TU WIEN Informatics

Umwandlung von Geschäftsprozessmodellen in systemdynamische Modelle

Entwicklung eines Transformationswerkzeugs

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Masterstudium Business Informatics

eingereicht von

Eralda Ruci
Matrikelnummer 12038076

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Henderik Proper, PhD

Wien, 14. Mai 2024

Eralda Ruci	Henderik Proper

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

Transforming Business Process Models into System Dynamics Models

Developing a Transformation Tool

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Master programme Business Informatics

by

Eralda Ruci
Registration Number 12038076

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Henderik Proper, PhD

Vienna, 14th May, 2024

Eralda Ruci	Henderik Proper

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Eralda Ruci

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 14. Mai 2024

Eralda Ruci

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Zuerst und vor allem möchte ich meinem Betreuer, Henderik Proper, meinen tiefsten Dank aussprechen für seine unschätzbare Unterstützung und fachkundige Anleitung während des gesamten Prozesses der Erstellung dieser Arbeit. Sein tiefgreifendes Wissen und sein aufschlussreiches Feedback waren entscheidend für die Gestaltung der Durchführung dieser Arbeit.

Zuletzt möchte ich meiner Familie und meinen Liebsten meinen herzlichen Dank aussprechen, die mir während dieser herausfordernden Reise unermüdlichen moralischen Beistand, Motivation und Unterstützung geboten haben.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

First and foremost, I extend my deepest gratitude to my supervisor, Henderik Proper, for his invaluable support and expert guidance throughout the process of writing this thesis. His profound knowledge and insightful feedback have been crucial in shaping the execution of this work.

Lastly, I would like to express my heartfelt thanks to my family and loved ones, who have provided endless moral support, motivation, and assistance throughout this challenging journey.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Diese Arbeit untersucht die Integration von Business Process Model and Notation (BPMN) und System Dynamics (SD), um die Modellierung und Analyse im Geschäftsprozessmanagement zu verbessern. Ziel dieser Forschung ist die Entwicklung eines automatisierten Werkzeugs, das BPMN-Diagramme in Stock-and-Flow-Diagramme übersetzt, indem es die strukturellen Vorteile von BPMN für eine genaue Prozesskartierung und die dynamischen Modellierungsfähigkeiten von SD nutzt.

Im Rahmen dieser Studie implementieren wir ein experimentelles Werkzeug auf der Simplified Modelling Platform, basierend auf den ersten von [57] vorgeschlagenen Kartierungsmethodologien. Die Simplified Modelling Platform bringt einen Low-Code-Ansatz in Modellierung und Metamodellierung, der aufgrund seiner anpassbaren Notationen, relativ flexiblen Benutzeroberfläche und Fähigkeit, Visualisierungen und Transformationen bereitzustellen, für geschäftliche und Forschungsanwendungen geeignet ist.

Die Dissertation zeigt auf, dass das Werkzeug zwar erfolgreich die Machbarkeit der automatischen Umwandlung von BPMN- in SD-Modelle demonstriert, jedoch eine stabilere Entwicklungsplattform benötigt wird, um die Funktionen der Plattform zu erweitern. Ferner sind zur Optimierung der automatischen Transformation weitere Verbesserungen notwendig, um den Einsatz komplexerer Modelle zu erleichtern. Diese Forschungsarbeit liefert nicht nur eine praktische Lösung für die (automatische) Transformation von BPMN- zu SD-Modellen im Bereich des Geschäftsprozessmanagements, sondern schafft auch eine Grundlage für zukünftige Forschungen, die verschiedene Modellierungsansätze integrieren, um ein tieferes Verständnis zu entwickeln und organisatorische Prozesse zu optimieren.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

This thesis explores the integration of Business Process Model and Notation (BPMN) and System Dynamics (SD) to enhance modeling and analysis in business process management. This research aims to develop an automated tool that translates BPMN diagrams to Stock-and-Flow Diagrams by leveraging the structural advantages of BPMN for accurate process mapping and the dynamic modelling capabilities of SD.

Throughout this study we implement an experimental tool in the Simplified Modelling Platform, based on the first mapping proposed by [57]. Simplified modeling platform brings a low-code approach in modeling and meta-modeling, that is appropriate for business and research applications due to its adjustable notations, relatively flexible user interface, and ability to provide visualisations and transformations.

The thesis asserts that although the tool effectively demonstrates the feasibility to automatically transform BPMN to SD models, a more stable development environment is required to extend the platform's functionalities. Furthermore, to optimize the automatic transformation, further improvements are required, in order to facilitate the use of more complex models. This research not only provides a practical answer on (automatically) transforming BPMN-to-SD models in the field of business process management, but also establishes a guide for future research on integrating diverse modelling approaches to gain a better understanding and optimise organisational processes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

In the dynamic and ever-changing business world nowadays, businesses that want to remain efficient and competitive must effectively design and analyze their business processes. However, the rising complexity of businesses and their environment results in limitations of existing modeling approaches. To enhance the overall modeling and analysis capabilities, combining existing modeling methods can leverage the strengths of each method. This approach is also used in [28, 56, 37, 57, 67]. We believe that Business Process Modeling Notation (BPMN) diagrams, which present a clear and comprehensive view of the sequence of actions and events in a process, can be used as the basis for developing System Dynamics (SD) models. This enables more effective modeling and analysis of complex business processes, while also incorporating other important (dynamic) aspects from the environment of these business processes [57]. SD models are often used for policy analysis, given their holistic approach. They allow modelers to simulate various scenarios by adjusting variables and parameters, in order to identify possible bottlenecks or areas for improvement and predict how the process will behave over time, thereby enhancing decision-making and resource allocation.

Although BPMN and SD differ in their specific focus and methodology, they are both used to model processes in order to identify and understand their behavior and potential improvements. As such, both approaches can contribute to the design and management of processes in an organizational context [57]. While BPMN is limited in terms of dealing with different forms of "dynamics", SD captures nonlinear behaviors, accumulations, delays, and information feedback - factors often overlooked by existing modeling approaches. On the other hand, SD cannot be used to represent the flow control aspect of business processes; an important aspect where BPMN is particularly proficient.

In the world of business process modeling, creating a synergy between BPMN and SD is not only a challenge about theory but also practical usability. Being able to automatically transform BPMN models to SD models can significantly enhance the efficiency and effectiveness of process analysis and optimisation efforts. Therefore, we aim to extend the

work in [57, 58] and use it as a theoretical foundation for our study. The authors in [57] report on initial steps towards a more explicit combination of business process modeling and system dynamics. Furthermore, they provide an initial version of mapping between the constructs of these modeling languages as a first step towards a more complete integration between business process modeling and system dynamics. We use this version of the mapping between BPMN and SD components as the theoretical foundation for our research. Subsequently, in this study we intend to put that into practice through developing an experimental tool that facilitates the automatic mapping between the constructs of a BPMN and SD diagram.

The proposed automatic support intends to minimise the need for manual translation between the two notations, saving time and reducing the risk of errors. Furthermore, it allows organizations to leverage the strengths of both BPMN and SD, by making use of the visual (workflow) clarity of BPMN and the analytical power of SD to gain deeper insights into their processes. In other words, this automated solution intends to be a starting point in increasing the quality and efficiency of the integration between BPMN and SD, while also speeding up the process, making it more accessible to businesses trying to improve their business process modeling.

1.1 Identification of the problem area

The rising complexity of businesses and their environment, results in limitations of existing modeling approaches. There, for instance, we observe BPMN's limitations in terms of its capacity to deal with different forms of "dynamics". BPMN diagrams, which present a clear and comprehensive view of the sequence of actions and events in a process, can be used as the foundation for developing System Dynamics (SD) models to simulate and analyze the dynamic behavior of the process. Although BPMN and SD have a different focus and methodology, they are both used to model processes in order to identify and understand their behavior and potential improvements [57]. At the same time, the fundamental difference in the primary focus of these two approaches can be seen as a potential benefit of explicitly combining the two approaches [57]. As such, both approaches can contribute to the design and management of processes in an organizational context.

Existing modeling approaches often struggle to effectively handle the increasing complexity and dynamics present in modern business environments. In order to address this issue, different researches propose the integration of BPMN and SD to enhance the modeling and analysis of complex business processes, making it easier to simulate, analyze, and improve them. For example, in [28] it is suggested in a general sense, that dynamic simulation should be involved in modeling languages related to model-driven development, such as BPMN. The idea is further supported by authors in [56] and [37] who explicitly advocate the inclusion of a complementary System Dynamics perspective alongside Business Process Modeling Notation. This synergy is intended to make it easier to simulate hybrid processes at various levels of detail. The authors in [57] report that

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

this explicit combination of BPMN and SD could enhance the modeling and analysis of complex business processes while also taking into consideration other relevant (dynamic) characteristics of the context in which these business processes are used.

Now that we have defined our main problem area, we intend to ensure the practical usability of this integrated approach. In our study, we address this issue by extending the study of [57] and use it as a theoretical foundation to develop automatic support for transforming BPMN to Stock-and-Flow Diagrams (SFD). The output diagram can be further enhanced with variables to enable its execution as an SD model [57]. We use the version of mapping between the BPMN and SFD constructs in fig. 4.3 and proceed to develop an (experimental) tool to automate the mapping process, along with the associated modeling activities. By enabling a smooth transition between BPMN and System Dynamics, automated assistance intends to avoid and minimise the time-consuming manual mapping and model building that is needed. Moreover, this prototype can be the initial step in improving the accuracy and reliability of models, reducing the risk of errors and inconsistencies in the model-to-model translation process. Overtime, the overall goal of this approach is to improve the usability and practicality of integrating these two modeling languages in real-world business settings.

1.2 Justification of the study

In order to explain the justification behind this study, we need to explain that this work is part of a larger effort as described in [57]. There, the authors report on initial steps towards a more explicit combination of business process modeling and system dynamics.

Why this combination? – The theoretical foundation to combine BPMN and SD is based on [57, 58]. The methodological focus of BPMN and SD varies significantly; however, they are both used to model processes in order to identify and understand their behavior and potential improvements. System dynamics focuses on the analysis of complex and multifaceted systems and offers advanced simulation capabilities, whereas business process modeling primarily focuses on the (control flow of) business processes. SD captures nonlinear behaviors, accumulations, delays, and information feedback [16] – factors which are often overlooked by existing modeling approaches – enabling organisations to explore and analyze various scenarios, identify bottlenecks, and predict how a process will evolve over time [25]. However, the flow control aspect of business processes cannot be represented by SD, an important aspect at which BPMN is particularly proficient. As such, both approaches can contribute to the design and management of processes in an organizational context. At the same time, the fundamental difference in the primary focus of these two approaches points at a potential benefit of explicitly combining the two approaches [57].

Why is the tool beneficial? – It is the intention of this study that the integration between two modeling methods is supported by an (experimental) tool which implements the initial mapping of the constructs in fig. 1.1 and associated modeling activities. By enabling a smooth transition between BPMN and System Dynamics, automated assistance intends

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

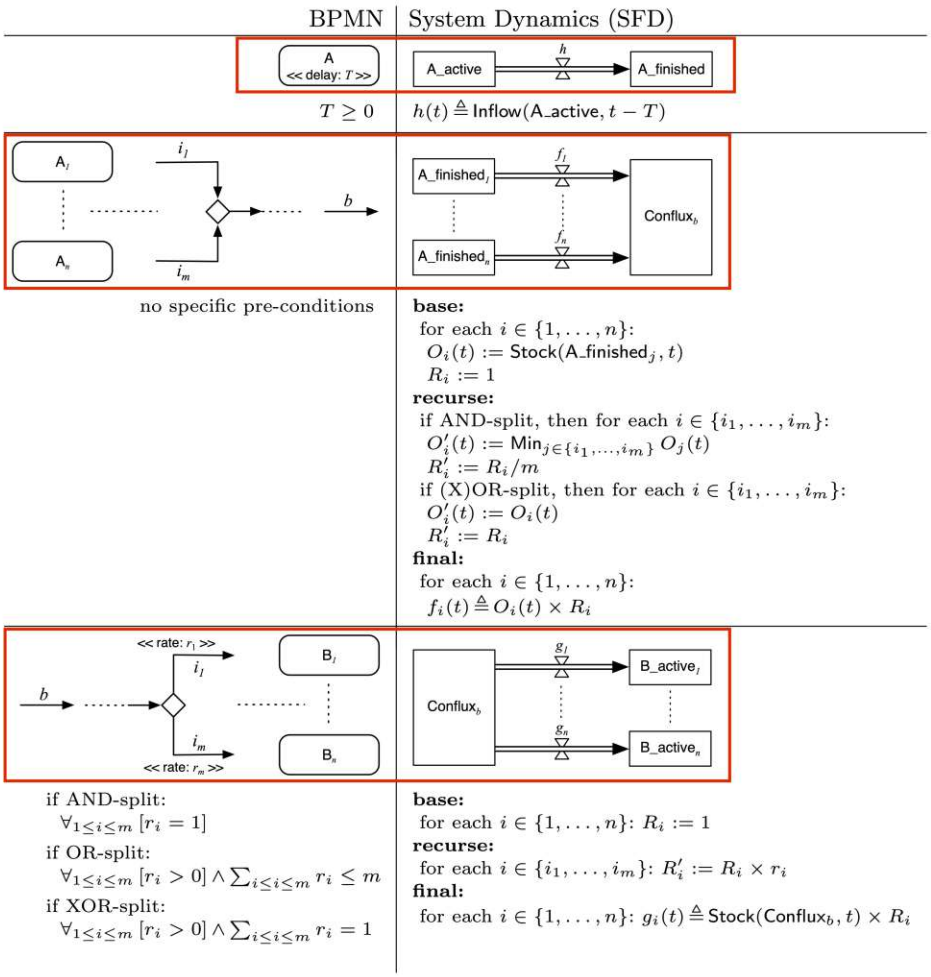


Figure 1.1: BPMN to SD Mapping in [57]

to avoid and minimise the time-consuming manual mapping and model building that is needed. After the transformation, other variables (mostly related to time and delay) and relationships (i.e. connectors) will be added manually to the SD model to enable the simulation runs (see [57]). The overall goal of this approach is to improve the usability and practicality of integrating these two approaches in real-world business settings.

1.3 Research questions

From the identified problem, we formulate our main research question for our study:

How can we design a tool that effectively transforms BPMN models to SFD models?

This research topic sets the foundation for a highly practical and solution-focused study, which focuses on the development of a transformation tool in the domain of process

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

modeling. Comprehensive knowledge of both BPMN and SFD is essential in order to develop a practical tool that is beneficial for users in this field.

We also identify the following more specific, manageable sub-questions in order to answer the specified primary research topic:

- **Concepts:** What are the existing BPMN and SFD concepts?
- **Tool:** What are the steps to develop an (experimental) tool that performs automatic transformation of BPMN diagrams to SFDs?
- **Design/ Mapping algorithm:** How can we formulate and implement an algorithm that optimally converts BPMN models to SFD models, as defined on the mapping provided in [57, 58]?
- **Evaluation:** To what extent does the existing Simplified modeling platform efficiently handle the defined model-to-model transformation steps, as measured by the mapping provided by [58]? What are the key technical limitations and challenges in implementing these transformations within the platform?

The main artifact is the prototype tool that modelers can use to create, edit and transform BPMN diagrams to SFDs. The results of this study can help progress knowledge in BPMN-to-SD model transformation and could enhance the efficiency of such model-to-model transformations.

1.4 Scope

The scope of this study includes:

- an overall comprehension of BPMN and SFD concepts,
- an overview of the development process of a transformation tool,
- a prototype that implements the provided guide, by extending the existing “Simplified modeling platform” to execute the BPMN-to-SFD transformation process.

The results of our research work can help progress knowledge in BPMN-to-SD model transformation and could enhance the efficiency of such model-to-model transformations. The main artifact is a prototype tool that modelers can use to create, edit and transform BPMN diagrams to SFDs. As a prototype, it is an initial iteration of automatically transforming a BPMN model to SFD, based on the mapping that is provided in [57] and further elaborated in [58]. The need for such a tool emerges from the potential improvements in process management that can be achieved by bridging the gap between these two modeling languages.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The prototype supports BPMN-to-SD model translations, minimizing manual effort and errors. In order to effectively evaluate the tool, this thesis provides both a literature review about each methodology's constructs and a practical implementation of the mapping via extending the Simplified modeling platform. In other words, as a result of this study we provide a web-based modeling environment that can automatically transform the elements and connections, as defined in the provided mapping in fig. 1.1, from a BPMN diagram into a Stock-Flow-Diagram (SFD).

Initially, we define the meta models for BPMN and SD (focusing on Stock-Flow Diagrams). Further, the mapping as summarized in fig. 1.1, and in [58], is the foundation to preparing the requirements for the prototype tool. The main body of the research consists in finding an efficient approach to build a model-to-model transformation tool that supports BPMN-to-SD transformation. As part of our evaluation, we implement this step-by-step guide by extending the Simplified modeling platform, which allows users to edit BPMN models on one diagram and generates its transformed stocks-and-flows diagram. The latter can then be exported as an xml file, which in further studies can be enhanced accordingly, in order to be simulated on other existing tools and used for the analysis of (complex) business processes. That being said, testing the tool in a practical environment is out of the scope of this research paper.

As a result of our study, we provide both the newly designed artifact (experimental tool) and the "design knowledge" that provides a better understanding of why the tool enhances the relevant application context, following the definition of Design Science in [68].

1.5 Research Approach

We use the Design Science Research (DSR) Framework as described in [68] as a guideline for our study. However, we do tailor the DS approach to fit our way. We focus on the prescriptive nature of design science; it emphasizes proposing solutions to practical problems on a certain domain [15] and creating artifacts [68], rather than merely describing or understanding phenomena. The outcomes of Design Science Research (DSR) cover both the newly developed artefacts and the associated design knowledge (DK), which offers a more comprehensive comprehension through design theories of how the artefacts improve (or complicate) the relevant application contexts [68]. In this regard, within this thesis, we provide the design knowledge (DK) for a better understanding of why the tool is a value-added to modeling business processes, what is an effective way to develop a tool that "works", as well as describe the functionalities of (experimental) tool itself.

Following the guidelines in fig. 1.2, we first define the added value of our solution (environment). Then, given our decision to extend an existing tool to implement the idea, we define a step-by-step procedure (adjusted from the team providing the existing modeling platform) to develop a model-to-model transformation tool. As a next step, we design the algorithm for transforming the models in the tool (design phase) and develop the prototype tool to evaluate the algorithm that we adjusted from [58] in the design

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

phase. The DS methodology emphasizes the iterative construction and evaluation of artifacts, which in our case takes place in between the steps of developing the new artifact and evaluation. The reliability and validity of the transformation algorithm is carefully and iteratively evaluated by running the transformation in the Simplified Modeling Platform, using examples of common BPMN patterns and its SFD transformation [53]. Relevant metrics have been monitored, as well as (BPMN and SD) experts' feedback was provided on the semantic explanation of the transformed patters. Their critical comments were used to adapt and improve the tool functionality, as well as support the added value to the business process modeling field. These steps were summarized in fig. 1.2, adopted from [68, 35].

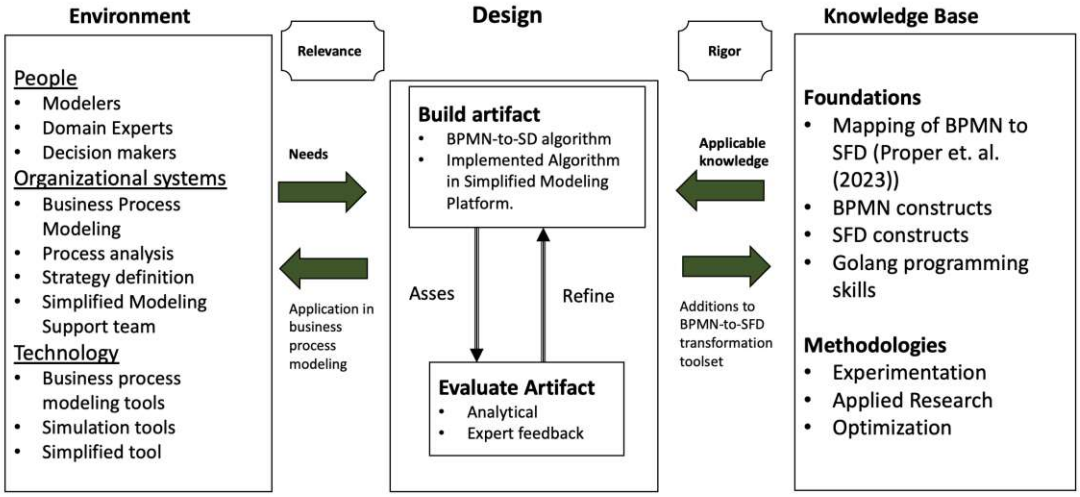


Figure 1.2: Design Cycle based on DSR Framework adopted from [68, 35]

As a baseline for our research approach, we take the DSR Methodology Process Model as described in [68] and adapt it to the following six steps: problem identification, define models' constructs and relations, requirements specifications to build a model-to model (BPMN-to-SD) transformation tool, design and implementation of transformation algorithm in a practical environment, evaluation of new artifact, and conclusions. This approach not only tackles specific technical difficulties but also enhances the overall understanding in the field of business process modelling, providing valuable insights that might potentially be used in other modelling challenges and scenarios.

In fig. 1.3 we provide an overview of the research steps we used in our study, adapted from [68]. In the first step, we identify the problem, define its scope, and describe the potential added value of the solution by reviewing existing literature. In step two, we provide an overview of existing grammar and constructs for each methodology and explain how we have defined them in our study. The output, meta-models of BPMN and SD, are a result of compressed literature review. As a next step, we specify requirements to build the BPMN-to-SD transformation tool(prototype). In step four, we provide the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

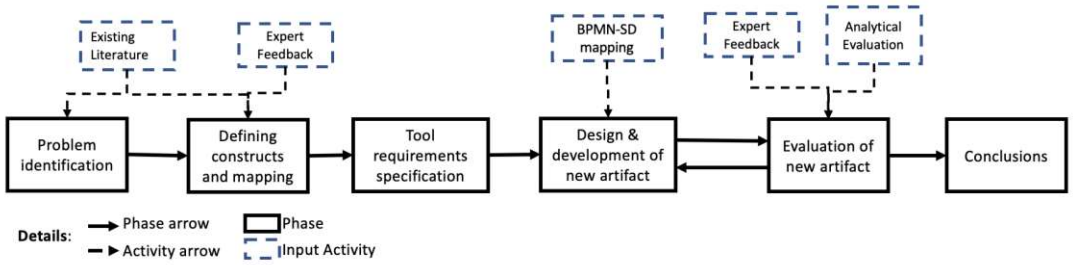


Figure 1.3: Research methodology steps adapted from [68]

generic algorithm to map BPMN-to-SD and implement it in a practical environment by extending the existing Simplified Modeling Platform [51]. We provide an explanation of the taken actions and explain the implemented algorithm. Step five, we evaluate the transformed diagrams produced by the tool based on the tool performance. Furthermore, we consult few examples of transformed patterns with BPMN and SD experts to get critical feedback on the semantic meanings behind the transformation. Note that here we iterate between the development and evaluation phases to improve the tool within the scope defined in the paper. Finally, as a last step we present our findings and suggestions for further improvement and the benefits this model to model transformation tool intends to bring.

1.6 Thesis Outline

In the chapters that follow, we go over the theoretical foundations of BPMN and SD, explain the specific challenges and complexities of this transformation process, and present the development and validation of the proposed transformation tool. More specifically:

- **Chapter 2 Literature Review:** In this chapter we investigate on existing knowledge body for BPMN and SD modeling languages, exploring on existing tools that support model-to-model (M2M) transformations. We provide a comprehensive understanding of the current state of research and existing methodologies.
- **Chapter 3 Meta-models and Modeling Constructs:** We examine the meta-models and constructs of BPMN and SD, providing a detailed investigation of their specifications and roles in the transformation process.
- **Chapter 4 Mapping BPMN to SD Components:** The mapping process is presented, including a discussion of the key focus areas and the detailed transformation algorithm from BPMN to SD.
- **Chapter 5 Automating BPMN-to-SD Transformation:** We outline the design, implementation, and validation of the transformation tool within the Simplified

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Modeling Platform. This includes the development process, and the results from evaluating the tool.

- **Final Chapter, Conclusions:** Summarizes the key findings, discusses the implications of the study, and suggests future improvements. We also highlight how the study contributes to advancing process modeling methodologies and addresses the complexities of modern business environments.

Each chapter builds upon the previous to develop a comprehensive understanding of BPMN to SD transformations, aiming to provide insights that enable organizations to enhance their process modeling and analysis capabilities.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Literature Review

In this chapter we review existing literature for the Business Process Management Notation (BPMN) and System Dynamics, in order to acknowledge the work of others in the relevant research area, as well as identify knowledge gaps. Our main purpose is to lay the foundation and provide a relevant justification behind our research. There exist several studies [28, 41, 21, 67, 56, 57] where authors "play" with the idea of combining two modeling methods, and have put light on the advantages of "getting the best of both worlds". This chapter consists of 3 main sections: we present existing literature on Business Process Methods (BPM), System Dynamics (SD) modeling and state-of-art for the existing model-to-model transformation tools. This intends to provide a clear overview of the methods' current and past state and the tools that support their transformation. Under section 2.1 we provide background on business process modeling notation and simulation. In section 2.2 we present an overview on system dynamics as a methodology of studying complex systems with a focus on the quantitative aspect of SD, namely Stock-and-Flow Diagrams (SFD). In the last section of this chapter we discuss the challenges and lack of specialized tools for the direct automatic transformation of Business Process Modeling Notation (BPMN) to Stock-and-Flow Diagrams (SFD).

2.1 Business Process Modeling Notation and simulation

2.1.1 Background on BPMN 2.0

Although Levitt first emphasized the value of business processes in [45] in 1960, it was not until the last 20 years that processes really started to take on significance in enterprise architecture [7]. Business process management (BPM) is being widely used by organisations nowadays to help them be prepared for and cope with the increasingly challenging conditions of modern business environments [54, 34]. As a result, a variety of methodologies, modeling techniques, and tools to support it have emerged [7]. The

first version of the Business Process Modeling Notation(BPMN) was written in 2001 and introduced later in 2004 as the standard notation for modeling business processes by [69]. Since then, BPMN has been evaluated in various ways by the academic community and has found broad support in the industry. As the BPMN approach has become accepted worldwide, BPMN tool support has also increased [54]. It has revolutionized the way enterprises represent and communicate their business processes [55].

According to White in [69], the primary goal of BPMN is to provide a notation that is understandable to all stakeholders in organizational processes, including business analysts who document or define business process models, technical developers who are in charge of creating IT solutions to support those processes, and users who will ultimately be controlling and managing the developed processes. Additionally, the business process models that were used by business people were technically different when compared to process models used by the systems that implemented and executed these processes. This required to manually translate the original process model to execution model; therefore, making these translations error-prone, as well as limited process owners in understanding the evolution and performance of the process they modeled. This gap was an important factor in the development of BPMN, to build a bridge from a visual notation to execution languages [4].

BPMN primarily offers a control-flow focused graphical notation for business process modeling. BPMN is based on the revision of other notations and methods, such as UML Activity Diagram, UML EDOC Business Process, IDEF, ebXML BPSS, Activity-Decision Flow (ADF) Diagram, RosettaNet, LOVeM and event-driven process chains (EPS) [53]. Version 1.0 of BPMN was firstly published in 2004 and two years later, it was taken over by Object Management Group (OMG) for further maintenance. This was followed by BPMN 1.1 [52] in 2008 and the current version BPMN 2.0 [53] in 2011.

Main focus points for BPMN 2.0 development were [4] standard meta-model and serialization format for BPMN, standard diagram exchange format, comprehensive notation for cross-organizational interactions, additional elements for modeling, standard execution semantics for BPMN etc.

BPMN 2.0 is composed of three types of diagrams: business process diagram (BPD), conversation diagram, and choreography diagram [53, 4]. Each of them has a distinct function in offering insight and clarity into various facets of how processes are planned and carried out inside a company. In our study we focus only on BPDs. Business process diagrams (BPDs) allow enterprises to model end-to-end business processes, in order to determine whether their processes have anomalies, inconsistencies, inefficiencies and, consequently, whether there is potential for improvement.

BPMN 2.0 categorizes its constructs into five distinct categories (flow objects, data, connecting objects, swimlanes and artifacts). Furthermore, the BPMN structuring elements — groups, pools, lanes — make it easier to visually distinguish different areas of a BPMN diagram. Given the large number of BPMN concepts, 3 levels of use have been defined [53].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Level 1: Descriptive modeling – focuses primarily in documenting the process flow. Widely used to represent as-is and to-be models. This level is where BPMN is most commonly used [4].

- Level 2: Analytical modeling – enables more accurate modeling by including exceptions and events. Used to better understand and measure how well a process is performing (qualitative aspect) and specific metrics (quantitative aspects).

- Level 3: Executable modeling – enables transformation of visual models into XML-based specifications to support automatic activation of the models.

The constructs of BPMN 2.0 are treated in more detail in section 3.1.

2.1.2 BPMN and simulation

Business Process management (BPM) is a broad concept that includes business process modeling and business process simulation as two important techniques. Business process modeling deals with visually representing the steps, activities, and relationships within a business process. Business process simulation takes it a step further, by using models to simulate the actual operation of the processes over time.

In [32], the authors evaluate how suitable is BPMN as a business simulation language by using the Agent-Based Discrete Event Simulation Ontology (ABDESO) and extending the Discrete Event Simulation Ontology (DESO). Only a core part of BPMN 2.0 elements were chosen to be part of the study, therefore the results are preliminary. In the end, the results showed that there is 70% lucidity (ambiguous elements), 60% completeness (missing concepts) and 32% laconicity (redundant elements) in BPMN [32].

In order to apply efficient analysis techniques that are available for Petri Net models, [24] map BPMN models to Petri Nets in [24]. By exporting the BPMN model to an XML file, they implemented a tool to translate BPMN models to Petri Nets via the Petri Net Markup Language (PNML). The study was limited to the control-flow perspective of BPMN and the order in which activities and events are allowed to occur. By doing this, they were able to evaluate the semantic accuracy of BPMN models as well as explain the fundamental constructs of BPMN.

BPMN was initially created with the focus on standardizing the graphical depiction of business process flows, without any consideration for simulation. Hence, a set of "extensions" to the BPMN language have emerged in order to simulate process models created in BPMN [54]. While there are a number of BPM tools that support BPMN and include simulation features, there are still several limitations on how process models may be simulated [54, 55].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.2 System Dynamics Modeling and Simulation

In this section we provide a brief overview of System Dynamics, emphasizing its usage in complex system analysis through simulation. SD is a methodology for studying complex systems that involves creating mathematical models, in order to simulate the behavior of the system over time. It was developed by Professor Jay Forrester at the Massachusetts Institute of Technology (MIT) in the 1950s [29]. More on SD history can be found in [31] and [29]. In a nutshell, System Dynamics is composed of two notions: 'system' - a collection of ordered, interconnected parts [33] organised for a purpose [22] and 'dynamics' - refers to change over time.

The majority of the data that experts have access to in SD, is qualitative rather than numerical [46]. When it comes to the modeling process, researchers like Richardson and Pugh (1981) [62], Randers (1980) [61], Roberts et al. (1983) [9], Wolstenholme (1990) [70] and Sterman (2000) [66] have established a set of activities for modelers to follow, in order to build their SD models. The arrangement of activities varies from three to seven stages (fig. 2.1): in [70] Wolstenholme breaks down the structure in three stages, in [61] Randers depicts 4 stages, in [66] Sterman gives 5 stages, in [9] visualizes the process in 6 stages and lastly, in [62] Richardson an Pugh conceptualizes the modeling process in 7 steps.

Randers (1980)	Richardson and Pugh (1981)	Roberts *et al.* (1983)	Wolstenholme (1990)	Sterman (2000)
Conceptualization	Problem definition	Problem definition	Diagram construction and analysis	Problem articulation
	System conceptualization	System conceptualization		Dynamic hypothesis
Formulation	Model formulation	Model representation	Simulation phase (stage 1)	Formulation
Testing	Analysis of model behavior	Model behavior		Testing
	Model evaluation	Model evaluation		
Implementation	Policy analysis	Policy analysis and model use	Simulation phase (stage 2)	Policy formulation and evaluation
	Model use			

Figure 2.1: The system dynamics modeling process across the classic literature [46]

No matter how differently the researchers organize the activities, they all view them as components of an iterative process where the modeler tests a dynamic hypothesis. The latter represents a feedback theory or causal structure generating a series of behaviors through time, allowing the issued actors to analyse the situation and build or revise their policies. The ability of this approach to depict a system's changing state over time is by far its strongest point [46].

In [41], Kožusznik defines the models in System Dynamics as systems of interconnected feedback loops and stocks-and-flows of resources. Understanding the cause and effect in a system enables decision-makers to analyze, sort out and explain certain changes. That being said, SD is considered as a great means of exploring and finding knowledge gaps [23, 62]. While mental models are good at analyzing the basic actions that a system is composed of, they are not able to comprehend the dynamic results of these actions [26].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

SD models are used in a wide range of fields, including business [47, 6, 8], economics [59, 60], engineering [19, 48], environmental science [36]. They can be used to model and simulate complex systems such as supply chains, financial systems, ecological systems, and social systems. The key strength of system dynamics is its ability to capture the complexity of a system and to represent the inter dependencies and feedback loops that exist within it, which can help to simulate and reveal unexpected or counter intuitive behaviors. By doing so, it allows decision-makers to test different policies and strategies in a safe and controlled environment, without the risks and costs associated with real-world experimentation.

In [12, 41, 10] the studies reveal several advantages to using SD for simulation, such as:

- *Including the "time" aspect while modeling complex systems*: SD is convenient for modeling complex systems that involve feedback loops, non-linear relationships, delays, and other dynamic interactions.
- *Holistic perspective*: SD encourages a holistic perspective that takes into account the interconnections between different elements of a system, which can help decision-makers to identify unintended consequences and better understand the long-term implications of their decisions.
- *Scenario testing:* SD allows decision makers to test different scenarios and policies in a safe, controlled environment. This can help decision makers to identify the most effective policies and avoid costly mistakes.
- *Communication:* SD can be a powerful tool for communicating complex concepts to stakeholders and decision makers. Visual representations of the system dynamics can help to make the information more accessible and understandable.
- *Flexibility:* SD models can be adapted and modified over time as new information becomes available or as the system itself evolves. This can help decision makers to stay responsive to changing circumstances and make better-informed decisions.

2.2.1 SD Quantitative Modeling - Stocks and Flows

In System Dynamics, there are two fundamental diagrams used to represent the dynamic interactions and accumulations within a system: Causal Loop Diagrams (CLD) and Stock-Flow Diagrams. CLDs are qualitative in nature, while SFDs are quantitative. While CLDs depict variables and how they influence each other, SFDs can be considered as a materialization of CLDs with a framework to easily set up differential equations. The quantitative aspect of a Stock and Flow Diagram entails sets of equations or functions that are input into the model resulting into simulations. Graphically, the equations display the relationships between stocks and flows that contain underlying information of the model. Moreover, SFDs offer a basis for rigorous takeaways of dynamic behavior because the variables and links can be used to explain a wider range of counter intuitive dynamic phenomena than CLDs [43].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Stocks and flows are two basic building blocks in SD modeling [25]. SFD uses a flexible notation to represent the components and relationships of a system in a visual and intuitive way. This notation is often used in conjunction with software tools, such as Vensim, Stella, or AnyLogic, which provide a visual interface for building and simulating models. SFD can be used to model a wide range of systems, including social, economic, and environmental systems.

Stock and Flow Diagrams refer to a visual representation of a complex system as applied in system dynamics by quantitatively analyzing the structural components of a system, to observe changes that result in a certain behavior. As such, stocks are defined as system reservoirs that are quantified at a particular moment in time. Stocks are known to create delays in a system. As defined in [12], a flow is a variable that causes a change in the stock quantities. In other words, a flow can be defined as the rate of change of stocks; the measurement of flows is per unit time. The value of a flow is determined by the stocks in the system and also by external influences. SFDs are very useful when simulating a model because the variables involved are well-defined and it involves a clear and visual depiction of stocks (accumulations) and flows (movement) within a system over time.

Delay is an important factor in the flow of information and materials, as well as the overall performance of a system. In [11] the author emphasizes the following aspects of "delay" in SFDs. (1) It is essential to acknowledge and consider the impact of delay while making decisions. (2) Delays can be a factor in determining success or failure. (3) Consider delay as a significant factor in calculating the value of change. Stocks are represented as quantities, and flows as quantity per unit time.

2.2.2 SD and simulation

Simulation is the imitation of the operation of a real-world process or system over time [13]. In [5] Van der Aalst states that simulation is particularly appealing due to its versatility, lack of limitations, and being able to provide results that are very simple to grasp. Many researchers have provided their own definition of the concept of "simulation". Shanon [65] defines simulation as "the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system." Therefore, it can be said that the dynamic behavior of a system can be called simulation.

In order to examine how a process may work in practice before it is constructed, it can be run through a simulation. Also, process simulators provide modelers the freedom to be creative and experiment with a variety of situations and scenarios until they get their desired results, without interfering with the actual production cycles. Models used for simulation may be simple or complex. One may build sophisticated models of business processes with a high level of accuracy using several modeling and simulation technologies [14]. Although a highly accurate model would seem to be ideal, this does not imply that the model must be complicated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Due to the counter-intuitive nature of complex systems, the human mind is incapable of precisely tracking the dynamic behaviour of such systems. In [11], the authors argue that an SD simulation model can offer improved understanding and deeper insights into these systems. The ultimate objective of system optimisation is to establish an optimal management and control policy that satisfies specific criteria and restrictions. The use of a simulation model is crucial for this type of system optimisation. Policy planning requires analysing the alternatives in order to choose the optimal course of action for enhancing system performance. The SD simulation model can help with management and developing policies, in addition to serving as a computer laboratory for policy analysis.

2.3 Existing tools that support model-to-model transformation

The process of transforming input models that conform to a source meta-model into output models, which also conform to a target meta-model, is technically supported by model-to-model (M2M) tools fig. 2.2. A model transformation definition presented in a model transformation language specifies the process of transforming one or more input models into one or more output models. If the transformation definition's language is rule-based, it comprises a collection of transformation rules. The transformation engine uses the model transformation definition to generate output model(s) from input model(s).

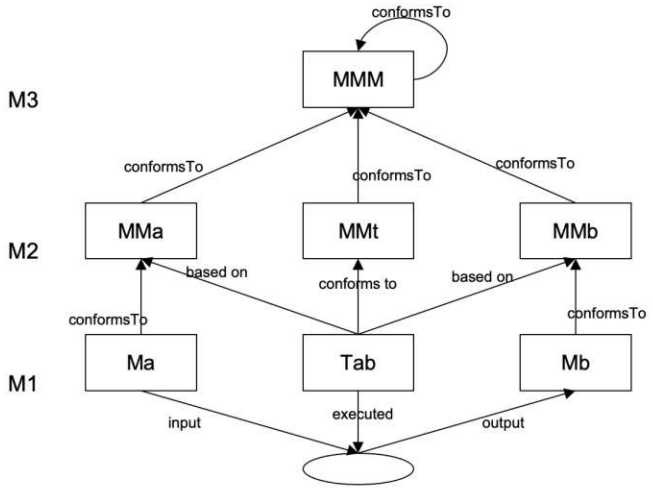


Figure 2.2: Model-to-model transformation in MDE

In [39], the authors classify 60 existing meta-model based M2M tools based on their transformation approach fig. 2.3. This approach involves the specific language constructs or mechanisms used to define and implement transformations, which fall into four basic

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

categories: relational/declarative, imperative/operational, graph-based, and hybrid. The M2M transformation that we implement within this study, concerns relational/declarative methods fig. 2.3. The focus in relational/declarative methods is to identify the input constructs that need to be converted and the corresponding output constructs. By defining predicates[1] or input-output constraints, this is accomplished. Additionally, rather than stating clearly how the transition has to be carried out, the focus lies in establishing connections between the components of the input and output models [39]. In our study, we use the declarative mapping between BPMN and SFDs as described in [58, 57] and continue to implement it in a M2M tool. More in chapter 4.

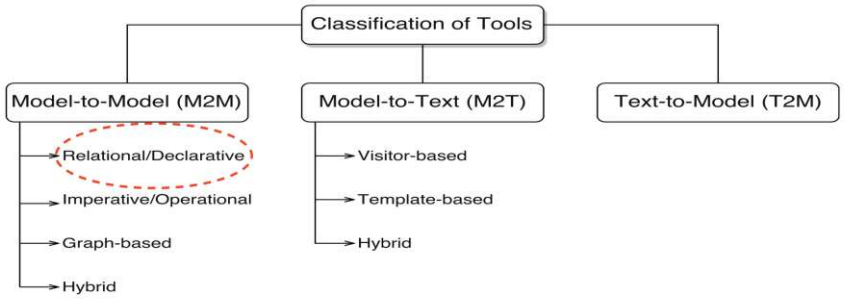


Figure 2.3: High-level classification of transformation tools from [39]

Existing BPMN tools, such as Bizagi, ARIS, and Signavio, are useful in modeling business processes and primarily focus on workflow design, task allocation, and business rule management [40]. These tools are useful for process optimisation and organisational efficiency because they clearly illustrate the actions and decisions that must be made in a sequential order. However, they do lack features for direct translation to SD models or SFD representations [27].

SD softwares like Vensim, AnyLogic, and Stella Architect are proficient in creating quantitative models with SFD to represent accumulations and interactions in complex systems like used in [42]. These tools are extensively used in various fields, including healthcare, environmental management, and supply chain analysis, offering simulation capabilities for understanding system behavior over time.

While BPMN and SD models offer comprehensive insights into business processes and system behaviors, the direct automatic transformation from BPMN to Stock-and-Flow Diagrams (SFD) is challenging and currently lacks specialized tools dedicated to this specific conversion. The absence of dedicated tools specifically focused on BPMN to SFD transformation stems from the conceptual and semantic differences between the two modeling languages. The transformation involves translating qualitative business processes into quantitative system dynamics, demanding complex algorithms to preserve

[1] A predicate is a statement that expresses a condition or a relationship between entities. It helps define the rules for transforming elements from one model to another.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

both the visual structure and the underlying meaning [42]. The direct transformation from BPMN to SFD is a complex interdisciplinary challenge due to the fundamental differences in their structures.

Currently there are existing tools [54](i.e Bizagi, BIMP, Visual Paradigm, BPSim) with an approach to BPMN that also provide process simulation. However, they provide simulation techniques that are mainly focused on the straight-through process flow and not on the other facets that System Dynamics can offer - especially with the top down causal loop analysis. SD simulation provides a holistic view to understand where is the added value created in the process flow or how it is realized and consumed by the services in an organisation. As until now, businesses and researchers often resort to manual interpretation and adaptation or might use a combination of multiple tools to achieve this transformation, which emphasizes the need for dedicated and automated BPMN-SFD transformation tools even more.

2.3.1 ATL model transformation language

Eclipse ATL (Atlas Transformation Language) is a model transformation language and toolkit created by the Eclipse Foundation, used primarily by software developers and designers, particularly those working with model-driven development (MDD) and model-driven engineering (MDE) approaches. Within MDE, ATL offers methods to generate a set of target models from a set of source models [1]. The ATL Integrated Development Environment (IDE) is built on the Eclipse platform and offers various common development tools such as syntax highlighting and a debugger. ATL is a hybrid language that combines declarative and imperative constructs.

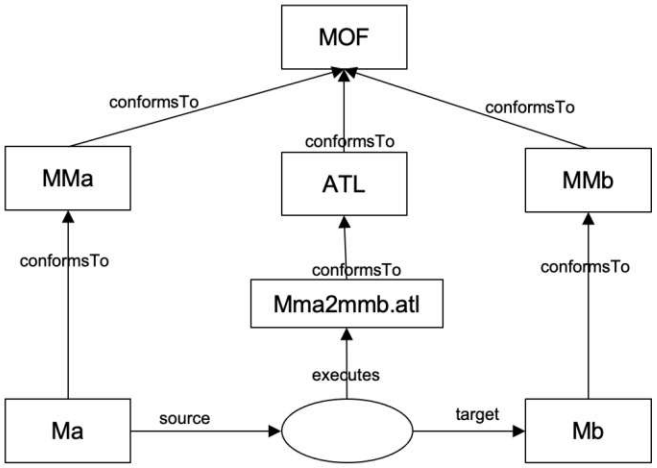


Figure 2.4: Overview of ATL transformation approach [38]

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

ATL follows the transformation approach provided in fig. 2.4. The source model Ma is transformed into the target model Mb using the transformation specification mma2mmb.atl written in the ATL language. These three components conform to the MMa, MMb, and ATL metamodels, respectively. All metamodels adhere to the meta-meta model, known as MOF within the OMG standards framework. Metamodels of source and target models can be expressed in XMI.

ATL's specialised language is designed for transforming models, to make the conversion process easier. The declarative method enables developers to specify transformations at a high level, improving comprehension and maintenance.

Using the ATL transformation technique to implement transformations (in particular BPMN-SD) provides benefits as well as drawbacks. There are numerous potential disadvantages linked to using Eclipse ATL for defining a BPMN-to-SFD transformation program.

Learning Curve: ATL has a challenging learning curve, particularly for individuals unfamiliar with model-driven programming or transformation languages. Comprehending the syntax, semantics, and best practices of ATL may necessitate a substantial amount of time and effort. Translating all BPMN constructs accurately into SFD may necessitate complex and comprehensive transformation rules, potentially leading to increased development and maintenance necessities.

Tooling Support: Eclipse offers support for ATL development through plugins and connection with other modeling tools, however the tooling support for ATL may not be as comprehensive and user-friendly as for other transformation languages. Furthermore, the Eclipse platform can be quite prone to malfunctions and unreliable at times. There are times the program freezes, and it's not uncommon to encounter errors without clear explanations of their causes. This could affect productivity and development efficiency.

Performance: ATL transformations can result in performance overhead, particularly with large or complex models. Complex transformation rules or operations that require a lot of resources can result in extended execution duration and increased memory use. Therefore, higher levels of complexity within BPMN diagrams, tends to correspond with a decrease in transformation performance.

Lack of bidirectional transformations: ATL primarily supports unidirectional transformations, meaning it may be less suitable for scenarios requiring bidirectional synchronization between BPMN and SFD models. Integrating bidirectional transformations in ATL might require extra effort and alternative solutions.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 3

Meta-models and modeling constructs

In this chapter we present the meta-models and constructs of Business Process Modeling Notation & Stocks and Flow Diagrams as used in our study. Every modeling methodology has a set of constructs that define its vocabulary and entities. Constructs define what can be represented with that certain method and determine the way that method interprets and perceives the real world [68]. That being said, the modeling methods that we include in the scope of this study, use their own specific rules and visuals to model certain aspects of a system. In order to ensure consistency and coherence in the models created using the constructs of each modeling method, meta-models specify the kinds of structures that can be used, their relationships, and the rules controlling their usage. Therefore, before delving further into the implementation process of the transformation tool, in this chapter we introduce and explain the constructs and meta-models of both methods outlining the fundamental principles that guide them.

3.1 BPMN constructs and meta-model

BPMN has been intentionally limited to supporting only those modeling concepts relevant to conventional business processes. As such, while relevant, elements of extended business process modeling such as linking processes to business goals, executing Value-Oriented Process Modeling, conveying aspects of measurements and reporting, or creating rule sets (business, application, etc.) are not the central focus of BPMN [63].

In the formal BPMN 2.0 specification document by OMG [53], it is explicitly stated that the primary goal for creating BPMN was to provide an easy-to-understand notation for visualizing Business Process models, while also providing the semantics and underlying mechanisms to handle even complex business processes. In order to handle these two

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

conflicting requirements, the graphical aspects were organized into 5 specific categories. This way, the basic types of BPMN elements and diagrams can be easily understood by any reader of any background. In any case, additional information and variation can be added to the basic categories, in order to support the complexity in modeling business processes, while still maintaining the basic look and feel of the diagram. The 5 basic categories of the elements are (1) Flow Objects, (2) Data, (3) Connecting Objects, (4) Swimlanes, (5) Artifacts.

1. Flow objects: the main visual elements that define the behavior of the business process.
    - **Events**: represent occurrences that happen during the course of a business process. They can trigger or result from activities, indicating the start, intermediate points, or end of a process. In this study we include start and end event; intermediate events are out of the scope.
    - **Activities**: represent the work or tasks that are performed within a business process.
    - **Gateways**: are used to control the flow of a process. They determine which path the process should take based on certain conditions or events. We use the concepts of Join- & Split-trees for the Gateways in the implementation of BPMN-to-SD transformation, and do not dive into the details for XOR, OR, AND Gateway types.
2. Data: is represented as data objects, data inputs, data outputs, data stores. Data elements are out of the scope of our BPMN definition in the Simplified tool.
3. Connecting Objects
    - **Sequence Flows**: depict the sequential order of actions in a process by linking flow objects, showing the flow direction from the source to the target element. In our BPMN-to-SD transformation we only use Sequence Flows.
    - Message Flows: represent the exchange of messages between different participants in a business process. They are used to show the communication or information flow between various entities involved in a process. Message flows are out of the scope of our implementation.
    - Associations: are used to connect additional information or artifacts to BPMN elements without affecting the flow of the process. They provide a way to attach text annotations, data objects, or other artifacts to certain elements in the diagram. Associations are out of the scope of our implementation.
4. Swimlanes can be categorized as below (not included in our implementation):
    - Pools: represent high-level organizational entities or participants in a business process. They can be thought of as separate, independent organizations or systems that participate in a collaborative process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Lanes: are subdivisions within a pool, used to organize and categorize activities based on roles, departments, or functional areas. They are contained within a pool and help in visually grouping related activities.

5. Artifacts can be defuned as the following (not included in our implementation):

   - Group: used to visually group and highlight a set of related BPMN elements.
   - Text Annotation: used to provide additional textual information or comments about a specific BPMN element. Annotations help in adding details, explanations, or comments to clarify the meaning of a process or specific elements within it.

To summarize, as part of our research work, we only include the fundamental concepts of BPMN; namely start event, end event, activities, gateways, sequence flows.

The meta-model in fig. 3.1 outlines the types of constructs that were used to create BPMN models in our tool, how they relate to one another, and how they assist in displaying the business process.

The BPMN constructs were defined using the self-defined notation script of *Simplified modeling platform* [51] as a first step towards the development of the transformation tool.

3.2 Stock and Flows Constructs and meta-model

This section provides an introduction to the concepts of the stock and flow language (initially named level/rate language [30]).

As part of our study, we have implemented the following essential components from SFD in our transformation tool: stock, flow (inflow/outflow), source and sink. We leave converters (auxiliary/constant) and connectors out of the scope of our implementation.

The following explanations of the SFD constructs refer to textual descriptions of the Stock and Flow constructs found in [30, 44, 20, 67, 43].

The SFD components can be grouped into node and edge types [20]. As such we describe as following:

Node types

Stocks (also known as levels) describe the state of the system and are defined as containers/reservoirs; they are visually represented as boxes (rectangle). The magnitude of stocks adjusts over time due to in/outflows [43].

Sources and **sinks** are considered the boundaries of the system model. Source illustrates the systems outside the model's bounds, and is located at the start of the flow arrow. The sink is the point at which flows end outside the system, and is positioned at the tip of the arrow. Both source and sink are represented graphically as cloud symbols.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

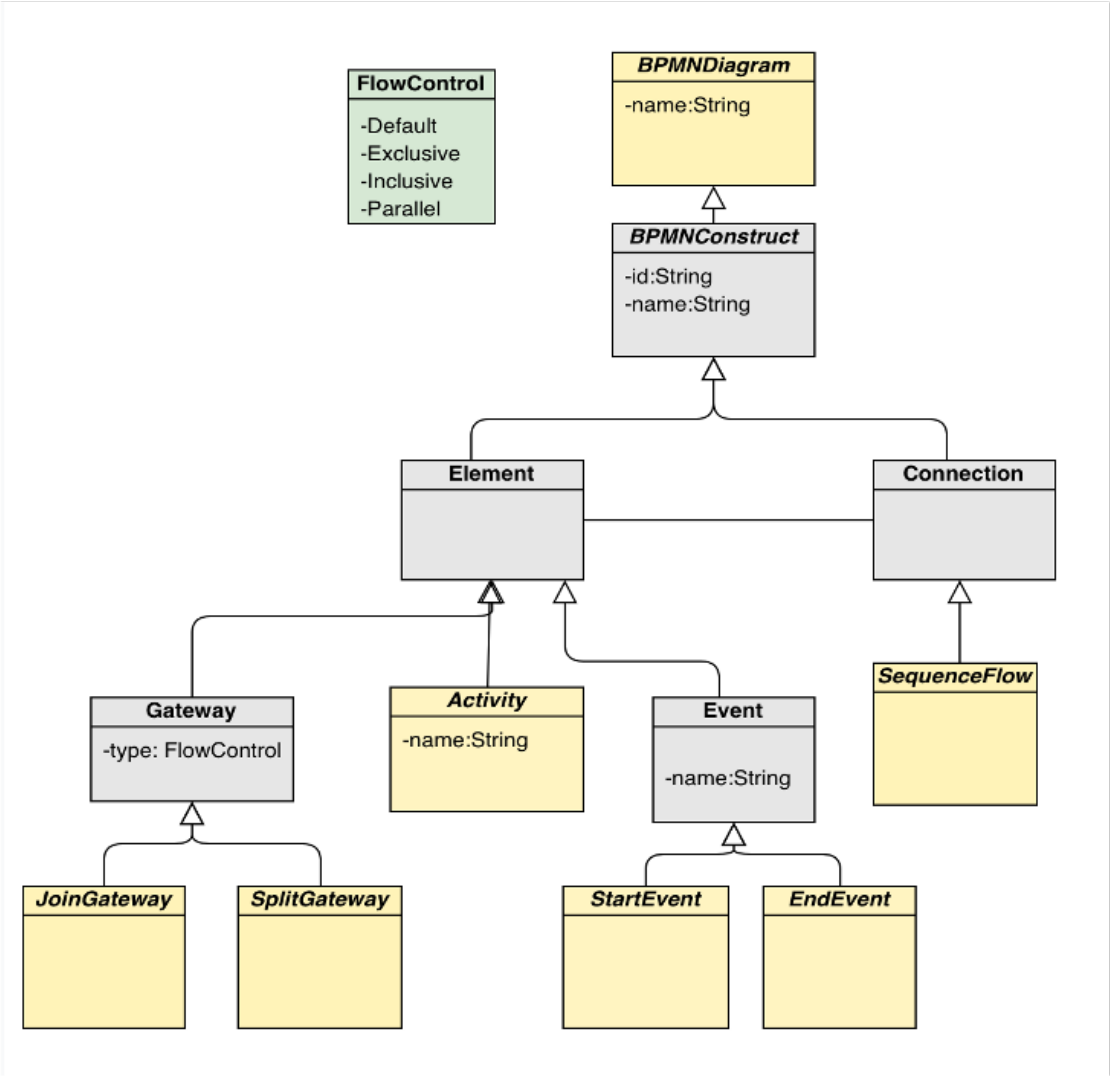


Figure 3.1: Simplified BPMN meta-model

Rates are responsible of controlling the flow of "things" among stocks in a system. Rates are presented via decision functions that decide the amount of flow based on information about stocks in the system model.

Converters can be constants (fixed quantity) or auxiliaries (variable quantity). **Constants** are state variables. They do not change or their change is negligible and can be assumed constant throughout the time scope of the model [20]. **Auxiliary variables** are information notions that can be in the form of functions or values applied to stocks, flows or other converters in the model [44]. Auxiliaries and constants are represented by small circles.

Edge types

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Flows control the rate of accumulation to (inflows) and from (outflows) the stocks. One can think of flows as a pipeline with a valve to control the accumulation rate that goes into or out of stocks. Visually it consists of two solid lines with a direction arrow in one end and a valve in the middle. Flows originate from a *source* and terminate at a *sink*. The valve can be represented as a circle with a lever on top, or alternatively, as two triangles positioned on top of each other, with a shared vertex. In this study, we visualize the valve using the triangles.

Connectors (information links) share information between converters and flows. Visually they are represented by arched arrows.

A summary of all the discussed stock and flow building constructs is presented in fig. 3.2.

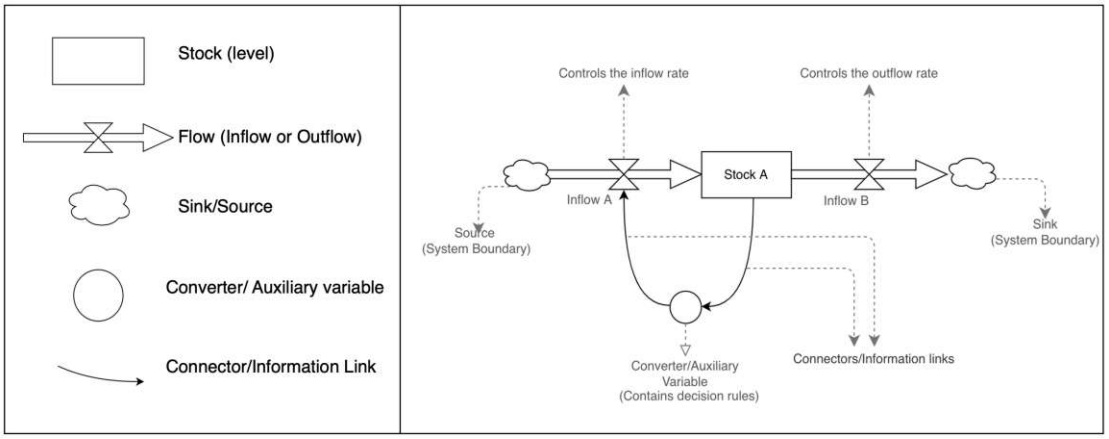


Figure 3.2: Overview of the key components in SFD

Before we conclude our description on SFD constructs, we present a summary of the design rules for SFDs.

Stocks connect to flows that are either directed towards or away from the stock. A stock either succeeds or precedes one or many flows [30]. This translates to a cardinality of (0, 1) for stock and (0, n) for flows [20]. Moreover, stocks can only be affected by flows. To explain in other words, connectors/information links cannot point directly to a stock, but can point away from a stock.

Flows are influenced by variables in the model(can be stock or converter) through the connectors. This is how the values in the in/outflows are able to change the amount of stocks. In order to define a rate there should be at least one connector to influence the flow, otherwise the flow becomes inactive. To be more precise, flows can be influenced by stocks, converters or exogenous variables but they cannot directly affect converters, exogenous variables or other flows [67].

Converters: Converters can be constants or auxiliary variables. They can be influenced by other elements in the model and are generally used to influence flows, other converters or exogenous variables.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

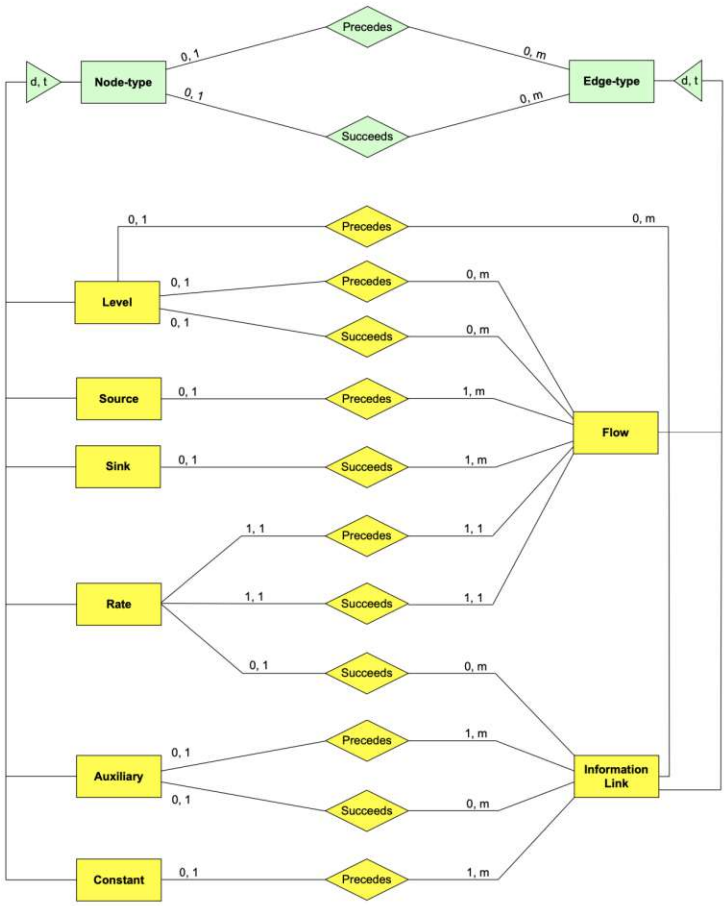


Figure 3.3: Abstract Syntax of the Stock/Flow(Level/Rate)-Language [20]

Sink and source are points in the system model where a flow terminates or originates from.

Connectors/Information Links can input information into or out of converters, flows, exogenous variables. However, they can only extract information from the stock (recall design rule for stocks: stocks can only be affected by flows).

In [20] Burmester and Goeken provide an Entity-Relationship (ER) model of the abstract syntax of the Level/Rate-Language (Flow/Stock-Language) in fig. 3.3. The reading order of the ER model is from node to edge types. Moreover, the naming approach (precedes, succeeds) of the relationship types refers to the direction of the edges (i.e "Level(Stock) precedes Flow means the flow-edge is directed away from the level-node. More details can be found in [20].

Formal definition of Stock-Flow-Diagrams (SFDs)

Though stock and flow diagrams are graphical representations, they can be defined in mathematical expressions to describe the underlying dynamics.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The value of stock S at a specific time t is represented as $\text{Stock}_t(A)$ - representing the accumulated quantity at a time t. We can explain *flows* through *inflows* and *outflows*. Let's say $\text{Flow}_t(A \to B)$ is the amount of flow from stock A to B, through flow A to B. Then, total inflow in stock A via all possible incoming flows at time t is expressed as $\text{Inflow}_t(A)$, and total outflow is expressed as $\text{Outflow}_t(A)$ (see [66, 67]). So we have:

$$\text{Inflow}_t(A) \triangleq \sum_{B \to A} \text{Flow}_t(B \to A)$$

$$\text{Outflow}_t(A) \triangleq \sum_{A \to B} \text{Flow}_t(A \to B)$$

The stock accumulation is denoted as

$$\text{Stock}_t(A) = \text{Stock}_{t_0}(A) + \int_{t_0}^{t} \text{Inflow}_u(A) - \text{Outflow}_u(A), du$$

at any moment t$\geq t_0$, where t_0 is starting time. As a result, the change in flow can be denoted as:

$$\frac{\text{d}(\text{Stock}_t(A))}{dt} = \text{Inflow}_t(A) - \text{Outflow}_t(A)$$

Meta-model

As mentioned previously in this chapter, we only use the base elements to implement in our transformation tool (stock, flow (inflow/outflow), source and sink). In fig. 3.4 we present the metamodel we use for the implementation in our study, conforming to the mapping provided in [57], also approved by our selected SD expert (Quan Zhu).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

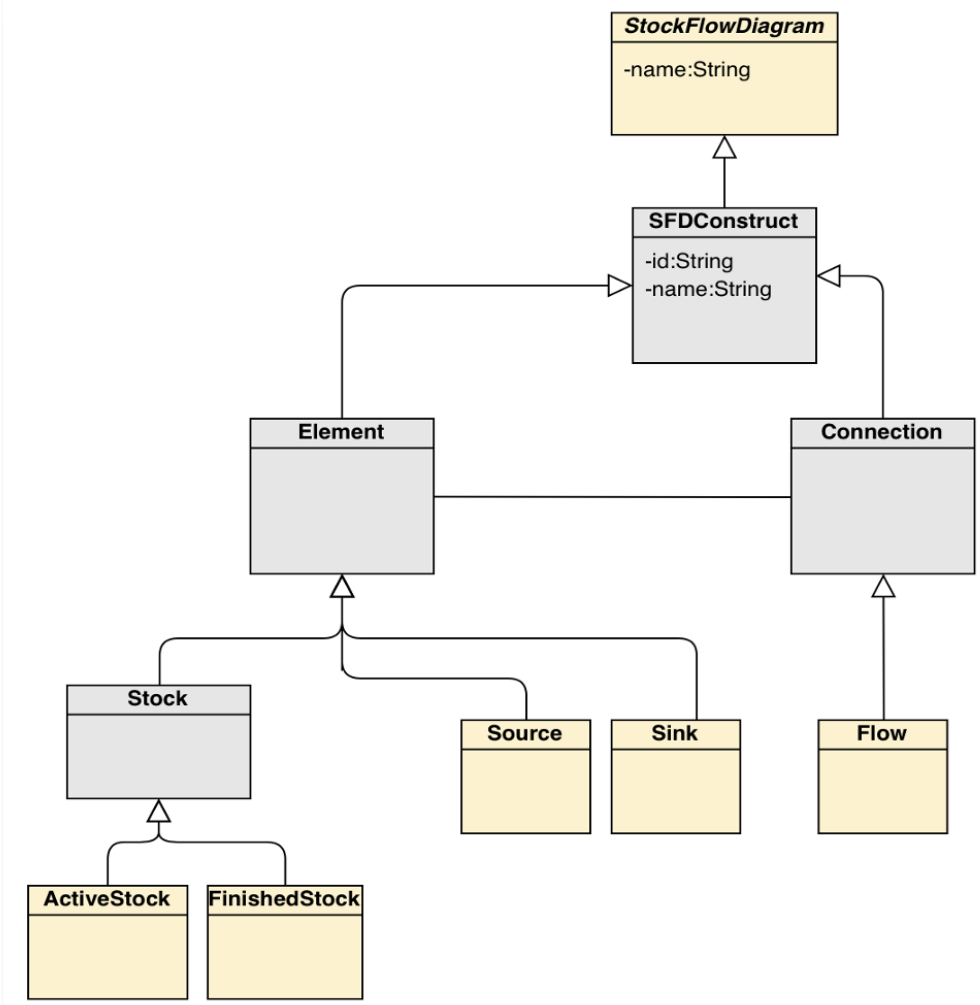


Figure 3.4: Simplified SFD meta-model

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Mapping BPMN to SD components

To map BPMN components into Stocks and Flows, we consider the work in [57, 58] as a foundation to our study. The authors in [57] propose a step-by-step guide, as an integrated procedure to analyze complex business processes by using the BPMN and SD models, visualized in fig. 4.1. They highlight the potential advantages of integrating these two methodologies, notably in the modeling of complex business processes that involve dynamic environmental factors. The comprehensive guide comprises of a sequential process that involves: outlining the scope and goals, *constructing a BPMN diagram, mapping it manually into an SFD,* incorporating logic and variables, verifying the SFD model, simulating process behaviour, identifying areas for enhancement, and conveying outcomes to relevant parties.

Furthermore, the authors propose several further actions as future work, one of which is *developing tool support*. The latter is our focus on this study, as an initial step to automate the transformation of BPMN diagrams to SFDs. The steps that concern the scope of our study include step 2 and 3 in fig. 4.1 presented by the authors as a guide to transform BPMN diagrams into SFDs. By extending the existing Simplified Modeling Platform, we support the creation of BPMN diagrams and implement the transformation algorithm to automatically transform into a Stock-and-Flow diagram.

In the preceding chapters we have defined the modeling methods, outlined their constructs and language rules in order to provide a better theoretical understanding of the synergy between BPMN and SFD, though their primary focus is very different. However, in this chapter, we provide an overview of the BPMN-to-SD mapping based on [57, 58] and explain in detail the (adjusted) mapping we implement in the Simplified Modeling Platform.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

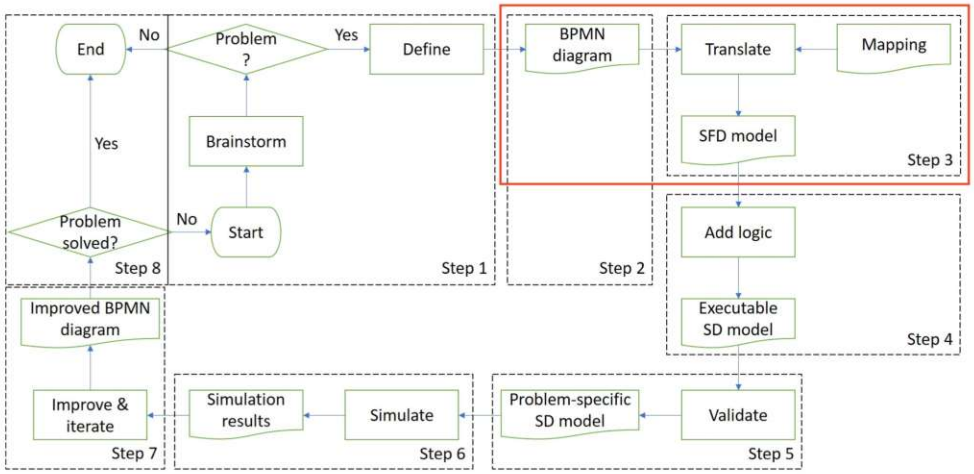


Figure 4.1: Visualized step-by-step guide to transform BPMN diagrams into SFDs [57]

4.1 Mapping Overview

In this section, we provide an overview of the mapping of BPMN-to-SD as proposed in [57, 58]. The authors explicitly state that the mapping as illustrated in fig. 4.2 is designed to serve as the "backbone" of an SD model, which may be further enhanced with other aspects such as energy consumption, waste production, worker workload, and equipment usage (step 4 in 4.1). However, the primary factor influencing the flows connected to the consumption/production of these aspects will be the flow(s) through the business activities, as derived from the original BPMN model.

BPMN and SD both use a sequential progression logic, which sets the foundation to establish the conceptual links between flow-and-gate notation in BPMN and stocks-and-flows in SD. Consequently, the authors in [57] suggest three fundamental patterns to map BPMN-to-SD (see fig. 4.2) which is further extended in [58]. Referring to fig. 4.2 we give a brief explanation on the transformation logic as described in [57], and in section 4.2 we go into further detail to explain the (adjusted) mapping we have implemented on Simplified Modeling Platform.

The first row in fig. 4.2 maps an activity of BPMN to SFD stocks, where a valve regulates the flow of resources from an active stock to a finished stock. Here, the authors present the concept of «$delay : T$» in the BPMN activities, given that an activity requires time to be finished.

The patterns that follow in row two and three in fig. 4.2 consider the join-trees (zero or more join-gateways) and split-trees (zero or more split-gateways) as an approach to the control flow between activities. Here, the trigger b is presented as a new concept, that the authors refer to as $Conflux_b$. In the BPMN side, this is analogous to a signal that transitions from the convergence point of the process elements to the point of divergence where the process branches into different paths. It functions as a switch that instructs

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

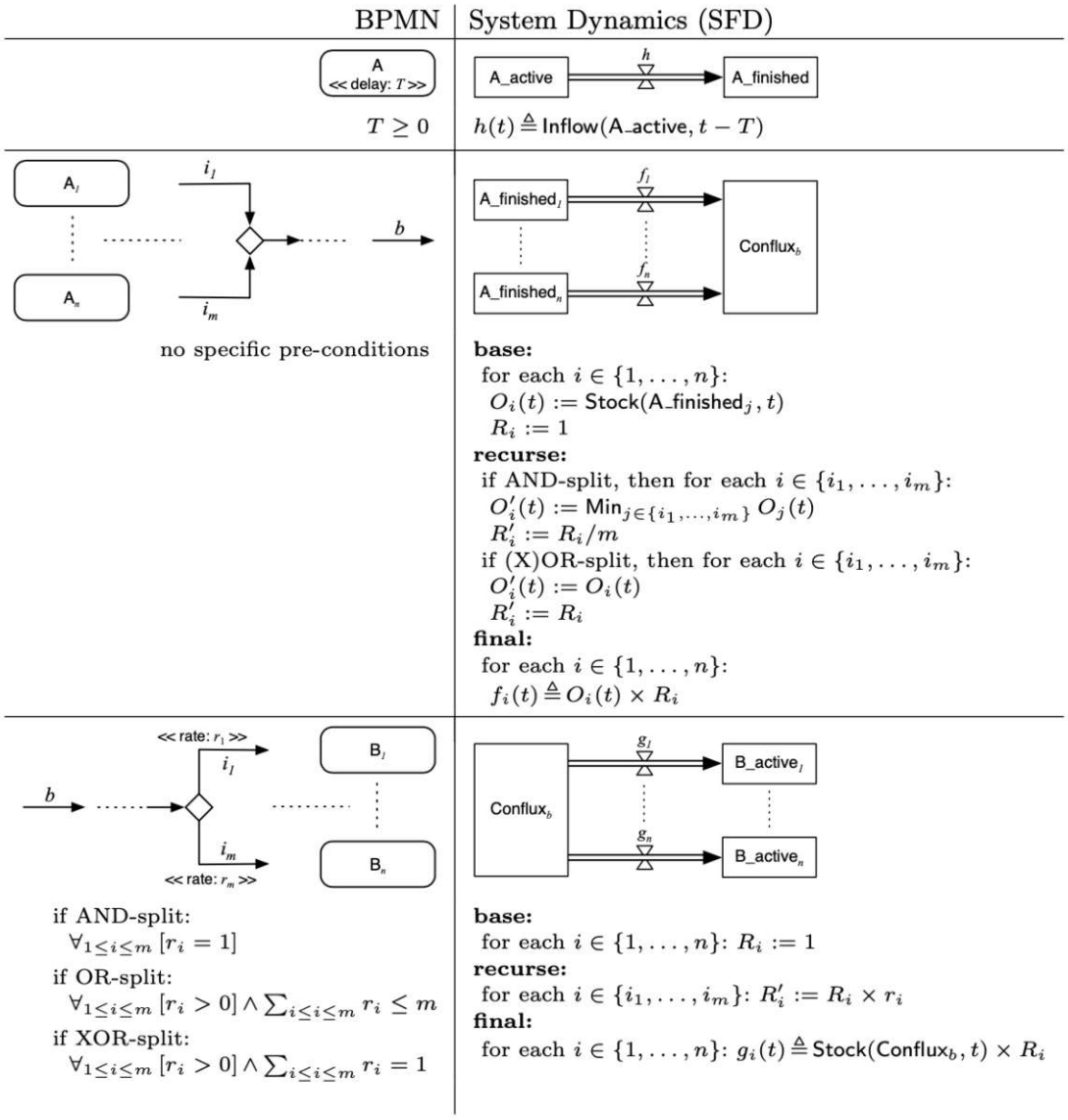


Figure 4.2: BPMN to SD mapping [57]

the process to transition from a particular phase to another. In [58] conflux is defined as a trigger with a *source* (join gateway, start event, end event or activity) and a *target* (split gateway, start event, end event or activity). This implies that the trigger must originate from a (final) join and be directed towards a (first) split. Nevertheless,the join-tree and split-tree can be "empty", which is why the source and/or target of the conflux can also be an activity or a start/end event.

To summarize, for a join-tree composed of sources $A_1, \ldots, A_n$, where n can take any positive value, we observe the SFD pattern with $A\_finished_i$ stocks and a flow f_i to $Conflux_b$. For a split-tree with targets $B_1, \ldots, A_n$, for any value of n, the SFD pattern

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

is created with A_active_i stocks and flows g_i originating from $Conflux_b$.

Depending on the type of gateways (inclusive, exclusive, parallel) in the join-/split-trees, the requirements for the flows in the SFD that are defined in the lower sections of rows two and three are still work in progress and will be explained in a more detailed way in future works. Therefore, they are out of our focus for our study.

In the next section, we define the scope of the implemented mapping in Simplified Modeling Platform, based on [57, 58], as well as provide the justification behind our slight adjustment to this mapping structure.

4.2 BPMN-to-SD transformation algorithm adjusted

While the mapping between BPMN and SFD summarized in the previous section is still in progress and to be enhanced in the future, in this section we explain the mapping algorithm that was used during the development of the tool. From the mapping defined in fig. 4.2, we added start/end events transformation and decided to adjust the transformation of join- and split-trees to stocks instead of the concept of "conflux". Therefore, we provide a semantic description for the new mapping. As part of the evaluation, we integrate these explanations while transforming some common BPMN patterns, extracted from [53]. Both, the semantic description of the mapping table and the transformed patterns was validated by (BPMN and SD) experts. The selected experts are:

Quan Zhu: SD Expert, Senior Researcher of Circular Supply Chain and Digitalization.

Henderik A. Proper: BPMN Expert, Full Professor in Enterprise and Process Engineering in the Business Informatics Group at the TU Wien.

We present an overview of the adjusted mapping structure as visualised in fig. 4.3. To describe what each transformation indicates, we provide the following semantic explanations. We leave the details about the definitions of the flow rates out of this study.

Row 1: Start Event to Source:

Start Event (BPMN) represents the beginning of a process and is mapped to a *Source* element (SFD), which indicates the starting point or input location from which resources or elements enter the system. Both elements serve as the starting points in their respective systems, indicating where processes begins and resources initially enter the system.

Row 2: End Event to Sink:

End Event (BPMN) represents the termination of a process and is mapped to *Sink* (SFD), which indicates the location where resources or elements exit the system, so concluding their journey within the model. Both elements indicate the conclusion point of a process in their respective modelling systems, indicating where elements are released from the system.

Row 3: Activity to (Active & Finished) Stocks and Flow:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

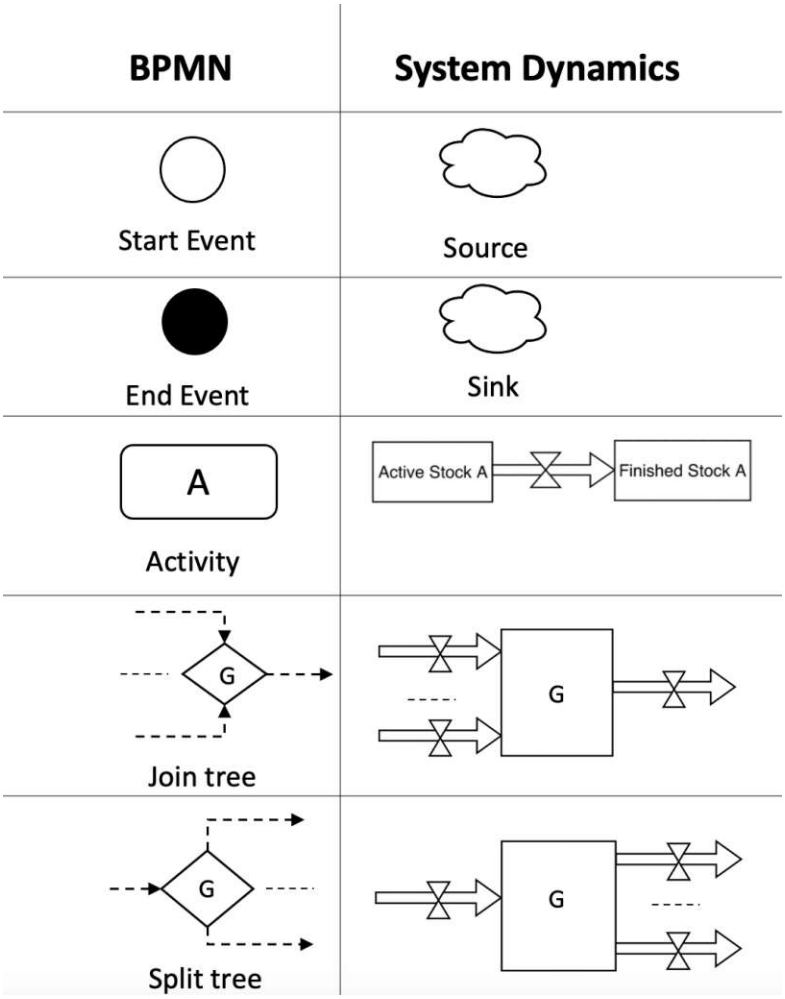


Figure 4.3: Overview of BPMN to SD mapping, adjusted from [57]

An *Activity* (BPMN) represents a task or work that is performed. Since an activity must have a minimum duration of T 0 from its start to its completion, this needs to be considered when mapping it to an SFD [57]. In SD, this translates into an *Active Stock* where resources are being processed while the activity is in progress, and a *Finished Stock* where resources end up after the process is complete. The *flow*arrow between them shows the rate of process completion. [57] define the flow to be regulated by a valve h, which is influenced by the input of A in such a way that it releases the complete input from time t - T. However, we keep the mathematical definitions of the flow out of this study and leave the further specific definition of the flow rate to further studies.

Rows 4 and 5: In BPMN, *Gateways* are used to control the flow of processes, either by splitting them into multiple paths or by converging multiple paths into one. [57] introduced the concept of "conflux," which plays a crucial role in transitioning between different phases of a process modeled both in BPMN and SD. The "conflux" is identified

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

as a critical point where a join-tree (a structure that brings together multiple process flows into a single point) meets a split-tree (a structure that diverges a single process flow into multiple paths). This pivotal point handles the transition from processes converging from multiple paths to those diverging into multiple new paths. Essentially, the conflux ensures that all incoming processes are accounted for before the process can split and continue along new paths. The mapping ensures that each gateway's fundamental purpose is preserved while adapting it to the systemic flow and stock manipulations inherent in SD modeling.

In this study we decide to simplify this transformation to simplify the implementation of BPMN-to-SD transformation on the web-based Simplified Modeling Platform. This approach suggests to map join and split trees to finished stocks. Similar logic has been used by [24] where he maps BPMN to Petri Nets. An XOR split-gateway in BPMN guides the flow to only one of the available paths, depending on a condition that is assessed during runtime. In Petri nets, this is denoted by a single transition that connects to many conditional points. However, the condition determines that only one option can be chosen, so resembling the exclusive decision-making process in BPMN. The XOR join gateway in BPMN selects one of the incoming flows to proceed with the operation. In the context of Petri nets, this situation is represented by multiple places converging into a single transition. This means that the process can continue once any of the preceding tasks is finished, effectively capturing the non-deterministic nature of the XOR-join, where the flow continues as soon as one incoming path is available.

Similarly, the "AND" split-gateway represents the simultaneous execution of several pathways, indicating that all paths originating from the split must progress concurrently. In the context of Petri nets, this concept is depicted by a transition that, once being triggered, enables all subsequent activities at the same time by depositing tokens in all outgoing spots. This means that all branches of the process can run simultaneously, reflecting the meaning of concurrent task execution in BPMN. However, it is not clear exactly how it will be handled to wait until the slowest task is done, in order to proceed with the next step of the process. Similar logic is used for AND join-gateway.

This reflects an adjustment of the initial mapping, in how the model handles information flow and decision points, which is why we received the validation/feedback of BPMN expert (Henderik A. Proper) and SD expert (Quan Zhu). Below we provide a summary of the comparison and an explanation on how the transformed SFD can be impacted.

Conflux Approach (Original):

- The conflux specifically distinguishes between points where processes converge (join-tree) and where they diverge (split-tree).
- It captures the dynamics of transitions between multiple process flows, emphasizing the complexity and the conditions under which these transitions occur.

Mapping Join-/Split trees to Finished Gateways:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Simplifying the model by using a uniform stock type for both joins and splits makes the model easier to build, understand, and use, particularly for those less familiar with the complexities of process modeling.

- Simplification could speed up the transformation process from BPMN to SFD, making it more efficient to generate models.

By mapping join and split trees to a finished stock, the simplified version represents both convergence and divergence points uniformly as places of accumulation (finished Stocks). For split-trees that would simply mean that the finished resources are accumulated into a certain stock before being distributed to other locations in the system.

In terms of join-trees, this would mean that the resources coming from precedent parts of the system will have to "wait" on the "Finished Stock G" until all possible flows have been concluded.

In the scope of this study, we disregard whether it's an "AND", "OR", "XOR" gateway and leave it to be examined in future work. However, as part of the semantics validation process, we got overall feedback from our selected SD expert whether this gateway transformation explanation can be valid for SD. The overarching feedback stated that, the suggested mapping of the Gateways, is valid for XOR gateways. Allocating resources directly from a 'Finished Stock' requires careful consideration of the rates and conditions under which resources are released to various processes, particularly when transforming complex gateway behaviors like "OR" or "AND" decisions. The latter, are more complicated and need further examination to be able to preserve the semantic meaning during the transformation. However, based on his experience with SD models, the most common situation in dynamics simulation is the XOR decision. When transforming business process models to SD diagram, one of the noticeable differences is that BPMN can provide more details on the workflow, due to its operational nature. SD on the other hand, is a modeling language that has a more strategic nature, which explains why the "XOR" decision is the most common example in dynamics simulation.

Our goal in the study is to provide clear, high-level overviews of process flows; therefore, the simplified transformation approach was considered to be more adequate for the implementation. However, for purposes that require deep analysis and optimization of business processes, the "conflux" approach might be more convenient. This can be considered in future studies.

4.2.1 Transforming common BPMN patterns to Stocks-and-Flows

To take the semantic validations for the proposed mapping to a further step, we provide in fig. 4.4 and fig. 4.5 a list of common patterns in BPMN and their transformation to System Dynamics, considering the mapping described in fig. 4.3.

Top Row, fig. 4.4 : Start Event to Activity

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

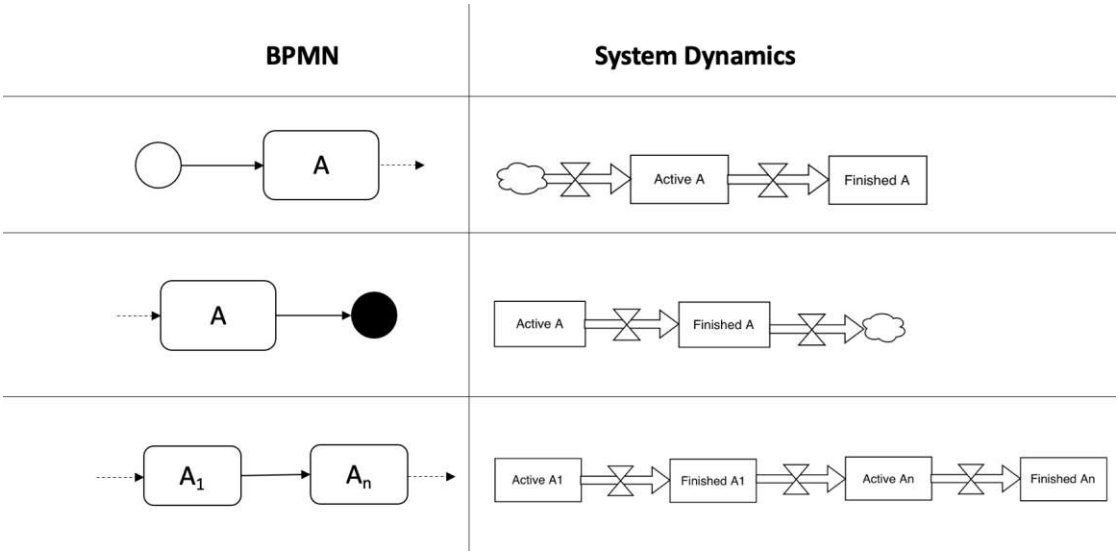


Figure 4.4: Common BPMN Patterns transformed to SFD - Part 1/2

The circle denotes a start event (BPMN), leading to a task 'A', representing the beginning of a process. The corresponding SFD pattern shows an inflow into 'Active A' stock, which denotes the resources currently being processed. The outflow from 'Active A' to 'Finished A' represents the completion of the task.

Middle Row, fig. 4.4: Task to End Event:

In BPMN A task 'A' leads to an end event (black circle), representing the end of the process. The pattern in SFD consists of resources in 'Active A' that are being processed, and then moved to 'Finished A' once they have reached their final state. The end event in BPMN is represented by an outflow from 'Finished A', signifying the resources leaving the system or process.

Bottom Row, fig. 4.4: Sequence of Tasks

A sequence of BPMN tasks, 'A_1' to 'A_n', shows a series of processes taking place one after another. For each task in BPMN, there's a corresponding 'Active' and 'Finished' stock in SFD. 'Active A_1' flows into 'Finished A_1, indicating the completion of the first task, and then the resources move to 'Active A_n for the next task in the sequence, ending in 'Finished A_n'.

Top Row, fig. 4.5: Split and Join

An activity (**A**) is both preceded and followed by a gateway (**G**). The preceding gateway is a split-tree based on conditions (**G__n**), while the following gateway signifies a join. The transformation to SFD takes the conditionality and convergence and represents them as feedback loops. The activity (**A**) is active, and once completed (**Finished A**), it influences the gateway (**G**). This can represent various scenarios such as iterations or dependencies, where the completion of **A** might affect the conditions at $\mathbf{G_m}$.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

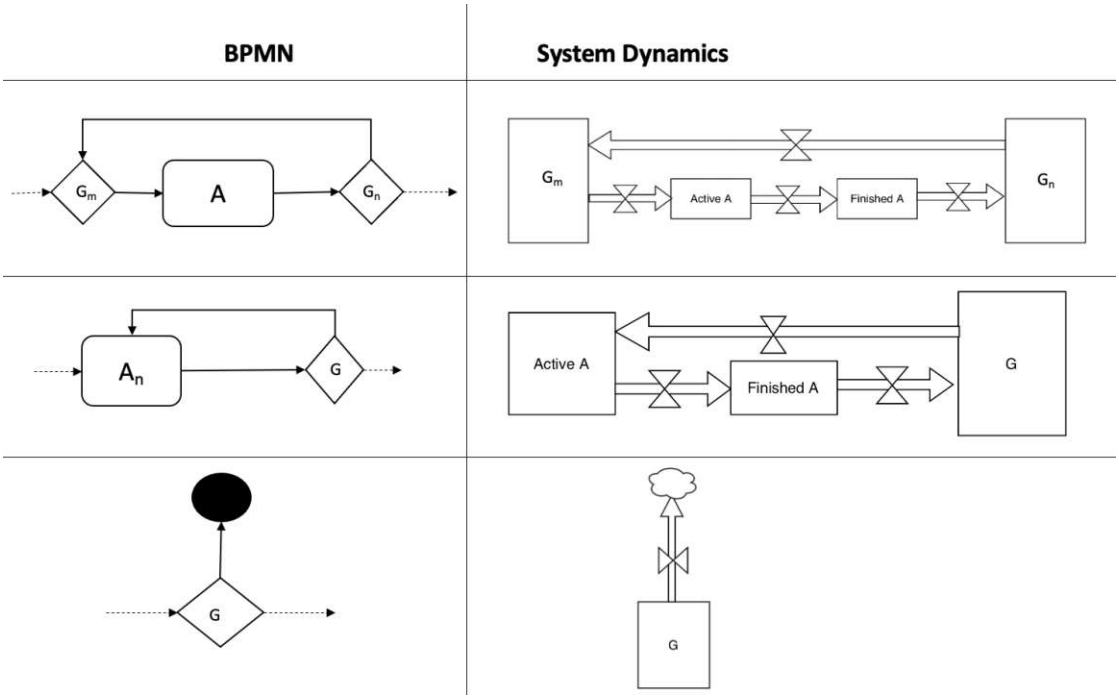


Figure 4.5: Common BPMN Patterns transformed to SFD - Part 2/2

Middle Row, fig. 4.5: Feedback Loop

Here, we see an activity (**A__n**) followed by a gateway (**G**), with a sequence flow looping back to the activity. This implies some form of iteration or repeat activity based on a condition at the gateway. The corresponding SFD pattern, based on our proposed mapping, consists of *Active A* and *Finished A* stocks with flows between them and a connection back to the gateway (**G**). This indicates that the activity can be active multiple times and its completion feeds back into the decision process at **G**.

Bottom row, fig. 4.5: Decision and Activity

A gateway (**G**) diverges into two paths, where one of them is represented by the "end event", representing an exclusive or inclusive decision that triggers different flows based on conditions. In the transformed SFD pattern, the finished stock (**G**) is connected to a cloud and a valve leading to a stock. This seems to symbolize a decision point with an environmental factor (the cloud can represent an external input or uncertainty) influencing the flow into an activity or state.

These transformations abstract the decision-making and iterative aspects of BPMN into the quantitative accumulations and rates of change in system dynamics. Each transformation provides a perspective that emphasizes the stock of "activities" and their "flow" rather than the specific decision logic in the BPMN. It's a more conceptual view that can help in understanding and modeling the dynamics of the system over time rather than the specific procedural logic.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Automating BPMN-to-SD transformation

In this chapter, we elaborate on the steps required to build a model-to-model transformation tool, implement the proposed algorithm in an existing tool, transition towards a more practical exploration, and present the limitations and challenges. The implementation is facilitated through the extension of the existing Simplified Modeling platform, which is a server-based solution designed to support models and their visual representations. The following sections delve into a comprehensive description of these practical aspects.

5.1 Requirements for BPMN-to-SFD transformation tool

Going back to our main research question - "*How can we design a tool that efficiently transforms BPMN models to SFD models?*" - this chapter focuses on the implementation of mapping between the models in an existing modeling platform (Simplified Modeling Platform), ensuring adherence to established scientific frameworks throughout the procedure. While conducting our research on how to automate BPMN-to-SD transformation, we considered two options:

- Build a web-based tool from scratch that supports model-to-model transformation.
- Use/extend the functionalities of existing modeling tools.

One of the key elements in designing a tool is to define the requirements. While defining the overall functionality of the Simplified transformation tool, the following criteria were used as a baseline:

- The tool must be able to support creating/editing BPMN and SFD models.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- The tool must be able to support the transformation of BPMN diagram to SFD.
- The tool must be able to have an efficient storage of the model.

For our Simplified transformation tool we define the following requirements, based on the above criteria:

1. **Requirement 1:** The tool must be able to support modeling diagrams in BPMN methodology, as defined in the simplified version of the BPMN metamodel in fig. 3.1.
2. **Requirement 2:** The tool must be able to support modeling diagrams in SFD methodology, as defined in the simplified version of the SFD metamodel in fig. 3.4.
3. **Requirement 3:** The tool must transform the elements on the BPMN diagram to the corresponding patterns in the Stock and Flow diagram (as defined in fig. 1.1).
4. **Requirement 4:** The tool should allow users to modify, add, delete items in the BPMN diagram.
5. **Requirement 5:** The tool should allow users to modify, add, delete items in the transformed Stock-Flow diagram.
6. **Requirement 6:** The tool must be able to save the produced models to be accessed by users anytime.
7. **Requirement 7:** The tool must be able to provide the possibility to extract the transformed SD model for further simulation in other tools.

Considering the above requirements, we decided to conduct further research on using the potential of existing tools to serve the purpose of our study. By extending the functionalities of an existing modeling tool, our goal was to achieve faster implementation in terms of required coding, leverage the benefits of community support, and contribute to the advancement of process modeling.

One alternative to integrate our BPMN-to-SD transformation was Eclipse ATL, often known as ATL Transformation Language. It is a language and toolkit specifically designed for model transformation. The ATL Integrated Environment (IDE) is built on the Eclipse platform and offers a variety of standard development tools, such as syntax highlighting and a debugger, to facilitate the development of ATL transformations. However, ATL is widely regarded as having a challenging learning process. In addition, although the Eclipse community makes significant efforts to guarantee stability and reliability, it is not unusual for users to come across bugs or issues. What also made us more open to explore other tools for our research study were (1) personal experience of Eclipse ATL instability and issues during Model Engineering course in TU Wien, and (2) ongoing initiatives of the Institute of Information Systems Engineering in TU Wien to explore other model-to-model transformation tools (i.e [18, 17]).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Therefore, we decided to extend the Simplified Modeling Platform; an existing web-based modeling tool which attempts a low code approach for modeling tool development. Simplified modeling platform applies academic expertise and supports modeling organisations. It is a cloud based modeling tool, supported by TEEC2 team, that offers the opportunity to define own model notations and create model diagrams.

Below we have listed few pros and cons of our decision to extend the Simplified Modeling Platform for BPMN-to-SD transformation.

Pros:

- It is considered a **low-code solution**, given that the basic modeling framework is provided by the TEEC2 team.
- Existing structure for model-to-model transformation (to some extent valid for our transformation).
- The platform can be extended by adding new features, functionalities or defining new modeling languages.
- Support and expertise of the TEEC2 team in organisational modeling tool development.
- Provides a support platform/community for possible bugs/issues/questions.
- The supervisor's direct contact with the TEEC2 team made it easier to establish the collaboration.
- Further definitions were done in the platform needed to extend the transformation structure(to support one or more source elements to be transformed to one or more target elements), which affected the timeline of the research project.

Challenges:

- The platform's ongoing expansion makes it prone to instability.
- The documentation is not sufficient/up-to-date, due to continuous updates with new features/functions. (This is to some extent compensated with the Simplified-Support platform).
- Initially, not all our transformation patterns (see fig. 4.2) were supported by the simplified transformation structure. Therefore, the platform needed to extend the transformation structure(to support one or more source elements to be transformed to one or more target elements), which affected the timeline of the research project.

Although the Simplified platform has yet to be fully refined to ensure a seamless user experience for our study, we were aware that there was a likelihood of encountering technical difficulties while creating our BPMN-to-SFD transformation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.2 Implementing BPMN-to-SFD transformation within Simplified Modeling Platform

This section covers the practical implementation of our model-to-model (BPMN-SD) transformation tool, through extending the existing Simplified Modeling Platform. After outlining the transformation process, creating meta models for both BPMN and SFD, and providing a mapping mechanism adapted from [57, 58], we now focus on the practical implementation within the Simplified platform. We explore the complexities of converting our abstract framework into an engineering artifact; we put our theoretical ideas into actual existence through applying the transformation algorithm. The Simplified Modeling platform is our chosen implementation environment; it offers a server-based solution that facilitates the creation of models and the visualisation of their dynamic representations [51]. While doing this, we take into account the difficulties, and results of the process of actualization. Finally, we provide our insights into the practical challenges that were faced when developing our tool for transforming BPMN to SFD.

5.2.1 Simplified modeling platform

In order to describe our implementation, we provide a thorough description of the Simplified modeling platform capabilities and functionality. This also provides a clear view of the existing capabilities of the tool, provided by TEEC2 team, to support the development of our (experimental) transformation tool. The Simplified Modeling Platform is a *cloud-based* application created for modeling and meta-modeling. It provides adjustable notations, a flexible user interface, and the ability to modify and visualise models in real-time. The web application facilitates collaborative design, allows for the use of numerous notations, and enables customisation of the user interface. It is suitable for both business and academic applications.

The following explanations are based on [50, 51] and the personal experience from working with the Simplified Modeling Plaform throughout this study.

Simplified Modeling Platform brings a low-code approach in modeling and meta-modeling. This platform is a product of prior experience with a research tool used to model Design and Engineering Methodology for Organisations (DEMO). It is appropriate for business and research applications due to its cloud-based development. Users (modelers, developers etc.) can use the platform due to its adjustable notations, relatively flexible user interface, and ability to provide visualisations and transformations [51].

There are two ways of engaging with the platform:

Easy Access: This is a convenient option to obtain basic public information without the having to log in. This provides several notations such as ArchiMate, DEMO, and BPMN, as well as the process of registering free user accounts. Nevertheless, it is not possible to share detailed modeling information using this approach.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Advanced Access: This is a secure method that we used through our study, in order to engage in more complex operations, that require authentication. The system uses a *web-socket connection*, enabling the transmission and reception of messages, as well as the sharing of messages to the user connected through their local machine. This is primarily intended for authorised developers seeking to generate customised user experiences; in our case we used the web-socket connection to the platform, in order to implement the automatic BPMN-SD transformation through our local machine.

The platform's architecture comprises a total of six layers, which are divided into two servers.

The **application server** is responsible for managing a majority of the duties and encompasses five layers:

1. **Interface Layer**: facilitates interaction with users. It refers to the visual elements and actions that we perceive and engage with.
2. **Message Layer**: is responsible for managing the transmission of information and instructions inside the platform.
3. **Processing Layer**: is where the platform executes the instructions or tasks it receives.
4. **Cache Layer**: is utilised to temporarily store frequently accessed data in order to enhance the speed of the platform.
5. **Persistence Layer**: is responsible for the storage and retrieval of data, ensuring its availability even when the platform is powered down.

The second server, which consists of a single layer, is the **Database Server**:

1. **Database layer**: is responsible for the long-term storage of all data. The platform's primary repository for all utilised information is the main storage area.

Messages transmitted through the asynchronous interface use a dynamic payload structure in JSON format. This feature enables customised settings for each individual message and facilitates the platform's dynamic development of structures throughout the platform's lifetime. Any kind of user interface can use this messaging system. Additional users (developers) have the option of using the same structure to establish a connection with the back-end engine for the purpose of executing certain operations regarding notations. When a client, which also behaves as a server, requests the platform to carry out specific actions depending on user input, it initiates a *gRPC*[1] call to the server platform. The

[1]Remote Procedure Call - a protocol that allows one program to request a service from another program on a networked computer.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
1   //language script BPMN
2
3   ScriptVersion01 Notation for BPMN version 2.0
4
5   typedef FLOWCONTROL ENUM (Exclusive, Inclusive, Parallel, "Default")
6   element "SplitGateway" SplitGateway202("Flow Control" FLOWCONTROL...)
7   connection "Sequence Flow" SequenceFlow202 from Activity202 to Activity202
8   diagram "BPMN" BPMN contains (...SplitGateway202...)
9   visual "Activity" of Activity202 on (*){initialsize(100, 50) group (0,0) scale { penwidth(2)...}}
10  visual "Sequence Flow" of SequenceFlow202 on (*)line {penwidth(2)...} end {initialsize(5,5)...}
11
```

Figure 5.1: Example of defining BPMN constructs using the Simplified notationscript.

server platform computes the data, transmits it to the client's user interface, and the client graphically displays it.

The modeling component of the back-end is specifically built to store both the model and metamodel of a notation or method. The architecture includes the construction of a meta-model through the establishment of a standardised notation for both the model and its visualisation. This process is helped by a script language defined by a specific grammar in the Simplified platform. The platform implements various levels of abstraction, and the process of transitioning between these levels can be achieved through either interpretation or compilation. This approach is exemplified with a fragment from the script language for the 'BPMN' notation we have defined for our study.

Note: The "..." are included to represent that more can be added to the notation.

Features and benefits

The Simplified Web UI is built using modular web pages, each designed for a distinct set of features. These pages provide modular function blocks, where each block includes both functional capabilities and a corresponding visual representation fig. 5.2. The two modules we use for our study are "Meta Modeler" and "Modeler" therefore we focus on explaining the features and functionalities on these modules.

Meta-modeler

The meta-modeler module in Simplified enables users to create custom notations and visualisations by specifying elements, connections, and their representations across numerous tools. Users have the chance to define modeling rules, and the custom notation they build can be easily utilised in all Simplified components [2]. The meta-modeler enables users to define their own notations through the online compiler (for testing single notations), or by uploading files for method notations, delete existing notation versions, as well as create new variables.

The notations are defined in a script language developed by TEEC2 team for this purpose. The constructs of the modeling method are grouped into two key groups: elements and connections. As per the latest published documentation in [49], the notation script has a grammar for the following notation concepts: element, connection, typedef, toolbox, virtual element, rule, table, visual, diagram, cube, behaviour, attributes. The grammar

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

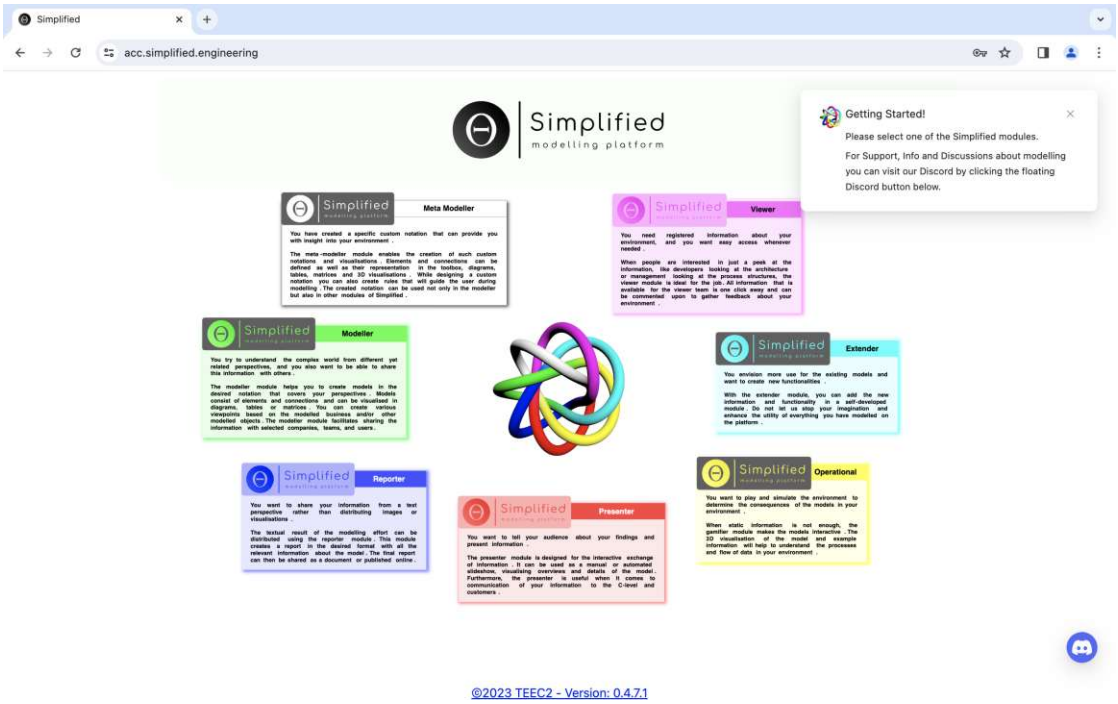


Figure 5.2: Simplified modeling platform: Homepage

of the Simplified platform aims to cover a comprehensive notation grammar that can represent all current and future notations. The notation is defined as the highest user meta-level in the Simplified platform [49].

The creation of this notation script is an iterative process that aims to provide a suitable grammar that covers all meta model descriptions. Throughout our own definitions in BPMN or SFD, there have been slight additions to the notation grammar. For instance, one thing that is in the future list of things to be added to the notation grammar in Simplified is “double lines”. These are currently not supported in Simplified notation script, which has resulted in a slight change of the visualisation of the “Flow” connection in our SFD constructs.

Modeler

The Modeler module in Simplified enables users to create models using their preferred notations. The features that this module provides are in line with our requirements presented in section 5.1. Notations are visualised in the toolbox, grouped into elements and connections that may be visualised in various formats such as diagrams, tables, or matrices. In our study we only use diagrams. The main characteristics of this module include automated saving, the ability to move elements by dragging and dropping, the option to name elements and connections, adding notes, resizing elements, initiating and terminating connections at any point on an element’s border, showing next most logical steps, creating anchors in connection lines, duplicating elements and connections, adding

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

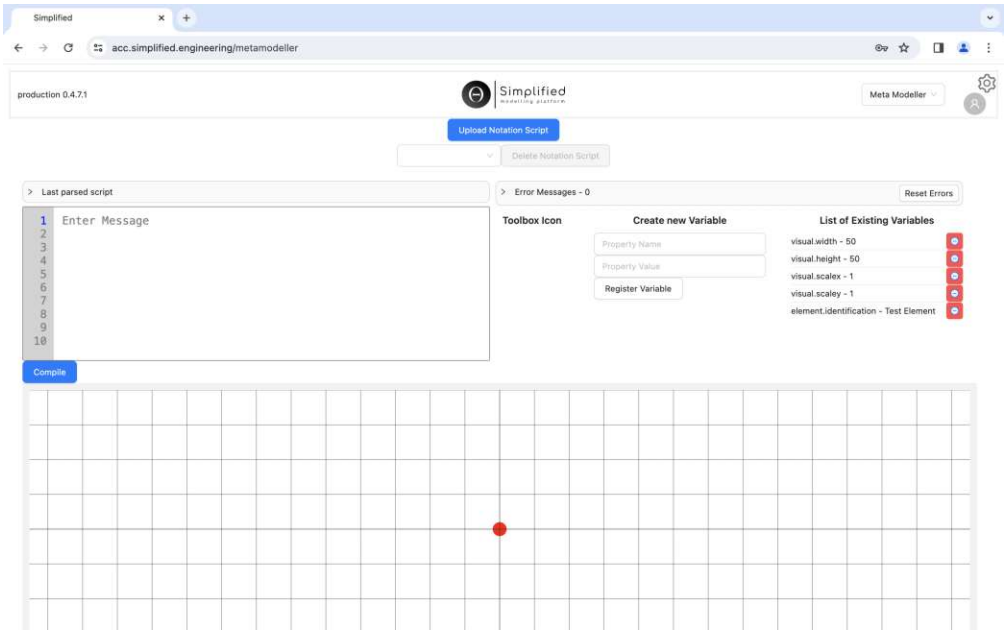


Figure 5.3: Simplified modeling platform: Meta-modeler

attributes to elements etc. Simplified has the capacity to accommodate several modeling approaches and their accompanying notations. The platform offers the option to either build new notations or enhance current ones.

As mentioned, gRPC is utilised to facilitate specific operations inside the system's architecture. This is implemented to enable remote servers to establish a connection with the central server and execute operations in the user interface. In our study, this is what we do: using our local client, which serves as a server, we communicate with the Simplified server to carry out the desired transformations and incorporate them into the target diagram. The communication of the servers is conducted via web-socket communication and the transformation program is developed in Golang.

5.3 BPMN-SD transformation program implemented in Simplified Modeling Platform

In previous sections we provided an outline of the architecture and existing functionalities of the Simplified Modeling Platform. We explained that, in order to automate BPMN-to-SD model transformation, we need to establish a web-socket connection to the platform server. Via web-socket connection to the platform's back-end we can perform specific actions related to notations, through the predefined methods to format or handle data in Golang. The Simplified team provided the framework to extend the server, an we implement the following steps in order to conduct model-to-model transformation through the Simplified Modeling Platform. Here we reference the GitHub repository [64].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The steps we followed to build the transformation tool are as below:

- **Step 1:** Define meta-models of the modeling languages.
- **Step 2:** Visualize shapes of the modeling language constructs via Simplified notation script.
- **Step 3:** Define mapping and transformation rules from source-to-target model.
- **Step 4:** Visualize the target diagram, defining the positions of the transformed elements in the target diagram.
- **Step 5:** Draw the source model in the modeling platform.
- **Step 6:** Execute transformation program (Input: source model in *Step 5*, transformation rules in *Step 4*).
- **Step 7:** Save and obtain the target model.

We explain in the rest of this section how we put the above steps into practice, in order to fulfill the aim of this paper: to automate BPMN-SD transformation.

Step 1: Define meta-models of the modeling languages

Each modeling methodology is characterised by its own set of constructs that define its vocabulary and entities. In chapter 3 we provide a comprehensive explanation of the meta-models and constructs used in Business Process Modelling Notation, and Stocks and Flow Diagrams through our stury. Meta-models are essential for maintaining the consistency and coherence of produced models. This is achieved by establishing appropriate structures, outlining their connections, and specifying the regulations for their application in Simplified Modeling Platform. We simplify the meta-models of BPMN and SD, in terms of limiting the scope of constructs that we take into account while transforming from one diagram to the other. Both meta-models have been defined and explained in chapter 3.

Step 2: Visualize shapes of the modeling language constructs via Simplified notation script.

The notation grammar embedded in the Simplified platform, makes it possible for us to use the notation script for the visual definitions of BPMN and SFD constructs. We defined the visual elements/connections based on the meta-models in sections fig. 3.1 and fig. 3.4. In this part we explain with examples how we used the notation script to define the constructs for BPMN and SFD.

In each of the notation scripts (BPMN and SD) we define the notation name and version. In the listing 5.1 we provide a snippet of the BPMN notation. The notation name and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Figure 5.4: Simplified Toolbox

version that we define here, is what appears as the label of the modeling language in the toolbox (see fig. 5.4) in the "Modeler" module.

Listing 5.1: BPMN script definition

```
//Start of the language definition
ScriptVersion01
Notation for BPMN version 2.0
```

In the BPMN definition, we define one enumerated data type as presented in the listing 5.2. This data type defines possible states of gateway elements. In SFD we haven't defined such specific types.

Listing 5.2: BPMN script notation - Type Definitions

```
typedef FLOWCONTROL ENUM (Exclusive, Inclusive, Parallel, ``Default")
```

As also previewed in fig. 5.4, the BPMN and SFD constructs are defined in two standard categories: connections and elements. Within the scope of our study we have only defined *sequence flow* as a connection in BPMN and *flow* in SFD. To define a connection, we provide the sourceElementReferenceName(the source element), and the targetElementReferenceName(the target element).

Listing 5.3: SFD Flow Notation

```
connection ``Sequence Flow'' SequenceFlow202 from Activity202 to
    Activity202
```

The BPMN notation contains the following elements: start event, end event, activity, split gateway, join gateway. For the gateways we have defined typedef "Flow Control" to specify the type of gateway. In the SFD notation the following elements are defined: active stock, finished stock, conflux, source, sink.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Listing 5.4: BPMN elements notation

```
element ''Start Event'' StartEvent202
element ''Activity'' Activity202
element ''SplitGateway'' SplitGateway202("Flow Control'' FLOWCONTROL)
element ''JoinGateway'' JoinGateway202("Flow Control'' FLOWCONTROL)
element ''End Event'' EndEvent202
```

The representation of models is done via diagrams in our case. A diagram is defined as a type of element that can hold other elements to support their visualisation. To ensure the accuracy of our model, we define what elements are permitted on a diagram, as presented in listing 5.5.

Listing 5.5: BPMN diagram notation

```
diagram ''BPMN'' BPMN contains (Activity202, StartEvent202,
   EndEvent202, SplitGateway202, JoinGateway202, SequenceFlow202)
```

We use the visual script to define the visualisation of elements and connections on the screen. When specifying the shape, basic geometric forms can be used such as line segments, arcs, polygons, rectangles, and ellipses. These basic shapes can be combined, grouped, scaled to create a single enclosed shape. A minimum size for the shapes can be determined by defining an initial size. Furthermore, we can provide specific parameters such as pen width, color etc.. Using the keyword switch, different visualisations of the defined types of an element can be made. To print any information (identification, name etc.) of the shape, print statement can be used.

To avoid redundancy, we provide visual notation example for one element and one connection. In the listing 5.6 we provide the example of switch case by defining a different visual for every defined type of the split gateway element (BPMN). Lastly, in the listing 5.7 we present an example of the definition of a connection; this time we specify the line, a shape in the center of the line, and the arrow in the end. Here, it is important to mention that, due to limitations on the visual notation script we couldn't visualize the "flow" connection type with double lines (see the graphics of SFD in section 3.2).

Listing 5.6: BPMN Visual Split Gateway element

```
visual ''SplitGateway'' of SplitGateway202 on (*){
    initialsize(50, 50)
    switch ("element''. "Flow Control")
        case ''Default'' {
            group (0,0) scale {penwidth(2) pencolor(74, 63, 59)
               rectangle(25, 0, 35, 35, 0, 45) penwidth(12) } }
        case ''Exclusive'' {
            group (0,0) scale {  penwidth(2)  pencolor(74, 63, 59)
               rectangle(25, 0, 35, 35, 0, 45) penwidth(3) line(16,
               16, 35, 33, 0) line(16, 33, 35, 16, 0)}}
         ...}
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

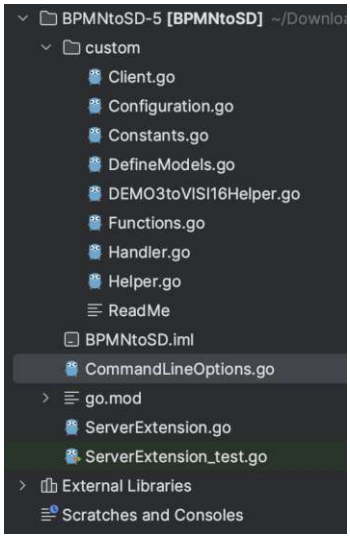


Figure 5.5: Go program to establish connection to the Simplified server - provided by TEEC2 team

Listing 5.7: BPMN Visual Flow connection

```
visual Flow of Flow1 on (*)
     line {penwidth(2) linestyle(solid)pencolor(0,0,0)}
    centre {initialsize(3,3) penwidth(2) pencolor(0,0,0) polygon(25,
       8, 3, 8, 0) polygon(25, -8, 3, 8, 60)}
    end {initialsize(2,2) fillcolor(0, 0,0) polygon(0, 0, 3, 5, 90)}
```

Step 3: Define mapping and transformation rules from source-to-target (BPMN-to-SD) model.

To extend the functionalities of Simplified, we first set up the (web-socket) connection to the Simplified server. The back-end of the Simplified server is developed in Go, also known as Golang [3] (see fig. 5.5). On a system architecture level, the interface layer of Simplified provides two ways to interact with the platform. We connect via the authenticated asynchronous web socket message interface, which allows authorized developers to exchange messages with the Simplified server. Our aim is to extend the capabilities of the existing Simplified platform to define BPMN and SFD modeling tools, as well as support the automatic transformation. To do that, we establish a WebSocket connection with the Simplified server. The TEEC2 team provided a Go project with pre-defined methods to support on setting the connection. To run the WebSocket-Client connection, credentials need to be defined. The development conducted through this research study is all supported in the Goland application.

Once the connection is set, we apply the transformation algorithm within the context of the Simplified server-extender. The primary goal is to translate the conceptual mapping

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

discussed in chapter 4, into a functional implementation that can be possibly integrated into the server-extender framework. This implementation serves as a crucial step in realizing the envisioned Stock-and-Flow diagram generation process within the server environment. The following step-by-step explanation provides the key components, methods, and considerations involved in successfully integrating the transformation into the Simplified server-extender. In this stage, we define the algorithm for transforming BPMN constructs into SFD constructs using pseudocode. The algorithm is based on the existing paper [58] and we do the necessary adjustments to make it simplify the development process on the server extender program. The relevant adjustments have been described and validated in chapter 4. To operationalise the transformation, we develop two key artifacts: (1) a generic algorithm, which is adapted from the methodology proposed by [58], and (2) an implementation of this algorithm within a Go programming environment, specifically designed to enhance the functionality of the Simplified server. The generic algorithm is designed to be platform-independent, enabling its implementation across various programming environments, regardless of the development platform. On the other hand, the second artifact is tailored to the predefined structures and methods built into the Simplified server. We provide the (adjusted) generic algorithm below. In order to ensure the compliance to scientific methodology, we do provide an expert feedback on the semantic meaning for the adjustments that were done (see chapter 4).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Algorithm 1 BPMN-SD transformation algorithm

```
# Define BPMN (source) model structure
Σ_bpmn = ⟨StartEvents, Gateways, Activities, EndEvents, SequenceFlows, SourceOf, TargetOf, Name⟩
# Define SFD model structure
Σ_sfd = ⟨SFDSources, SFDStocks, SFDSinks, SFDFlows, SFDNew⟩
# Initialize variables for SFD (target) model creation
L: StartEvents, Activities, EndEvents are the sets of SFD 'locations' where there may
be work 'waiting'
F: set of SFDFlows
SFDFlowSource, SFDFlowTarget : SFDLocations used to track SFDFlows
SFDName : String
# Initialize empty SFD (target) model
L := ∅
F := ∅
SFDFlowSource := []
SFDFlowTarget := []
SFDName := []
# Create SFD sources for each BPMN start event
for each s in StartEvents do
    l := SFDNew(L, SFDSources)
    SFDName[l] := Name(s)
end
# Create SFD for each BPMN end event
for each e in EndEvents do
    l := SFDNew(L, SFDSinks)
    SFDName[l] := Name(e)
end
# Create SFD stocks for each BPMN join gateway
for each j in JoinGateway do
    l := SFDNew(L, SFDStocks)
    SFDName[l] := Name(j)

end
# Create SFD stocks for each BPMN split gateway
for each g in SplitGateway do
    l := SFDNew(L, SFDSinks)
    SFDName[l] := Name(g)

end
# Create SFD stocks for each BPMN activity
for each a in Activities do
    l := SFDNew(L, SFDStocks)
    m := SFDNew(L, SFDStocks)
    SFDName[l] := "Active_" + Name(a)
    SFDName[m] := "Finished_" + Name(a)
    f := SFDNew(F, SFDFlows)
    SFDFlowSource[f] := l
    SFDFlowTarget[f] := m

end
```


Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
# Create flows for each sequence flow in BPMN
for each c in SequenceFlow do
    # Find SFD locations corresponding to source and target of the trigger
    sourceLocation := L[x] where SourceOf(c).id == L[x].id
    targetLocation := L[y] where TargetOf(c).id == L[y].id
    # Create flow in SFD
    f := SFDNew(F, SFDFlows)
    SFDFlowSource[f] := sourceLocation
    SFDFlowTarget[f] := targetLocation

end
```

The resulting SFD diagram is composed of the outputs L, F, and SFDName. It is important to emphasize that the previously described algorithm functions at the abstract or back-end level; the graphical layout is yet undefined.

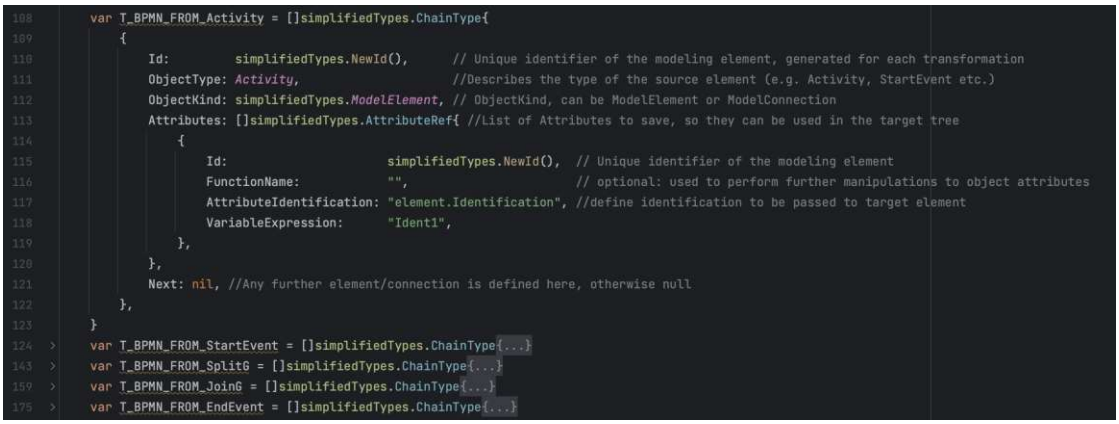
When we integrate the above algorithm on the Simplified server, we apply the back-end transformations, as well as establish the visual layout (step 4). Within this step we also explain the implemented algorithm on the Simplified server-extender, which represents the second artifact of our study.

The TransformOriginalToTarget() function in our Golang program interfaces with a WebSocket client to transform incoming messages (source diagram) from the server. This transformation process is executed by fetching data from the WebSocket stream, applying *a predefined set of transformation rules*, and producing the transformed messages as an output, while also handling any potential errors during the process. The predefined set of transformation rules use a predefined transformation structure, so-called "ChainTypes" to define the mappings between BPMN (FromChainType) and SFD (ToChainType) constructs. It is important to note that the transformation structure has been upgraded by the Simplified team to accommodate our specific needs for this study, enabling support for many-to-many transformations. Previously, the program was capable only of supporting one-to-one transformations. Below we elaborate the concept of the transformation structure on our program, including TransformRequest, FromChainType and ToChainType.

The transformation framework uses a declarative approach to describe how different types of components/patterns are to be converted into another model components/patterns. This is carried out by specifying the BPMN constructs in the FromChainType (fig. 5.6), and their target component/pattern in ToChainType (fig. 5.7). Following that, the transformation chains are run inside a transformation engine, so called TransformRequest (fig. 5.8), that interprets these rules, applies them to the BPMN (source) model, and produces the SD (target) model in line with the rules. This module-based approach, makes the system adaptable to modifications in transformation rules or model requirements.

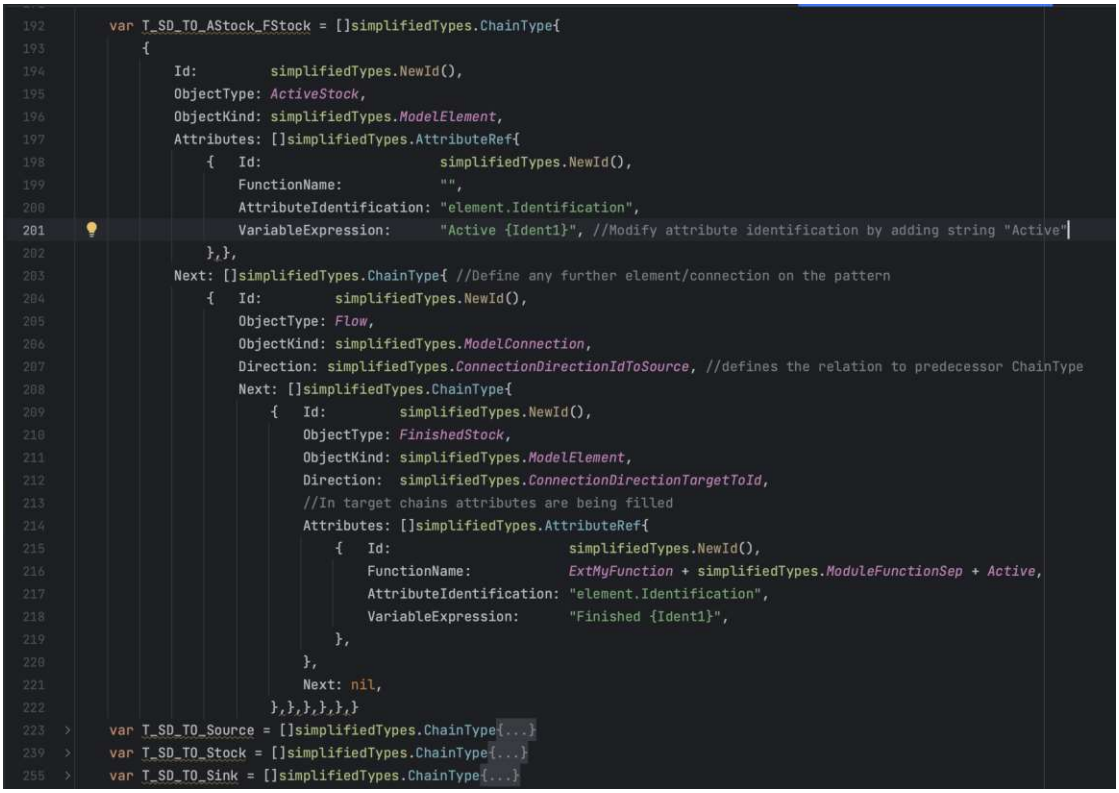
It is worth mentioning that we only define the transformations for BPMN elements to

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN



```
var T_BPMN_FROM_Activity = []simplifiedTypes.ChainType{
    {
        Id:         simplifiedTypes.NewId(),      // Unique identifier of the modeling element, generated for each transformation
        ObjectType: Activity,                     //Describes the type of the source element (e.g. Activity, StartEvent etc.)
        ObjectKind: simplifiedTypes.ModelElement, // ObjectKind, can be ModelElement or ModelConnection
        Attributes: []simplifiedTypes.AttributeRef{ //List of Attributes to save, so they can be used in the target tree
            {
                Id:                   simplifiedTypes.NewId(),  // Unique identifier of the modeling element
                FunctionName:         "",                       // optional: used to perform further manipulations to object attributes
                AttributeIdentification: "element.Identification", //define identification to be passed to target element
                VariableExpression:      "Ident1",
            },
        },
        Next: nil, //Any further element/connection is defined here, otherwise null
    },
}
var T_BPMN_FROM_StartEvent = []simplifiedTypes.ChainType{...}
var T_BPMN_FROM_SplitG = []simplifiedTypes.ChainType{...}
var T_BPMN_FROM_JoinG = []simplifiedTypes.ChainType{...}
var T_BPMN_FROM_EndEvent = []simplifiedTypes.ChainType{...}
```

Figure 5.6: Define BPMN Elements in Simplified Transformation Structure (FromChainType)



```
var T_SD_TO_AStock_FStock = []simplifiedTypes.ChainType{
    {
        Id:         simplifiedTypes.NewId(),
        ObjectType: ActiveStock,
        ObjectKind: simplifiedTypes.ModelElement,
        Attributes: []simplifiedTypes.AttributeRef{
            {   Id:                   simplifiedTypes.NewId(),
                FunctionName:         "",
                AttributeIdentification: "element.Identification",
                VariableExpression:      "Active {Ident1}", //Modify attribute identification by adding string "Active"
            },},
        Next: []simplifiedTypes.ChainType{ //Define any further element/connection on the pattern
            {   Id:         simplifiedTypes.NewId(),
                ObjectType: Flow,
                ObjectKind: simplifiedTypes.ModelConnection,
                Direction: simplifiedTypes.ConnectionDirectionIdToSource, //defines the relation to predecessor ChainType
                Next: []simplifiedTypes.ChainType{
                    {   Id:         simplifiedTypes.NewId(),
                        ObjectType: FinishedStock,
                        ObjectKind: simplifiedTypes.ModelElement,
                        Direction:  simplifiedTypes.ConnectionDirectionTargetToId,
                        //In target chains attributes are being filled
                        Attributes: []simplifiedTypes.AttributeRef{
                            {   Id:                   simplifiedTypes.NewId(),
                                FunctionName:         ExtMyFunction + simplifiedTypes.ModuleFunctionSep + Active,
                                AttributeIdentification: "element.Identification",
                                VariableExpression:      "Finished {Ident1}",
                            },
                        },
                        Next: nil,
                    },},},},},}
var T_SD_TO_Source = []simplifiedTypes.ChainType{...}
var T_SD_TO_Stock = []simplifiedTypes.ChainType{...}
var T_SD_TO_Sink = []simplifiedTypes.ChainType{...}
```

Figure 5.7: Define SFD constructs in Simplified Transformation Structure (ToChainType)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
transformRequest := simplifiedTypes.TransformModelRequest{
    FromModel:           originalModel.Id,
    ToModel:             targetModel.Id,
    FolderDestiny:       folder.Id,
    TransformConnection: translationConnection,
    FromChainTypeToChainType: []simplifiedTypes.FromChainTypeToChainType{
        {
            FromChainType:                 T_BPMN_FROM_Activity,
            ToChainType:                   T_SD_TO_AStock_FStock,
            CreateOnePerSameIdentification: true,
        },
        {
            FromChainType:                 T_BPMN_FROM_StartEvent,
            ToChainType:                   T_SD_TO_Source,
            CreateOnePerSameIdentification: true,
        },
        {
            FromChainType:                 T_BPMN_FROM_SplitG,
            ToChainType:                   T_SD_TO_Stock,
            CreateOnePerSameIdentification: true,
        },
        {
            FromChainType:                 T_BPMN_FROM_JoinG,
            ToChainType:                   T_SD_TO_Stock,
            CreateOnePerSameIdentification: true,
        },
        {
            FromChainType:                 T_BPMN_FROM_EndEvent,
            ToChainType:                   T_SD_TO_Sink,
            CreateOnePerSameIdentification: true,
        },
    },
```

Figure 5.8: Executing Transformation Request to Simplified Server

SFD patterns. The BPMN connections are not included in the above mentioned structure. We follow the logic explained in the algorithm (section 5.3) and create the connections in SFD, rather than transforming them through the TransformRequest. The algorithm sequentially examines each link in a BPMN model, retrieves the associated starting and ending components of the connection (referred to as the head and tail elements), and establishes a correspondence between these elements and their modified equivalents in the SD model. Subsequently, every BPMN connection (sequence flow) is converted into a fresh SD connection (flow), taking into account the corresponding mapped elements.

Step 4: Visualise target diagram

The procedure explained in *step 4* performs the transformations on the model element level. No visual elements/connections are created; therefore, the transformed patterns do not appear on the diagram. This means the xy coordinates, initial size, visual notation (shape) are not defined for the elements or connections in SFD. The transformed elements

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

only appear on the model repository, under the defined target SFD folder. Therefore we perform this step to define the visual elements so we can define the location on the diagram, the shape of the element, the size etc.

The CreateVisualTargetEl method illustrates a structured approach to visually transforming elements in model-driven architecture, specifically converting visual components of BPMN to a target (SD) notation. This method is a crucial component of a broader framework designed to ensure that there is a consistent and aligned visual representation between the initial and transformed models throughout the process. The CreateVisualTargetEl method iterates through each transformed element (output of the TransformRequest - step 4) and produces a the respective visual representation for the target model. Visuals are associated to their correspondent objects via a unique Id; i.e a visual element is associated to its respective element via ElementId.

The method of visualisation consists of the subsequent important stages:

Element Construction: For each transformed element, the method generates a new visual representation. The visual element is initialised with a unique Id and linked to the corresponding transformed model element. To ensure proper integration of the visual element into the target model environment, it is important to set essential attributes such as model IDs, diagram references, and notation-specific visual identifiers. Attribute Mapping: The BPMN model's visual attributes, like width, height, and position coordinates (X and Y), are mapped to the new visual element, maintaining the transformation's layout and size properties. The mapping guarantees that the visual representation in the SD model closely resembles that of the original BPMN model. Saving the Visual Element: After the visual element has been completely specified, it is stored in the model using the SaveModelVisualElement function. This function communicates with the database to ensure that the visual element is stored permanently.

Similar logic is followed for connections as well. For every connection, we create its visual, by mapping the relevant attributes.

Step 5: Draw/load the source model in the modeling platform.

We create a BPMN diagram in the website of the Simplified platform (Modeler module). After establishing the connection to the Simplified server, our Go program is used to generate an empty BPMN (source) model within the platform's current repository. During our development process, Simplified lacked the feature to construct a new model directly through the web platform; hence, we use the server extender in Golang to generate an empty BPMN model that we can use to create our BPMN diagram. Once the model is created in the repository, we can hierarchically create a folder and a diagram with a few clicks on the web.

Furthermore, given that the visual notations have been defined in the Simplified modeling platform, we can drag and drop elements/connections from the toolbox to the diagram, in order to draw our BPMN model for the transformation. An example of a BPMN diagram in the Simplified platform looks like in fig. 5.9.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

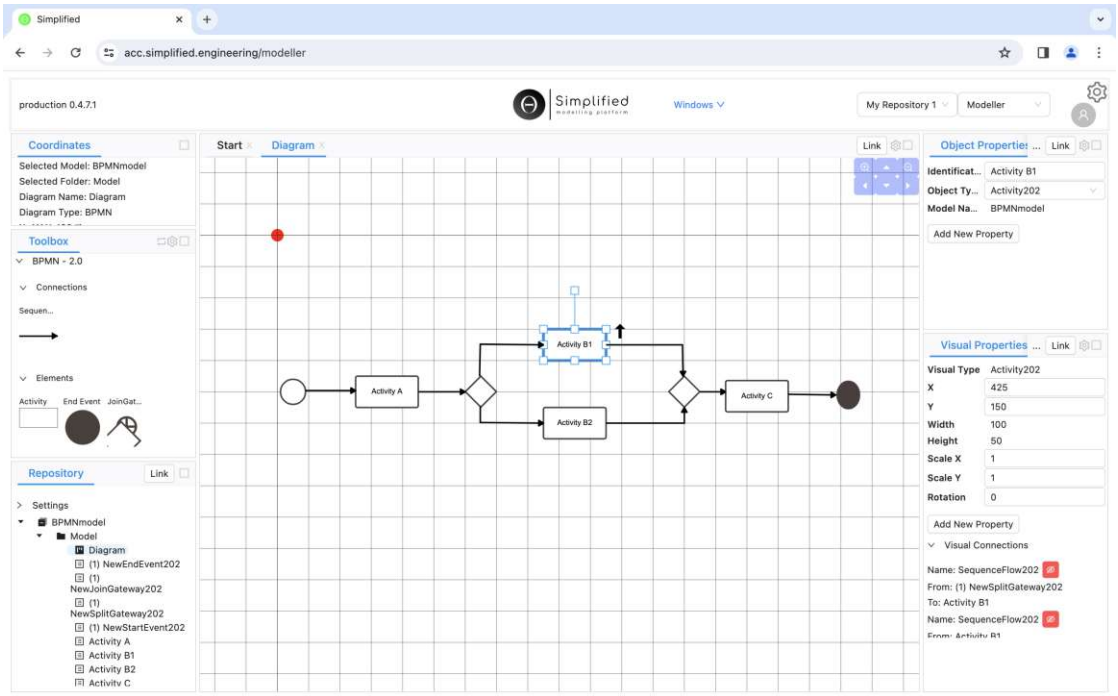


Figure 5.9: BPMN diagram visualised in Simplified platform

In the bottom left section, in the repository the model/s that we create are listed. The defined hierarchy in the repository starts with the the model at the highest level, then folder/s, which hold the diagram/s. The latter is where the visual elements/connections can be presented on the screen. Moving to the bottom right side, we can notice the "Visual Properties" section; here all the visual properties of the selected element/connection are shown. We use these properties in our transformation to define the coordinates, size etc. of the transformed components (explained in step 4). If we move up on the right side, the "Object Properties" are displayed; here we can add the description of the selected elements/connection as well as change the type.

Step 6 & 7: Run the transformation and save output model.

Step 6 and 7 are executed simultaneously. Once we run the transformation program, the client communicates with the Simplified server, retrieves the BPMN diagram with its elements and connections, performs the transformation rules, creates the visuals and then renders the output on the Simplified website, by showing the transformed diagram under the transformed model repository. The transformed model is saved and can be accessed any time, can be edited or exported in ".xml" file.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.4 Results

Throughout our study, we undertake several steps to develop a tool that transforms Business Process Model and Notation (BPMN) models into System Dynamics (SD) models. To ensure that our engineering effort is scientifically valid, we provide a thorough set of evaluation metrics in this section. When evaluating our model transformation tool, we define its "effectiveness" by assessing the tool's ability to achieve its intended objectives in transforming BPMN to Stock-and-Flow Diagram. Therefore, in this step we provide results of our tests for the transformation correctness and performance metrics.

Transformation Correctness: We evaluate whether each element and relationship from the BPMN model is correctly transformed into its corresponding element in the SFD model, using the following metrics:

1. **Syntactic correctness**: In chapter 4 we provide a thorough explanation of the mapping done between BPMN and SD components. The mapping used in the tool is mainly based on the existing papers [57, 58]; however, we include slight adjustments to it, to establish a smoother implementation on the Simplified Platform. Therefore, to ensure that the semantic meaning between BPMN and SD diagrams conforms to the standards of each modeling language, we seek for expert validation. Respectively, Prof. Dr. Henderik A. Proper (BPMN expert), and Dr. Quan Zhu(SD expert) provide their feedback on the transformation tool in regards to the semantic compliance of the transformed diagram. A thorough explanation was shared with them, explaining the how each component (within our study scope) would transform to Stock-and-Flow components. Then, a few common BPMN patterns(extracted from [53]) were presented, alongside their SFD transformation and the relevant explanations.

2. **Model Completeness**: To test model completeness and other metrics, we transform random examples (see "Data for BPMN Diagram" in fig. 5.10) of BPMN diagrams into SFD, using the developed tool. To determine model completeness, we count the number of transformed components from the tool and compare it to the output based on the generic algorithm we presented in section 4.2. The aim was to find out if there would be a count of missing or extra elements. For every example, the completeness rate was 100%, meaning that all expected SFD constructs were created from the tool.

3. **Error Rate:** Another control during the evaluation, was to monitor the frequency of transformation errors, whether identified by the tool or manually checking through the diagrams. No errors were identified by the running Go program, or via manual checks on the diagram.

Performance Metrics: When assessing the performance metrics of the transformation through Simplified platform, we considered analysing time metrics, scalability, and algorithm complexity, to offer a thorough comprehension of its effectiveness.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Data for BPMN diagram								Data for BPMN-SD transformation time (s)		
Start Event No.	End Event No.	Activity No.	Join Tree No.	Split Tree No.	Connections No.	Elements no.	Total components	Transformation time	Connect to server Time	Total Transformation Time
1						1	1	1,543	4,0	5,544
	1					1	1	1,479	4,0	5,479
			1			1	1	1,358	4,0	5,358
				1		1	1	1,279	4,0	5,279
		1				1	1	2,499	4,0	6,499
1	1				1	2	3	1,840	4,0	5,840
1	1	1			2	3	5	2,548	4,0	6,548
1	1	2			3	4	7	3,560	4,0	7,560
1	1	3			4	5	9	4,383	4,0	8,383
1	1	4			5	6	11	5,159	4,0	9,159
1	1	5			6	7	13	6,138	4,0	10,138
1	1	6			7	8	15	7,120	4,0	11,120
1	1	7			8	9	17	7,854	4,0	11,854
1	1	4	1	1	8	8	16	8,166	4,0	12,166
1	1	8			9	10	19	9,222	4,0	13,222
1	2	7	0	1	10	11	21	9,657	4,0	13,657
1	1	7	1	1	10	11	21	8,910	4,0	12,910
1	1	9			10	11	21	9,964	4,0	13,964
1	1	4	1	1	8	8	16	10,632	4,0	14,6327
1	1	10			11	12	23	11,307	4,0	15,307
1	1	4	1	1	9	8	17	11,2357	4,0	15,2358
1	2	10		1	15	14	29	15,261	4,0	19,261
1	1	15			16	17	33	18,539	4,0	22,539
1	1	20			21	22	43	25,831	4,0	29,831
3	3	14		1	22	21	43	22,397	4,0	26,397
3	4	21	1	2	34	31	65	44,457	4,0	48,457
4	5	28	2	3	46	42	88	70,171	4,0	74,171
4	5	35	3	4	12	51	63	103,729	4,0	107,729
4	7	45	3	4	21	63	84	140,631	4,0	144,631
4	6	46	3	4	24	63	87	146,820	4,0	150,820
4	6	48	3	4	72	65	137	153,258	4,0	157,258

Figure 5.10: Transforming BPMN-to-SD: Time metrics of example patterns

- **Time metrics**: Through our examples of BPMN diagram transformations, we monitored time taken to perform transformation, as we increased the number of constructs in the BPMN diagram. When running the transformation program, there is a fixed overhead time of 4 seconds for establishing a connection to the server where the transformation is processed. The total time taken to transform the BPMN diagram into the (SFD) target diagram, including the server connection time, increases significantly as the complexity of the BPMN components increases. We provide the time metrics in fig. 5.10.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- **Scalability:** Based on the results retrieved from the example transformations, it is evident that the duration of the transformation process is significantly affected by the quantity of BPMN components involved. There is a visible trend between the number of components in the source diagram and the amount of time it takes for the transformation to take place. The fixed duration for server connection (4,00 seconds) signifies a baseline overhead required for every transformation execution, regardless of the complexity of the source diagram. Also, as expected, the number of "activity" elements in BPMN considerably affects the transformation time, given that it is transformed into three "things" (two elements and a connection) on the target diagram. Therefore, if we transform two BPMN diagrams with the same overall number of components, but one of them has a higher number of activities, we can expect that diagram to take more time to be transformed.

- **Complexity of algorithm**: The algorithm's overall complexity tends to be linear (O(n)) in regards to the number of elements being transformed. However, it has the potential to become quadratic (O(k×m)), in regards to the number of connections (k) and the number of transformed elements (m). This is due to the nested structure of checking connections against all transformed elements to find the head and tail of the new connection to be created on the Stock-and-Flow Diagram. This emphasises the reliance on the quantity of connections and transformations, indicating specific areas where improvements in performance could be highly advantageous, especially if either k (number of connections) or m (number of modified elements) becomes considerable. Our investigation verifies the algorithm's capacity to handle a large number of elements, while also highlighting the significant influence of connection complexity.

  To discuss further about the algorithm complexity analysis, it's important to mention the server interactions throughout our transformation process. Server interactions include retrieving, storing, and manipulation of data from the Simplified (Modeling Platform) server. Typically, these steps exhibit linear behaviour related to the size or complexity of the data being processed. Therefore, we would consider a worst-case scenario like so: (1) server interactions are slow, and (2) complex diagram with a high number of (activity) elements and connections. Nevertheless, from our calculations, we see that, for a complex BPMN diagram (i.e 72 elements, 65 connections), our program would take 2 minutes to process the transformation. Therefore, we would still consider the algorithm on a relatively good performance.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

Recap of Research questions

In this chapter we provide a summary of the answers we give to the research questions.

As presented in the first chapter, we formulated our main research question as below:

Main Research Question: *How can we design a tool that effectively transforms BPMN models to SFD models?*

This research topic sets the foundation for a highly practical and solution-focused study, which focuses on the development of a transformation tool in the domain of process modeling. A comprehensive knowledge of both BPMN and SFD is essential in order to develop a practical tool that is beneficial for users in this field. Through our research process, we decide to extend the (existing) Simplified Modeling Platform (instead of developing a transformation tool from scratch), to implement the mapping between two modeling languages. We implement the defined methodology from the Simplified framework and update the tool with the below functionalities:

1. We add visual definition for BPMN and SD modeling languages in the tool.
2. We implement transformation methodology through the server-extender, to support automatic transformation.

The main artifact is the prototype tool that modelers can use to create, edit and transform BPMN diagrams to SFDs. The results of this study can help progress knowledge in BPMN-to-SD model transformation and could enhance the efficiency of such model-to-model transformations.

We also establish more specific, manageable sub-questions in order to answer the specified primary research topic.

Research Question 1: What are the existing BPMN and SFD concepts?

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

We answer this question in chapter 3, where we provide an overview on existing contructs of BPMN and SD through a literature review. Then we select the scope of constructs that are used for our study, to implement the mapping from BPMN to Stock-and-Flows. Both simplified metamodels were validated through BPMN expert (Henderik A. Proper) and SD expert (Quan Zhu).

Research Question 2: What are the steps to develop an (experimental) tool that performs automatic transformation of BPMN diagrams to SFDs?

Simplified Modeling Platform brings a low-code approach in modeling and meta-modeling. This platform is a product of prior experience with a research tool used to model Design and Engineering Methodology for Organisations (DEMO). It is appropriate for business and research applications due to its cloud-based development. Users (modelers, developers etc.) can use the platform due to its adjustable notations, relatively flexible user interface, and ability to provide visualisations and transformations [51]. Using the tool's predefined steps, we implement the following steps to develop the transformation in Golang program:

- **Step 1:** Define meta-models of the modeling languages.
- **Step 2:** Visualize shapes of the modeling language constructs via Simplified notation script.
- **Step 3:** Define mapping and transformation rules from source-to-target model.
- **Step 4:** Visualize the target diagram, defining the positions of the transformed elements in the target diagram.
- **Step 5:** Draw the source model in the modeling platform.
- **Step 6:** Execute transformation program (Input: source model in *Step 5*, transformation rules in *Step 4*).
- **Step 7:** Save and obtain the target model.

We put the steps into practice (documented in section 5.3 as part of the validation. The output is the developed program in Golang, which extends the Simplified server functionalities and implements the mapping defined in fig. 4.3.

Research Question 3: How can we formulate and implement an algorithm that optimally converts BPMN models to SFD models, as defined on the mapping provided in [57, 58]?

We answer this question in chapter 4. Following the guidelines in fig. 1.2, we first define the added value of our solution (environment). Then, given our decision to extend an existing tool to implement the idea, we define a step-by-step procedure (adjusted from the team providing the existing modeling platform) to develop a model-to-model transformation tool. As a next step, we design the algorithm for transforming the models

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

in the tool (design phase) and develop the prototype tool to evaluate the algorithm that we adjusted from [58] in the design phase. The DS methodology emphasizes the iterative construction and evaluation of artifacts, which in our case takes place in between the steps of developing the new artifact and evaluation. The reliability and validity of the transformation algorithm is carefully and iteratively evaluated by running the transformation in the Simplified Modeling Platform, using examples of common BPMN patterns and its SFD transformation [53]. Relevant metrics have been monitored, as well as (BPMN and SD) experts' feedback was provided on the semantic explanation of the transformed patters. Their critical comments were used to adapt and improve the tool functionality, as well as support the added value to the business process modeling field. These steps were summarized in fig. 1.2, adopted from [68, 35].

Research Question 4: To what extent does the existing Simplified modeling platform efficiently handle the defined model-to-model transformation steps, as measured by the mapping provided by [58]? What are the key technical limitations and challenges in implementing these transformations within the platform?

To ensure that our engineering effort is scientifically valid, we provide a thorough set of evaluation metrics in this section. When evaluating our model transformation tool, we define its "effectiveness" by assessing the tool's ability to achieve its intended objectives in transforming BPMN to Stock-and-Flow Diagram. We evaluate whether each element and relationship from the BPMN model is correctly transformed into its corresponding element in the SFD model, by checking Syntactic correctness (expert validation), model completeness and error rate. When assessing the performance metrics of the transformation through Simplified platform, we considered analysing time metrics, scalability, and algorithm complexity, to offer a thorough comprehension of its effectiveness. Results and further details to answer this question are elaborated in section 5.4.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 7

Conclusions

In this study, building upon the work of [57, 58], we document our initial steps towards an automated support to combine business process modeling and system dynamics. We extended the features of an existing modeling tool to provide automatic support in transforming BPMN diagrams to SD diagrams.

As a next step on the automation between BPMN and SD integration, additional features can be added to the transformation, in order to (1) improve the tool to check the logical, internal correctness of the models (BPMN and SFD), (2) extend further the semantic transformation of gateways (join-/split-trees) to include the logic for each type of gateway (AND, OR, XOR), (3) test the simulation of the (transformed) SD model in tools that support SD simulation.

Regarding the generalisability of the engineering artifact produced by our experiment, we do believe that the steps we followed to extend the Simplified Modeling Platform, can be applied to any other modeling languages to support automatic transformation.

Our engineering experiment focused on extending the capabilities of the Simplified Modelling Platform to enable automatic transformations between BPMN models and System Dynamics (SD) notation. During this procedure, we investigated whether it was possible to create a tool that would be efficient for this specific goal. The outcome artefacts - generic algorithm for BPMN-SD mapping and the implementation in the Simplified framework - effectively illustrated the possibility of such transformations, serving as a functioning proof of concept. This result not only confirms our main research question but also allows us to consider the applicability of our technique to different modelling languages. The experiment indicates that the approach used through our study may be modified for wider applications, thereby adding to the field by showing a path for similar engineering issues.

The transformation procedure, although effective, revealed some important considerations that are essential for practical implementations, particularly regarding scalability and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

architectural decisions. As models become larger and more complex, the current implementation on the client-side may not be sufficient, suggesting the need for server-side integration or alternate approaches to efficiently manage larger datasets. This aspect of our research underlines the need to align architectural decisions with the operational requirements of the tool, to ensure that performance doesn't decrease with scale. Additionally, the transformation process was shown to be considerably dependent on server interactions, indicating that in order to reduce performance time and improve speed, future implementations should optimise database access and potentially run closer to the database.

Lastly, during the effort to expand the functionalities of the Simplified Modelling Platform for the purposes of this study, we encountered considerable challenges, namely related to the practical limitations in the development environment. The current level of maturity of the Simplified Modelling Platform has been a an important factor in the progress of our research study. The main drawbacks of the system are mostly due to limited documentation, a challenging learning process linked with its tools, and limited technical assistance. In order to optimise the platform for future research and development, improvements are required to provide a more reliable, adaptable, and thoroughly documented setting.

Reflection on the use of the Simplified platform

In the process of extending the Simplified platform to implement a model-to-model transformation, both positive aspects and areas for improvement were encountered. The core concept of Simplified as a low-code platform for model transformation is, in theory, a commendable conceptual framework.

However, the extension process was not without its challenges. Usability issues and lacking documentation of the platform affected the development workflow. The absence of comprehensive documentation regarding the code-base posed a notable challenge. A more detailed code documentation system is of high importance for developers to understand and navigate the platform effectively during extension projects. Furthermore, instability during the development process was experienced, impacting the workflow and highlighting the need for a more stable development environment.

Throughout our experience of working with the Simplified Modeling platform, we encountered several challenges while developing the transformation algorithm in the provided framework. We list them below:

1. Lack of (updated) documentation of the tool features, visual notation, Go program of the server extender. 2. Learning curve for the self-defined visual notation & Go programming language. 3. Continuous need of the support of the Simplified team. Lack of independence throughout the programming procedure. 4. Server-extender instability. Same code would provide different results, or would work on the local server, but not for the server-extender. 5. Additional (structural) improvements were done: many-to-many transformations were implemented in the framework during our study. Previously, only one-to-one transformations were supported in the transformation structure.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

While the identified challenges present opportunities for improvement, it is essential to recognize the platform's innovative concept and its potential utility for modeling purposes. Recommendations for improvement include the implementation of a comprehensive and accessible developer documentation, enhancing the understanding of the platform's complex code-base. Also, usability enhancements, such as refining the development environment, are crucial for a smoother extension process.

To summarize, while the experience of extending the Simplified platform presented challenges, the recognition of its innovative concept and potential utility for modeling purposes is crucial. Addressing the identified areas for improvement can transform Simplified into an even more valuable tool for developers working on model transformations within a low-code environment.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] Eclipseatl, projects.eclipse.org/projects/modeling.mmt.atl, (Accessed on [21.92.2024])

[2] Simplified, https://acc.simplified.engineering, (Accessed on [01.12.2023])

[3] Go programming language (2023), https://go.dev

[4] Aagesen, G., Krogstie, J.: Bpmn 2.0 for modeling business processes. Handbook on Business Process Management 1: Introduction, Methods, and Information Systems pp. 219–250 (2015)

[5] van der Aalst, W.M.P.: Business process simulation revisited. In: Enterprise and Organizational Modeling and Simulation: 6th International Workshop, EOMAS 2010, held at CAiSE 2010, Hammamet, Tunisia, June 7-8, 2010. Selected Papers 6. pp. 1–14. Springer (2010)

[6] Abdelkafi, N., Täuscher, K.: Business models for sustainability from a system dynamics perspective. Organization & Environment **29**(1), 74–96 (2016)

[7] Aguilar-Savén, R.S.: Business process modelling: Review and framework. International Journal of production economics **90**(2), 129–149 (2004)

[8] An, L., Jeng, J.J.: On developing system dynamics model for business process simulation. In: Proceedings of the Winter Simulation Conference, 2005. pp. 10–pp. IEEE (2005)

[9] Andersen, D., Roberts, N., Deal, R., Garet, M., Shaffer, W.D.: Introduction to computer simulation: The system dynamics approach (1983)

[10] Azar, A.T.: System dynamics as a useful technique for complex systems. International Journal of Industrial and Systems Engineering **10**(4), 377–410 (2012)

[11] Bala, B.K., Arshad, F.M., Noh, K.M.: System dynamics

[12] Bala, B.K., Arshad, F.M., Noh, K.M., et al.: System dynamics. Modelling and Simulation **274** (2017)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[13] Banks, J.: Discrete event system simulation. Pearson Education India (2005)

[14] Barnett, M.W.: Modeling & simulation in business process management. Gensym Corporation pp. 6–7 (2003)

[15] Baskerville, R.L., Kaul, M., Storey, V.C.: Genres of inquiry in design-science research. Mis Quarterly **39**(3), 541–564 (2015)

[16] Binder, T., Vox, A., Belyazid, S., Haraldsson, H., Svensson, M.: Developing system dynamics models from causal loop diagrams. In: Proceedings of the 22nd International Conference of the System Dynamic Society. pp. 1–21 (2004)

[17] Bork, D.: bigER - A Visual Studio Code extension for conceptual and relational data modeling. `https://github.com/borkdominik/bigER` (2023), accessed: 2023-04-10

[18] Bork, D.: bigER - Data Modeling Tool for Entity-Relationship Diagrams. `https://github.com/borkdominik/bigER` (2023), accessed: 2023-03-25

[19] Brown, F.T.: Engineering system dynamics: a unified graph-centered approach, vol. 8. CRC press (2006)

[20] Burmester, L., Goeken, M.: Combining system dynamics and multidimensional modelling-a metamodel based approach. AMCIS 2008 Proceedings p. 332 (2008)

[21] Chang, L.C., Tu, Y.M.: Attempt to integrate system dynamics and uml in business process modeling. In: Proceedings The 23rd International Conference of the System Dynamics Society July. pp. 17–21 (2005)

[22] Coyle, R.: System Dynamics Modelling: A PRACTICAL APPROACH. CRC Press (1996)

[23] Coyle, R.G.: System dynamics modelling: a practical approach. Journal of the Operational Research Society **48**(5), 544–544 (1997)

[24] Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and analysis of bpmn process models using petri nets. Queensland University of Technology, Tech. Rep pp. 1–30 (2007)

[25] Doebelin, E.: System dynamics: modeling, analysis, simulation, design. CRC Press (1998)

[26] Doyle, J.K., Ford, D.N.: Mental models concepts for system dynamics research. System dynamics review: the journal of the System Dynamics Society **14**(1), 3–29 (1998)

[27] Erasmus, J., Vanderfeesten, I., Traganos, K., Grefen, P.: Using business process models for the specification of manufacturing operations. Computers in Industry **123**, 103297 (2020)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[28] Fayoumi, A., Loucopoulos, P., Fayyoumi, A.: Hybrid Enterprise Modelling: Integrating Modelling Mechanisms for Socio-Technical Systems Analysis and Design **7**(1), 6–13 (2014)

[29] Forrester, J.: Counterintuitive behavior of social systems. technology review (1971)

[30] Forrester, J.W.: Industrial dynamics—after the first decade. Management science **14**(7), 398–415 (1968)

[31] Forrester, J.W.: Industrial dynamics. Journal of the Operational Research Society **48**(10), 1037–1041 (1997)

[32] Guizzardi, G., Wagner, G.: Can bpmn be used for making simulation models? In: Enterprise and Organizational Modeling and Simulation: 7th International Workshop, EOMAS 2011, held at CAiSE 2011, London, UK, June 20-21, 2011. Selected Papers 7. pp. 100–115. Springer (2011)

[33] Hall, A.D., Fagen, R.E.: Definition of system. In: Systems Research for Behavioral Sciencesystems Research, pp. 81–92. Routledge (2017)

[34] Hammer, M., Champy, J.: Business process reengineering. London: Nicholas Brealey **444**(10), 730–755 (1993)

[35] Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. Management Information Systems Quarterly **28**(1), 6 (2008)

[36] Jifeng, W., Huapu, L., Hu, P.: System dynamics model of urban transportation system and its application. Journal of Transportation Systems engineering and information technology **8**(3), 83–89 (2008)

[37] Jnitova, V., Joiner, K., Efatmaneshnik, M., Chang, E.: Modelling workforce employability pipelines for organisational resilience **13**, 1–19 (2021)

[38] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., Valduriez, P.: Atl: a qvt-like transformation language. In: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications. pp. 719–720 (2006)

[39] Kahani, N., Bagherzadeh, M., Cordy, J.R., Dingel, J., Varró, D.: Survey and classification of model transformation tools. Software & Systems Modeling **18**, 2361–2397 (2019)

[40] Kechagias, E.P., Gayialis, S.P., Konstantakopoulos, G.D., Papadopoulos, G.A.: An application of a multi-criteria approach for the development of a process reference model for supply chain operations. Sustainability **12**(14), 5791 (2020)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[41] Kožusznik, J., Štolfa, S., Ježek, D., Kuchař, Š.: Comparison of system dynamics and bpm for software process simulation. In: Digital Information Processing and Communications: International Conference, ICDIPC 2011, Ostrava, Czech Republic, July 7-9, 2011, Proceedings, Part II. pp. 1–15. Springer (2011)

[42] Lagarda-Leyva, E.A., Acosta-Quintana, M.P.G., Portugal-Vásquez, J., Naranjo-Flores, A.A., Bueno-Solano, A.: System dynamics and sustainable solution: The case in a large-scale pallet manufacturing company. Sustainability **15**(15), 11766 (2023)

[43] Lane, D.C.: Diagramming conventions in system dynamics. Journal of the Operational Research Society **51**(2), 241–245 (2000)

[44] Leaver, J.D., Unsworth, C.P.: System dynamics modelling of spring behaviour in the orakeikorako geothermal field, new zealand. Geothermics **36**(2), 101–114 (2007)

[45] Levitt, T.: Marketing myopia, harvard business review, july-august, 1960. Harvard Business Review pp. 45–56 (1960)

[46] Luna-Reyes, L.F., Andersen, D.L.: Collecting and analyzing qualitative data for system dynamics: methods and models. System Dynamics Review: The Journal of the System Dynamics Society **19**(4), 271–296 (2003)

[47] Lyneis, J.M.: System dynamics for business strategy: a phased approach. System Dynamics Review: The Journal of the System Dynamics Society **15**(1), 37–70 (1999)

[48] Mak, T.M., Chen, P.C., Wang, L., Tsang, D.C.W., Hsu, S.C., Poon, C.S.: A system dynamics approach to determine construction waste disposal charge in hong kong. Journal of Cleaner Production **241**, 118309 (2019)

[49] Mulder, M.A.T., Mulder, R., Bodnar, F.: Towards a demo description in simplified notation script. In: Enterprise Engineering Working Conference. pp. 53–70. Springer (2022)

[50] Mulder, M.A.T.: Enabling the automatic verication and exchange of DEMO models. Ph.D. thesis, Sl: sn (2022)

[51] Mulder, M.A., Mulder, R., Bodnar, F., van Kessel, M., Gomez Vicente, J., et al.: The simplified platform, an overview. Modellierung 2022 Satellite Events (2022)

[52] Object Management Group (OMG): BPMN 1.1 Specification (2008), `https://www.omg.org/spec/BPMN/1.1/`

[53] Object Management Group (OMG): BPMN 2.0 Specification (2011), `https://www.omg.org/spec/BPMN/2.0/`

[54] Pereira, J.L., Freitas, A.P.: Simulation of bpmn process models: Current bpm tools capabilities. In: New advances in information systems and technologies. pp. 557–566. Springer (2016)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[55] Pereira, J.L., Silva, D.: Business process modeling languages: A comparative framework. In: New Advances in Information Systems and Technologies. pp. 619–628. Springer (2016)

[56] Pourbafrani, M., van der Aalst, W.M.P.: Hybrid business process simulation: Updating detailed process simulation models using high-level simulations. In: Research Challenges in Information Science. pp. 177–194. Springer (2022)

[57] Proper, H.A., Zhu, Q., Ravesteijn, J.P.P., Gielingh, W.: Adding dynamic simulation to business process modeling via system dynamics (1), 2–12 (2023)

[58] Proper, H.A.: Transformation algorithm (December 2023), complements the paper on Adding Dynamic Simulation to Business Process Modeling via System Dynamics

[59] Qi, C., Chang, N.B.: System dynamics modeling for municipal water demand estimation in an urban region under uncertain economic impacts. Journal of environmental management **92**(6), 1628–1641 (2011)

[60] Radzicki, M.J.: System dynamics and its contribution to economics and economic modeling. System dynamics: Theory and applications pp. 401–415 (2020)

[61] Randers, J.: Elements of the system dynamics method. Journal of the Operational Research Society **48**(11), 1144–1145 (1997)

[62] Richardson, G.P., Pugh III, A.L.: Introduction to system dynamics modeling with dynamo. Journal of the Operational Research Society **48**(11), 1146–1146 (1997)

[63] von Rosing, M., White, S., Cummins, F., de Man, H.: Business process model and notation-bpmn. (2015)

[64] Ruci, E.: BPMNtoSDTransformation – github repository (2023), `https://github.com/eraldaruci/BPMNtoSDTransformation`, accessed: 2024-05-12

[65] Shannon, R.E.: Simulation modeling and methodology. ACM SIGSIM Simulation Digest **8**(3), 33–38 (1977)

[66] Sterman, J.: System dynamics: systems thinking and modeling for a complex world (2002)

[67] Tulinayo, F.P.: Combining System Dynamics with a Domain Modeling Method. Oisterwijk: Uitgeverij BOXPress (2014)

[68] Vom Brocke, J., Hevner, A., Maedche, A.: Introduction to design science research. Design science research. Cases pp. 1–13 (2020)

[69] White, S.A.: Introduction to BPMN. IBM Cooperation **2**(0), 0 (2004)

[70] Wolstenholme, E.F.: System enquiry: a system dynamics approach. John Wiley & Sons, Inc. (1990)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN