Conference Automated Software Engineering*. IEEE. 1997, pp. 36–45.

- [AGG07] Alla Anohina, Vita Graudina, and Janis Grundspenkis. “Using concept maps in adaptive knowledge assessment”. In: *Advances in Information Systems Development*. Springer, 2007, pp. 469–479.
- [AL01] Maryam Alavi and Dorothy E. Leidner. “Knowledge management and knowledge management systems: Conceptual foundations and research issues”. In: *Management Information Systems Quarterly* (2001), pp. 107–136.
- [All95] James Allen. *Natural language understanding*. Pearson, 1995.
- [Alp+14] Sascha Alpers, Esmahan Eryilmaz, Stefan Hellfeld, and Andreas Oberweis. “Mobile Modeling Tool Based on the Horus Method”. In: *2014 International Workshop on Advanced Information Systems for Enterprises*. IEEE, 2014, pp. 65–71.
- [Amr+12a] Moussa Amrani, Jürgen Dingel, Leen Lambers, Levi Lúcio, Rick Salay, Gehan Selim, Eugene Syriani, and Manuel Wimmer. “Towards a model transformation intent catalog”. In: *Proceedings of the First Workshop on the Analysis of Model Transformations*. 2012, pp. 3–8.
- [Amr+12b] Moussa Amrani, Levi Lucio, Gehan Selim, Benoît Combemale, Jurgen Dingel, Hans Vangheluwe, Yves Le Traon, and James R Cordy. “A tridimensional approach for studying the formal verification of model transformations”. In: *2012 IEEE Fifth International Con-*

- ference on Software Testing, Verification and Validation*. IEEE. 2012, pp. 921–928.
- [And02] Ion Androutsopoulos. *Exploring Time, Tense and Aspect in Natural Language Database Interfaces*. Vol. 6. John Benjamins Publishing, 2002.
- [And96] Arthu Andersen. “The knowledge management assessment tool: External benchmarking version”. In: *The American Productivity and Quality Center, Winter* (1996).
- [ANGS11] Alla Anohina-Naumeca, Janis Grundspenkis, and Maija Strautmane. “The concept map-based assessment system: functional capabilities, evolution, and experimental results”. In: *International Journal of Continuing Engineering Education and Life Long Learning* 21.4 (2011), pp. 308–327.
- [App99] Douglas E. Appelt. “Introduction to information extraction”. In: *AI Communications* 12.3 (1999), pp. 161–172.
- [Bal08] Helmut Balzert. *Lehrbuch der Softwaretechnik: Softwaremanagement*. Spektrum Akademischer Verlag, 2008.
- [Bar95] Dorothy Leonard Barton. *Wellsprings of Knowledge: Building and Sustaining the Sources of Innovation*. Harvard Business School, 1995.
- [BDF05] Moty Ben-Dov and Ronen Feldman. “Text mining and information extraction”. In: *Data Mining and Knowledge Discovery Handbook*. Springer, 2005, pp. 801–831.

- [Ber07] Daniel M. Berry. “Ambiguity in natural language requirements documents”. In: *Monterey Workshop*. Springer. 2007, pp. 1–7.
- [BF17] Petter Bae Brandtzaeg and Asbjørn Følstad. “Why people use chatbots”. In: *International Conference on Internet Science*. Springer. 2017, pp. 377–392.
- [BFL98] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. “The berkeley framenet project”. In: *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Vol. 1*. 1998, pp. 86–90.
- [BI18] Taweh Beysolow II. “What Is Natural Language Processing?” In: *Applied Natural Language Processing with Python*. Springer, 2018, pp. 1–12.
- [BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O’Reilly Media, Inc.", 2009.
- [BKR13] Jörg Becker, Martin Kugeler, and Michael Rosemann. *Process management: a guide for the design of business processes*. Springer Science & Business Media, 2013.
- [BN08] Branimir Boguraev and Mary S. Neff. “Navigating through Dense Annotation Spaces.” In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*. 2008.

- [Boi98] Max H. Boisot. *Knowledge assets: Securing competitive advantage in the information economy*. OUP Oxford, 1998.
- [BPV12] Jörg Becker, Wolfgang Probandt, and Oliver Vering. *Grundsätze ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement*. Springer-Verlag, 2012.
- [BRR06] Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg. *Petri nets: central models and their properties: advances in petri nets 1986, part I proceedings of an advanced course bad honnef, 8.–19. September 1986*. Vol. 254. Springer, 2006.
- [BRS95] Jörg Becker, Michael Rosemann, and Reinhard Schütte. “Grundsätze Ordnungsmäßiger Modellierung”. In: *Wirtschaftsinformatik* 37.5 (1995), pp. 435–445.
- [BRVU00] Jörg Becker, Michael Rosemann, and Christoph Von Uthmann. “Guidelines of Business Process Modeling”. In: *Business Process Management*. Springer, 2000, pp. 30–49.
- [BW00] Wendi R. Bukowitz and Ruth L. Williams. *The knowledge management fieldbook*. Financial Times/Prentice Hall, 2000.
- [BW06] Madeleine Bates and Ralph M. Weischedel. *Challenges in natural language processing*. Cambridge University Press, 2006.

- [Cañ+04] A.J. Cañas, J.D. Novak, F.M. González, and L.A. Freeman. In: *Concept Maps: Theory, Methodology, Technology, Vol. 1* (2004).
- [CB02] Chun Wei Choo and Nick Bontis. *The strategic management of intellectual capital and organizational knowledge*. Oxford University Press on Demand, 2002.
- [Cha07] Daniel Chandler. “Semiotics. The Basics”. In: (2007).
- [Cho03] Gobinda G. Chowdhury. “Natural language processing”. In: *Annual review of information science and technology* 37.1 (2003), pp. 51–89.
- [Cho96a] Chun Wei Choo. “An integrated information model of the organization: The knowing organization”. In: *Retrieved February 10* (1996), p. 2006.
- [Cho96b] Chun Wei Choo. “The knowing organization: How organizations use information to construct meaning, create knowledge and make decisions”. In: *International journal of information management* 16.5 (1996), pp. 329–340.
- [CKO92] Bill Curtis, Marc I. Kellner, and Jim Over. “Process modeling”. In: *Communications of the ACM* 35.9 (1992), pp. 75–90.
- [CL18] Hyunji Chung and Sangjin Lee. “Intelligent virtual assistant knows your life”. In: *arXiv preprint arXiv:1803.00466* (2018).

- [Col+11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. “Natural language processing (almost) from scratch”. In: *Journal of machine learning research* 12.Aug (2011), pp. 2493–2537.
- [Cru86] David Alan Cruse. *Lexical semantics*. Cambridge university press, 1986.
- [CW14] Erik Cambria and Bebo White. “Jumping NLP curves: A review of natural language processing research”. In: *IEEE Computational intelligence magazine* 9.2 (2014), pp. 48–57.
- [Dal17] Kimiz Dalkir. *Knowledge management in theory and practice*. MIT press, 2017.
- [DDLB98] Thomas H. Davenport, David W. De Long, and Michael C. Beers. “Successful knowledge management projects”. In: *Sloan management review* 39.2 (1998), pp. 43–57.
- [DMM08] Marie-Catherine De Marneffe and Christopher D. Manning. *Stanford typed dependencies manual*. Tech. rep. Technical report, Stanford University, 2008.
- [DP07] Marina Du Plessis. “Knowledge management: what makes complex implementations successful?” In: *Journal of Knowledge management* (2007).
- [DP11] Kevin C. Desouza and Scott Pacquette. *Knowledge management: An introduction*. Neal-Schuman Publishers, 2011.

- [DP98] Thomas H. Davenport and Laurence Prusak. *Working knowledge: How organizations manage what they know*. Harvard Business Press, 1998.
- [DS11] Deva Kumar Deeptimahanti and Ratna Sanyal. “Semi-automatic generation of UML models from natural language requirements”. In: *Proceedings of the 4th India Software Engineering Conference*. 2011, pp. 165–174.
- [ELS10] Javier Esparza, Martin Leucker, and Maximilian Schlund. “Learning workflow petri nets”. In: *International Conference on Applications and Theory of Petri Nets*. Springer. 2010, pp. 206–225.
- [Epu+15] Elena V. Epure, Patricia Martín-Rodilla, Charlotte Hug, Rebecca Deneckère, and Camille Salinesi. “Automatic process model discovery from textual methodologies”. In: *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*. IEEE. 2015, pp. 19–30.
- [EW16] Lisa Ehrlinger and Wolfram Wöß. “Towards a Definition of Knowledge Graphs.” In: *SEMANTiCS (Posters, Demos, SuCCESS) 48 (2016)*, pp. 1–4.
- [FA18] Thomas Freytag and Philip Allgaier. “WoPeD goes NLP: Conversion between Workflow Nets and Natural Language.” In: *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018)*. 2018, pp. 101–105.

- [Far95] James A. Farrell. *Compiler Basics*. 1995.
- [FCMP03] Abílio Fernandes, Ana Maria de C Moura, and Fábio Porto. “An ontology-based approach for organizing sharing, and querying knowledge objects on the Web”. In: *14th International Workshop on Database and Expert Systems Applications, 2003. Proceedings*. IEEE. 2003, pp. 604–609.
- [Fel99] Susan Feldman. “NLP meets the Jabberwocky: Natural language processing in information retrieval”. In: *Online (Weston, CT) 23* (1999), pp. 62–73.
- [Fil+13] Hans-Georg Fill, Susan Hickl, Dimitris Karagiannis, Andreas Oberweis, and Andreas Schoknecht. “A Formal Specification of the Horus Modeling Language Using FDMM.” In: *Wirtschaftsinformatik*. 2013, p. 73.
- [FM04] Joseph M. Firestone and Mark W. McElroy. “Organizational learning and knowledge management: the relationship”. In: *The Learning Organization* (2004).
- [FMP11] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. “Process model generation from natural language text”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2011, pp. 482–496.
- [GB01] Paul R. Gamble and John Blackwell. *Knowledge management: A state of the art guide*. Kogan Page Publishers, 2001.

- [GBK07] M. E. Greiner, T. Böhmann, and H. Krcmar. “A strategy for knowledge management”. In: *Journal of knowledge management* (2007).
- [GG08] Vita Graudina and Janis Grundspenkis. “Concept map generation from OWL ontologies”. In: *Proceedings of the third international conference on concept mapping, Tallinn, Estonia and Helsinki, Finland*. 2008, pp. 263–270.
- [GKC07] Aditya Ghose, George Koliadis, and Arthur Chueng. “Process discovery from model and text artefacts”. In: *2007 IEEE Congress on Services (Services 2007)*. IEEE. 2007, pp. 167–174.
- [GL14] Yoav Goldberg and Omer Levy. “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method”. In: *arXiv preprint arXiv:1402.3722* (2014).
- [GMS01] Andrew H. Gold, Arvind Malhotra, and Albert H. Segars. “Knowledge management: An organizational capabilities perspective”. In: *Journal of management information systems* 18.1 (2001), pp. 185–214.
- [Gre+04] Gianluigi Greco, Antonella Guzzo, Luigi Pontieri, and Domenico Saccà. “An ontology-driven process modeling framework”. In: *International Conference on Database and Expert Systems Applications*. Springer. 2004, pp. 13–23.

- [Gri+17] Christian Grimme, Mike Preuss, Lena Adam, and Heike Trautmann. “Social bots: Human-like by means of human control?” In: *Big data 5.4* (2017), pp. 279–293.
- [GSB09] J. C. Gonçalves, F. M. Santoro, and F. A. Baião. “Business process mining from group stories”. In: *2009 13th International Conference on Computer Supported Cooperative Work in Design*. IEEE, 2009, pp. 161–166.
- [GŽ12] Nathan Green and Zdeněk Žabokrtský. “Hybrid combination of constituency and dependency trees into an ensemble dependency parser”. In: *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*. 2012, pp. 19–26.
- [Hav05] Michael Havey. *Essential Business Process Modeling*. O’Reilly Media, Inc., 2005.
- [HBH18] Donald Hislop, Rachelle Bosua, and Remko Helms. *Knowledge management in organizations: A critical introduction*. Oxford University Press, 2018.
- [Hea03] Marti Hearst. “What is text mining”. In: *SIMS, UC Berkeley 5* (2003).
- [Hen74] Nicholas L. Henry. “Knowledge management: a new concern for public administration”. In: *Public Administration Review* (1974), pp. 189–196.
- [HJ99] Clyde W. Holsapple and Kshiti D. Joshi. “Description and analysis of existing knowledge management frameworks”. In: *Proceedings of the 32nd Annual*

Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers. IEEE. 1999.

- [Höf07] Peter Höfferer. “Achieving Business Process Model Interoperability Using Metamodels and Ontologies.” In: *Proceedings of the Fifteenth European Conference on Information Systems, ECIS 2007*. 2007, pp. 1620–1631.
- [HS06] Irit Hadar and Pnina Soffer. “Variations in conceptual modeling: classification and ontological analysis”. In: *Journal of the Association for Information Systems 7.8* (2006), p. 20.
- [HS13] Martin Haspelmath and Andrea D Sims. *Understanding morphology*. Routledge, 2013.
- [HS92] William John Hutchins and Harold L. Somers. *An introduction to machine translation*. Vol. 362. Academic Press London, 1992.
- [HTT00] Christophe Hirel, Bruno Tuffin, and Kishor S. Trivedi. “Snpn: Stochastic petri nets. version 6.0”. In: *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer. 2000, pp. 354–357.
- [Hub91] George P. Huber. “Organizational learning: The contributing processes and the literatures”. In: *Organization science 2.1* (1991), pp. 88–115.

- [Hut04] William John Hutchins. “The Georgetown-IBM experiment demonstrated in January 1954”. In: *Conference of the Association for Machine Translation in the Americas*. Springer. 2004, pp. 102–114.
- [ID10] Nitin Indurkha and Fred J. Damerau. *Handbook of natural language processing*. Vol. 2. CRC Press, 2010.
- [IL99] Juhani Iivari and Henry Linger. “Knowledge work as collaborative work: A situated activity theory view”. In: *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences*. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers. IEEE. 1999.
- [Ind+09] Marta Indulska, Jan Recker, Michael Rosemann, and Peter Green. “Business process modeling: Current issues and future challenges”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2009, pp. 501–514.
- [JD12] S.D. Joshi and Dhanraj Deshpande. “Textual requirement analysis for UML diagram extraction by using NLP”. In: *International journal of computer applications* 50.8 (2012), pp. 42–46.
- [JHS01] Stefan Jablonski, Stefan Horn, and Michael Schlundt. “Process oriented knowledge management”. In: *Proceedings eleventh international workshop on research issues in data engineering. document management for data intensive business and scientific applications. ride 2001*. IEEE. 2001, pp. 77–84.

- [Jot+19] Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Gabriel Murray. “Discourse Analysis and Its Applications”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. 2019, pp. 12–17.
- [Jur00] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.
- [KB06] Brane Kalpič and Peter Bernus. “Business process modeling through the knowledge management perspective”. In: *Journal of knowledge management* (2006).
- [KG08] Sven J. Körner and Tom Gelhausen. “Improving Automatic Model Creation Using Ontologies.” In: *Twenty-First International Conference on Software Engineering and Knowledge Engineering (SEKE)*. 2008, pp. 691–696.
- [KHO11] Agnes Koschmider, Thomas Hornung, and Andreas Oberweis. “Recommendation-based editor for business process modeling”. In: *Data & Knowledge Engineering* 70.6 (2011), pp. 483–503.
- [KLS95] John Krogstie, Odd Ivar Lindland, and Guttorm Sindre. “Defining quality aspects for conceptual models”. In: *Information System Concepts*. Springer, 1995, pp. 216–231.
- [KM96] Nereu F. Kock and Robert J. McQueen. “Product flow, breadth and complexity of business processes: An empirical study of 15 business processes in three or-

- ganizations”. In: *Business Process Re-engineering & Management Journal* (1996).
- [KOS14] Holger Kohl, Ronald Orth, and Erik Steinhöfel. “Process-oriented knowledge management in SMEs”. In: *Proceedings of the 15th European Conference on Knowledge Management (ECKM)*. 2014, pp. 563–570.
- [KOS15] Holger Kohl, Ronald Orth, and Erik Steinhöfel. “A Practical Approach to Process-Oriented Knowledge Management.” In: *Electronic Journal of Knowledge Management* 13.1 (2015).
- [KR15] Agnes Koschmider and Hajo A. Reijers. “Improving the process of process modelling by the use of domain process patterns”. In: *Enterprise Information Systems* 9.1 (2015), pp. 29–57.
- [Kro16] John Krogstie. “Quality of business process models”. In: *Quality in Business Process Modeling*. Springer, 2016, pp. 53–102.
- [KSJ06] John Krogstie, Guttorm Sindre, and Håvard Jørgensen. “Process models representing knowledge for action: a revised quality framework”. In: *European Journal of Information Systems* 15.1 (2006), pp. 91–102.
- [LB02] Edward Loper and Steven Bird. “NLTK: the natural language toolkit”. In: *arXiv preprint cs/0205028* (2002).
- [LB92] Dorothy Leonard-Barton. “Core capabilities and core rigidities: A paradox in managing new product development”. In: *Strategic management journal* 13.S1 (1992), pp. 111–125.

- [LC00] Hsiangchu Lai and Tsai-hsin Chu. “Knowledge management: A review of theoretical frameworks and industrial cases”. In: *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*. IEEE. 2000.
- [LCM03] Gondy Leroy, Hsinchun Chen, and Jesse D. Martinez. “A shallow parser based on closed-class words to capture relations in biomedical text”. In: *Journal of biomedical Informatics* 36.3 (2003), pp. 145–158.
- [Len02] Anssi Lensu. *Computationally intelligent methods for qualitative data analysis*. 23. University of Jyväskylä, 2002.
- [Leo+13] Henrik Leopold, Rami-Habib Eid-Sabbagh, Jan Mendling, Leonardo G. Azevedo, and Fernanda A. Baião. “Detection of naming convention violations in process models for different languages”. In: *Decision Support Systems* 56 (2013), pp. 310–325.
- [Lid01] Elizabeth D. Liddy. “Natural language processing”. In: *Encyclopedia of Library and Information Science* (2001).
- [LMP14] Henrik Leopold, Jan Mendling, and Artem Polyvyanyy. “Supporting process model validation through natural language generation”. In: *IEEE Transactions on Software Engineering* 40.8 (2014), pp. 818–840.

- [Lóp+18] Hugo A López, Søren Debois, Thomas T Hildebrandt, and Morten Marquard. “The Process Highlighter: From Texts to Declarative Processes and Back.” In: *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018)* 2196 (2018), pp. 66–70.
- [Lóp+19] H. A. López, M. Marquard, L. Muttenthaler, and R. Strømsted. “Assisted declarative process creation from natural language descriptions”. In: *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE. 2019, pp. 96–99.
- [Lor18] Steven Loria. “textblob Documentation”. In: *Release 0.15 2* (2018).
- [LR+11] Marcello La Rosa, Arthur HM Ter Hofstede, Petia Wohed, Hajo A Reijers, Jan Mendling, and Wil M. P. van der Aalst. “Managing process model complexity via concrete syntax modifications”. In: *IEEE Transactions on Industrial Informatics* 7.2 (2011), pp. 255–265.
- [LSS94] Odd Ivar Lindland, Guttorm Sindre, and Arne Solvberg. “Understanding quality in conceptual modeling”. In: *IEEE software* 11.2 (1994), pp. 42–49.
- [Lúc+16] L. Lúcio, M. Amrani, J. Dingel, L. Lambers, R. Salay, G. M. K. Selim, E. Syriani, and M. Wimmer. “Model transformation intents and their properties”. In: *Software & systems modeling* 15.3 (2016), pp. 647–684.

- [LVD09] Niels Lohmann, Eric Verbeek, and Remco Dijkman. “Petri net transformations for business processes—a survey”. In: *Transactions on petri nets and other models of concurrency II*. Springer, 2009, pp. 46–63.
- [MAA08] F. Meziane, N. Athanasakis, and S. Ananiadou. “Generating natural language specifications from UML class diagrams”. In: *Requirements Engineering* 13.1 (2008), pp. 1–18.
- [Man+14] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. “The Stanford CoreNLP natural language processing toolkit”. In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.
- [Màr+08] Lluís Màrquez, Xavier Carreras, Kenneth C Litkowski, and Suzanne Stevenson. “Semantic role labeling: an introduction to the special issue”. In: (2008).
- [McD10] David D. McDonald. “Natural Language Generation.” In: *Handbook of Natural Language Processing* 2 (2010), pp. 121–144.
- [McE99] Mark W. McElroy. “The knowledge life cycle”. In: *Proceedings of the ICM. Conference on KM*. 1999.
- [Men08] Jan Mendling. *Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness*. Vol. 6. Springer Science & Business Media, 2008.

- [MHV03] Kai Mertins, Peter Heisig, and Jens Vorbeck. *Knowledge management: concepts and best practices*. Springer Science & Business Media, 2003.
- [Mil95] George A. Miller. “WordNet: a lexical database for English”. In: *Communications of the ACM* 38.11 (1995), pp. 39–41.
- [MMB15] Richard Mrasek, Jutta Mülle, and Klemens Böhm. “Automatic generation of optimized process models from declarative specifications”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2015, pp. 382–397.
- [Moo+02] Daniel L. Moody, Guttorm Sindre, Terje Brasethvik, and Arne Sølvberg. “Evaluating the quality of process models: Empirical testing of a quality framework”. In: *International Conference on Conceptual Modeling*. Springer. 2002, pp. 380–396.
- [Moo05] Daniel L. Moody. “Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions”. In: *Data & Knowledge Engineering* 55.3 (2005), pp. 243–276.
- [MR03] Ronald Maier and Ulrich Remus. “Implementing process-oriented knowledge management strategies”. In: *Journal of knowledge management* (2003).
- [MRA10] Jan Mendling, Hajo A Reijers, and Wil M. P. van der Aalst. “Seven process modeling guidelines (7PMG)”. In: *Information and Software Technology* 52.2 (2010), pp. 127–136.

- [MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.
- [MSR08] C. D. Manning, Hinrich Schütze, and P. Raghavan. *Introduction to information retrieval*. Cambridge university press, 2008.
- [MT00] David Milward and James Thomas. “From information retrieval to information extraction”. In: *ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval*. 2000, pp. 85–97.
- [Mur89] Tadao Murata. “Petri nets: Properties, analysis and applications”. In: *Proceedings of the IEEE 77.4* (1989), pp. 541–580.
- [MVG06] Tom Mens and Pieter Van Gorp. “A taxonomy of model transformation”. In: *Electronic Notes in Theoretical Computer Science 152* (2006), pp. 125–142.
- [MZ96] Marc H. Meyer and Michael H. Zack. “The Design and Development of Information Products”. In: *Sloan Management Review* (1996).
- [NC06] Joseph D. Novak and Alberto J. Cañas. “The theory underlying concept maps and how to construct them”. In: *Florida Institute for Human and Machine Cognition 1.1* (2006), pp. 1–31.
- [NCL06] Hong-Seok Na, O-Hoon Choi, and Jung-Eun Lim. “A method for building domain ontologies based on the transformation of UML models”. In: *Fourth International Conference on Software Engineering Research*,

- Management and Applications (SERA'06)*. IEEE. 2006, pp. 332–338.
- [Nel+12] H. James Nelson, Geert Poels, Marcela Genero, and Mario Piattini. “A conceptual modeling quality framework”. In: *Software Quality Journal* 20.1 (2012), pp. 201–228.
- [NH15] Tim Niesen and Constantin Houy. “Zur Nutzung von Techniken der Natürlichen Sprachverarbeitung für die Bestimmung von Prozessmodellähnlichkeiten-Review und Konzeptentwicklung.” In: *Wirtschaftsinformatik*. 2015, pp. 1829–1843.
- [NHLT17] Thi Hoa Hue Nguyen, Tuan Phan Hong, and Nhan Le Thanh. “An ontological approach for organizing a knowledge base to share and reuse business workflow templates”. In: *2017 Seventh International Conference on Information Science and Technology (ICIST)*. IEEE. 2017, pp. 271–277.
- [NOMC11] Prakash M. Nadkarni, Lucila Ohno-Machado, and Wendy W. Chapman. “Natural language processing: an introduction”. In: *Journal of the American Medical Informatics Association* 18.5 (2011), pp. 544–551.
- [Non94] Ikujiro Nonaka. “A Dynamic Theory of Organizational Knowledge Creation”. In: *Organization science* 5.1 (1994), pp. 14–37.
- [Nov91] Joseph Novak. “Clarify with concept maps”. In: *The science teacher* 58.7 (1991), p. 44.

- [Nov95] Joseph D. Novak. “Concept mapping to facilitate teaching and learning”. In: *Prospects* 25.1 (1995), pp. 79–86.
- [NT95] Ikujiro Nonaka and Hirotaka Takeuchi. *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford university press, 1995.
- [OBS12] Sven Overhage, Dominik Birkmeier, and Sebastian Schlauderer. “Quality marks, metrics, and measurement procedures for business process models”. In: *Business & Information Systems Engineering* 4.5 (2012), pp. 229–246.
- [Off97] Steve Offsey. “Knowledge management: linking people to knowledge for bottom line results”. In: *Journal of knowledge management* 1.2 (1997), pp. 113–122.
- [Omo15] Funmilola Olubunmi Omotayo. “Knowledge Management as an important tool in Organisational Management: A Review of Literature”. In: *Library Philosophy and Practice* 1.2015 (2015), pp. 1–23.
- [Pau17] Heiko Paulheim. “Knowledge graph refinement: A survey of approaches and evaluation methods”. In: *Semantic web* 8.3 (2017), pp. 489–508.
- [Pes08] Maja Pesic. “Constraint-based workflow management systems: shifting control to users”. In: (2008).
- [Pet62] Carl Adam Petri. “Kommunikation mit Automaten”. In: *Ph. D. thesis, University of Bonn* (1962).

- [Pet77] James L. Peterson. “Petri nets”. In: *ACM Computing Surveys (CSUR)* 9.3 (1977), pp. 223–252.
- [Pfl18] Nicolas Pflanzl. *Gamification for business process modeling*. Verlag readbox publishing GmbH, readbox uni-press, 2018.
- [Pic+11] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, and H. A. Reijers. “Imperative versus declarative process modeling languages: An empirical investigation”. In: *International Conference on Business Process Management*. Springer. 2011, pp. 383–394.
- [PK04] Martha Palmer and K. Loper Kipper. “VerbNet”. In: *The Oxford Handbook of Cognitive Science*. 2004.
- [Pol15] Michael Polanyi. *Personal knowledge: Towards a post-critical philosophy*. University of Chicago Press, 2015.
- [PRR97] Gilbert Probst, Steffen Raub, and Kai Romhardt. *Wissen managen*. Springer, 1997.
- [PV13] Nicolas Pflanzl and Gottfried Vossen. “Human-oriented challenges of social BPM: an overview”. In: *Enterprise Modelling and Information Systems Architectures (EMISA 2013)* (2013).
- [PV14] Nicolas Pflanzl and Gottfried Vossen. “Challenges of social business process management”. In: *2014 47th Hawaii International Conference on System Sciences*. IEEE. 2014, pp. 3868–3877.

- [RCK13] Xiaona Ren, Xiao Chen, and Chunyu Kit. “Combine constituent and dependency parsing via reranking”. In: *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.
- [RD00] Ehud Reiter and Robert Dale. *Building natural language generation systems*. Cambridge university press, 2000.
- [Rei12] Wolfgang Reisig. *Petri nets: an introduction*. Vol. 4. Springer Science & Business Media, 2012.
- [Rei13] Wolfgang Reisig. *Understanding petri nets: modeling techniques, analysis methods, case studies*. Springer, 2013.
- [RHF02] B. Rosario, M. A. Hearst, and C. Fillmore. “The descent of hierarchy, and selection in relational semantics”. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2002, pp. 247–254.
- [RL00] Ulrich Remus and Franz Lehner. “The role of process-oriented enterprise modeling in designing process-oriented knowledge management systems”. In: *Proceedings of the AAAI Symposium on Bringing Knowledge to Business Processes*. Stanford, CA, USA. 2000, pp. 30–36.
- [RMR10] Hajo A. Reijers, Jan Mendling, and Jan Recker. “Business process quality management”. In: *Handbook on Business Process Management 1*. Springer, 2010, pp. 167–185.

- [Ros96] Michael Rosemann. “Multiperspektivische Informationsmodellierung auf der Basis der Grundsätze ordnungsmäßiger Modellierung”. In: *Management & Computer* 4.4 (1996), pp. 219–226.
- [Row07] Jennifer Rowley. “The wisdom hierarchy: representations of the DIKW hierarchy”. In: *Journal of information science* 33.2 (2007), pp. 163–180.
- [RS03] Ulrich Remus and Stephan Schub. “A blueprint for the implementation of process-oriented knowledge management”. In: *Knowledge and Process Management* 10.4 (2003), pp. 237–253.
- [ŘS11] Radim Řehůřek and Petr Sojka. “Gensim—statistical semantics in python”. In: (2011).
- [RTT16] Maximilian Riefer, Simon Felix Ternis, and Tom Thaler. “Mining Process Models from Natural Language Text: A State-of-the-Art Analysis”. In: *Tagungsband der Multikonferenz Wirtschaftsinformatik. Multikonferenz Wirtschaftsinformatik (MKWI-16), March 9-11, Illmenau, Germany* (2016), pp. 1–12.
- [Rug98] Rudy Ruggles. “The state of the notion: knowledge management in practice”. In: *California management review* 40.3 (1998), pp. 80–89.
- [SA07] Bayan Abu Shawar and Eric Atwell. “Chatbots: are they really useful?” In: *Zeitschrift für Computerlinguistik und Sprachtechnologie* (2007), p. 29.

- [Sad15] Malgorzata Sadowska. “An approach to assessing the quality of business process models expressed in BPMN”. In: *e-Informatica Software Engineering Journal* 9.1 (2015).
- [SBP08] Flávia Maria Santoro, Marcos R.S. Borges, and José A. Pino. “Tell us your process: A group storytelling approach to cooperative process modeling”. In: *2008 12th International Conference on Computer Supported Cooperative Work in Design*. IEEE, 2008, pp. 29–34.
- [Sch+11] F. Schönthaler, G. Vossen, A. Oberweis, and T. Karle. *Geschäftsprozesse für Business Communities: Modellierungssprachen, Methoden, Werkzeuge*. Oldenbourg, 2011.
- [SFCP17] Josep Sànchez-Ferreres, Josep Carmona, and Lluís Padró. “Aligning textual and graphical descriptions of processes through ILP techniques”. In: *International Conference on Advanced Information Systems Engineering*. Springer, 2017, pp. 413–427.
- [SG+12] Laura Sánchez-González, Félix García, Francisco Ruiz, and Jan Mendling. “Quality indicators for business process models from a gateway complexity perspective”. In: *Information and Software Technology* 54.11 (2012), pp. 1159–1174.
- [SN00] August-Wilhelm Scheer and Markus Nüttgens. “ARIS Architecture and Reference Models for Business Process Management”. In: *Business Process Management*. Springer, 2000, pp. 376–389.

- [SP10] Avik Sinha and Amit Paradkar. “Use cases to process specifications in business process modeling notation”. In: *2010 IEEE International Conference on Web Services*. IEEE. 2010, pp. 473–480.
- [SSJP10] Avik Sinha, Stanley M Sutton Jr, and Amit Paradkar. “Text2Test: Automated inspection of natural language use cases”. In: *2010 Third International Conference on Software Testing, Verification and Validation*. IEEE. 2010, pp. 155–164.
- [STA05] August-Wilhelm Scheer, Oliver Thomas, and Otmar Adam. “Process Modeling Using Event-Driven Process Chains.” In: *Process-aware information systems* 119 (2005).
- [Sta91] Ronald Stamper. “The semiotic framework for information systems research”. In: *Information systems research: Contemporary approaches and emergent traditions* (1991), pp. 515–528.
- [SZ95] Eric W. Stein and Vladimir Zwass. “Actualizing organizational memory with information systems”. In: *Information systems research* 6.2 (1995), pp. 85–117.
- [Tan+98] S. Tan, H.H. Teo, B. Tan, and K.K Wei. “Developing a preliminary framework for knowledge management in organizations”. In: *AMCIS 1998 Proceedings* (1998), p. 211.
- [Thi99] Robert J. Thierauf. *Knowledge management systems for business*. Greenwood Publishing Group, 1999.

- [Toc+07] Eran Toch, Avigdor Gal, Iris Reinhartz-Berger, and Dov Dori. “A semantic approach to approximate service retrieval”. In: *ACM Transactions on Internet Technology (TOIT)* 8.1 (2007), pp. 1294148–1294150.
- [Usc15] Michael Uschold. “Ontology and database schema: What’s the difference?” In: *Applied Ontology* 10.3-4 (2015), pp. 243–258.
- [Usz00] Hans Uszkoreit. “Language technology a first overview”. In: *German Research Center for Artificial Intelligence, Saarbrücken* (2000).
- [VDA+11] Wil M. P. Van Der Aalst, Kees M. Van Hee, Arthur H.M. Ter Hofstede, Natalia Sidorova, H.M.W. Verbeek, Marc Voorhoeve, and Moe Thandar Wynn. “Soundness of workflow nets: classification, decidability, and analysis”. In: *Formal Aspects of Computing* 23.3 (2011), pp. 333–363.
- [VH02] Wil M. P. Van der Aalst and Arthur H.M. ter Hofstede. “Workflow patterns: On the expressive power of (petri-net-based) workflow languages”. In: *Proceedings of the fourth workshop on the practical use of coloured petri nets and CPN Tools (CPN 2002)*. Vol. 560. 2002, pp. 1–20.
- [VK98] Georg Von Krogh. “Care in knowledge creation”. In: *California management review* 40.3 (1998), pp. 133–153.
- [VKR95] Georg Von Krogh and Johan Roos. *Organizational epistemology*. Springer, 1995.

- [Web03] Dean Weber. *Network interactive user interface using speech recognition and natural language processing*. US Patent 6,532,444. Google Patents, 2003.
- [Wes10] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Publishing Company, Incorporated, 2010.
- [Whi04] Stephen A. White. "Introduction to BPMN". In: *IBM Cooperation 2.0* (2004), p. 0.
- [Wii94] Karl M. Wiig. *Knowledge Management Foundations: Thinking about Thinking-how People and Organizations Represent, Create, and Use Knowledge*. Schema Press, Limited, 1994.
- [Win72] Terry Winograd. "Understanding natural language". In: *Cognitive psychology* 3.1 (1972), pp. 1–191.
- [WK05] Robert Woitsch and Dimitris Karagiannis. "Process oriented knowledge management: A service based approach." In: *Journal of Universal Computer Science* 11.4 (2005), pp. 565–588.
- [Yin+05] Y. Yin, J. Vanides, M.A. Ruiz-Primo, C.C. Ayala, and R.J. Shavelson. "Comparison of two concept-mapping techniques: Implications for scoring, interpretation, and use". In: *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching* 42.2 (2005), pp. 166–184.

- [ZC06] Ji Zhang and Betty H.C. Cheng. “Model-based development of dynamically adaptive software”. In: *Proceedings of the 28th international conference on Software engineering*. 2006, pp. 371–380.
- [ZDJ02] Tong Zhang, Fred Damerau, and David Johnson. “Text chunking based on a generalization of winnow”. In: *Journal of Machine Learning Research* 2.Mar (2002), pp. 615–637.
- [ZZ94] Richard Zurawski and MengChu Zhou. “Petri nets and industrial applications: A tutorial”. In: *IEEE Transactions on industrial electronics* 41.6 (1994), pp. 567–583.

List of Abbreviations

7PMG Seven Process Modeling Guidelines.

BNF Backus-Naur Form.

BPM Business Process Management.

BPMN Business Process Model and Notation.

BPMod Business Process Modeling.

CL Computational Linguistics.

EPC Event-driven Process Chain.

ERM Entity–relationship model.

GoM Guidelines of Modeling.

HBM Horus Business Modeler.

KM Knowledge Management.

KMS Knowledge Management System.

List of Abbreviations

NER Named Entity Recognition.

NLP Natural Language Processing.

NLTK Natural Language Toolkit.

OWL Web Ontology Language.

PN Petri Net.

PNML Petri Net Modeling Language.

POS Part-of-Speech.

PP Prepositional Phrase.

SEQUAL Semiotic Quality Model.

SVO Subject, Verb, Object.

TF-IDF Term Frequency–inverse Document Frequency.

VP Verb Phrase.

WN Workflow Net.

WSD Word Sense Disambiguation.

WWU Westfälische Wilhelms-Universität.

Text to Process Model: Automating Process Model Creation from Text

Felix Reinold Nolte

The description and visualization of processes and underlying requirements are a core task in IT projects today. The translation of information between different forms of representation is used to include these efficiently in the different steps of a project, e.g., to describe the as-is and to-be states. However, the transformation of a process description into the visual representation of a process model is still a costly and complex task that often comes with an inevitable loss of information. One common challenge to be resolved during the process of modeling arises with the diversity of stakeholders involved. To improve the communication between stakeholders, different techniques of Natural Language Processing (NLP) are combined in this work to analyze the textual descriptions of Petri nets, in order to enable a transformation of texts into process models. The performed steps of analysis and transformation are means to ensure conformance between process models and their textual description.

37,90 €

ISBN 978-3-8405-0256-9



9 783840 502569